基于混合离散粒子群优化的控制模式分配算法

曾裕钦[1],蔡华洋[1],周茹平[1],刘耿耿[1*],黄 兴[2],徐 宁[3]
(1. 福州大学计算机与大数据学院,福建福州 350108;2. 西北工业大学计算机学院,陕西西安 710072;
3. 武汉理工大学信息工程学院,湖北武汉 430070)

摘 要: 连续微流控生物芯片是生物化学实验自动化、微型化的革命性技术. 多路复用器的控制模式分配作为连续微流控生物芯片自动化设计的关键环节之一,是难的NP(Non-deterministic Polynomial)优化问题. 现有工作采用粒子群优化算法求解控制模式分配问题存在过早陷入局部最优解、收敛速度慢以及算法稳定性差的缺点. 为此,本文提出一种连续微流控生物芯片下基于混合离散粒子群优化的控制模式分配算法. 首先,为了加快算法收敛速度及避免过早陷入局部最优解,提出了离散的自适应区域搜索策略. 其次,通过基于样例的社会学习机制提高了算法的稳定性. 然后,采用等距抽值的方式筛选出自适应区域搜索策略中重要参数的最佳组合,以进一步提高分配方案的质量. 最终实验结果表明,所提算法在多路复用器中阀门使用数量上平均优化了19.01%,在算法稳定性上提高了29.18%,且在现实的生化应用中有良好的性能表现.

关键词: 连续微流控生物芯片;控制模式分配;离散粒子群优化;样例学习;自适应区域搜索
基金项目: 国家自然科学基金(No.61877010)
中图分类号: TP391 **文献标识码:** A **文章编号:** 0372-2112(2024)08-2836-14
电子学报 URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20221166

Hybrid Discrete Particle Swarm Optimization Algorithm for Control Pattern Assignment

ZENG Yu-qin[1], CAI Hua-yang[1], ZHOU Ru-ping[1], LIU Geng-geng[1*], HUANG Xing[2], XU Ning[3]
(1. *College of Computer and Data Science*, *Fuzhou University*, *Fuzhou*, *Fujian* 350108, *China*;
2. *School of Computer Science*, *Northwestern Polytechnical University*, *Xi'an*, *Shaanxi* 710072, *China*;
3. *School of Information Engineering*, *Wuhan University of Technology*, *Wuhan*, *Hubei* 430070, *China*)

Abstract: Continuous-flow microfluidic biochip is a revolutionary technology for automation and miniaturization of biochemical experiments. As one of the key links in the automatic design of continuous microfluidic biochips, the control pattern assignment problem of multiplexers is an NP-hard combinatorial optimization problem. The existing particle swarm optimization algorithm for control pattern assignment problem has the disadvantages of falling into local optimal solution prematurely, slow convergence speed, and poor stability of the algorithm. In this paper, control pattern assignment algorithm based on hybrid discrete particle swarm optimization for continuous-flow microfluidic biochips is proposed. First, in order to accelerate the convergence speed of the proposed algorithm and avoid falling into a local optimum prematurely, a discrete adaptive region search strategy is proposed. Second, the stability of the proposed algorithm is improved by a sample-based social learning mechanism. Third, the optimal combination of the important parameters in the adaptive region search strategy is selected by equidistant sampling to further improve the results. The final experimental results show that the proposed algorithm optimizes the number of valves by an average of 19.01%, and improves the stability of the algorithm by 29.18%, and then performs well in practical biochemical applications.

Key words: continuous-flow microfluidic biochips; control pattern assignment; discrete particle swarm optimization; sample learning; adaptive region search
Foundation Item(s): National Natural Science Foundation of China (No.61877010)

收稿日期:2022-10-14;修回日期:2022-12-07;责任编辑:王天慧
*通讯作者:刘耿耿

1 引言

在传统的生物化学实验中,由于实验的每个环节都需要人为进行操作,使得整体的工作流程非常复杂繁琐,同时也对实验的正确执行以及昂贵试剂的节约带来了严峻的挑战. 因此,为了提高实验的执行效率以及减少试剂的使用量,连续微流控生物芯片应运而生,成为生物化学实验自动化、微型化的革命性技术[1~3]. 该芯片的基本结构由两个独立的逻辑层构成,分别为流层与控制层,其中流层内存在用于运输试剂的流通道,而控制层内存在用于传导压力的控制通道. 控制通道与流通道均采用弹性材料制作而成[4,5],在它们的交界处形成一个阀门. 为了打开和关闭阀门,外部压力首先通过控制端口连接到控制通道,然后气压通过控制通道传导至阀门. 通过这种方式向下挤压阀门的弹性体,以阻止流通道中试剂的流动. 当外部压力释放时,阀门的弹性体恢复原状,流通道再次打开以进行流体输送,因此阀门作为芯片的基本构件,其在连续微流控生物芯片中起着至关重要的作用. 同时为了发挥阀门的潜在作用,有关阀门的研究工作也开始出现. 例如,为了使一个控制引脚连接的多个阀门实现同步驱动,文献[6]提出了阀门的兼容性识别方法. 文献[7]提出了控制层布线算法,可在减少控制引脚数量的同时,保证阀门的同步驱动.

随着生物芯片的发展速度快于摩尔定律[8],一个硬币大小的微流控生物芯片已经能集成数千个阀门. 然而在一个大规模集成的微流控生物芯片中,需要通过大量的片外压力控制器来实现复杂的控制逻辑,增加了微流控分析设备的尺寸、功耗和成本. 因此,文献[9]和文献[10]采用多路复用器技术,用最少数量的控制引脚实现复杂的流体操作,以指数方式提高了芯片内复杂通道网络的处理能力. 在多路复用器中,控制通道与其上层的弹性通道在交界处形成控制阀门,控制阀门在气体的驱动下进行开关,从而控制流阀门的开关. 同时该多路复用器的有效性和优点在实际的生物化学应用中也得到了验证[11,12]. 另外,为了降低多路复用器的故障率和提高生物芯片的可靠性,文献[13]提出了基于汉明距离的切换序列优化算法,通过降低多路复用器中控制阀门的切换次数减少了控制阀门的磨损. 此外,文献[13]提出的基于异或的压力刷新算法防止了多路复用器中控制阀门切换过程出现的压力消退现象. 基于原始的多路复用器结构,文献[14]首次提出了一种新的多通道切换机制,实现了多个控制通道的同时切换,从而提高了芯片的执行效率.

在上述具有多通道切换机制的多路复用器中,为多路复用组合分配一个控制模式是至关重要的环节,其可以有效降低芯片的设计成本. 例如,文献[14]采用带有交叉变异操作的离散粒子群优化(Discrete Particle Swarm Optimization,DPSO)算法来求解一种具有较少控制阀门数量的控制模式分配方案. 其中,粒子群优化(Particle Swarm Optimization,PSO)算法是一种简单、高效的群智能算法,其设计了一个粒子群,所有的粒子都向当前粒子搜索到的历史最优解和整个种群的最优解分别学习,从而在多次迭代后找到全局最优解[15]. 目前,PSO算法已经被广泛应用于各类优化问题中[16~20]. 例如,文献[21]和文献[22]分别证明了PSO算法应用在超大规模集成电路以及连续微流控生物芯片物理设计问题中有良好的效果. 同时,为了增强PSO算法的性能,许多PSO算法的变体被提出. 例如,文献[23]通过引入局部搜索策略避免过早陷入局部最优解. 文献[24]采用两阶段搜索策略和多种群的策略提高了粒子探索能力. 文献[25]通过引入社会学习(Social Learning,SL)机制,使粒子在学习过程中保持了较好的多样性. 为了更好地解决实际问题,针对不同的问题将PSO算法与不同的策略进行融合,形成混合粒子群优化算法. 文献[26]将熵值作为扰动因子动态调节惯性权重,同时引入自适应学习策略,形成了基于直觉模糊熵的混合粒子群优化算法. 文献[27]基于遗传算子重新设计了PSO算法的离散更新机制,并结合精练策略提出了适用于集成电路布线问题的混合离散粒子群优化算法. 在无人机路径规划问题中,文献[28]通过融合模拟退火算法提出了一种新的混合粒子群优化算法,以快速地为无人机规划更高质量的路径. 为了加快算法的收敛速度,文献[29]在多种群的PSO算法中引入区域搜索策略. 基于区域搜索策略,文献[30]在文献[19]的基础上提出了自适应区域搜索(Adaptive Region Search,ARS)策略,所提算法在收敛速度和收敛精度上都表现出较好的性能.

然而,文献[14]中的控制模式分配算法仅采取简单的交叉和变异操作. 其中,粒子通过变异的方式保持种群的多样性,通过交叉操作向全局最优粒子或当前粒子历史最优位置学习,以实现种群的进化. 由于粒子的学习对象过于单一,种群在进化过程中易过早陷入局部最优解,且一旦陷入局部最优解就难以跳出,导致无法获得质量更高的控制模式分配方案. 此外,种群中各个粒子与全局最优粒子的差异大小不同,当较差的粒子向种群全局最优粒子学习时,较大的学习跨度使粒子在学习过程中移动太快,易错过全局最优解,从而导致算法稳定性较差. 因此,基于上述相关工作的分析,本文提出了一种连续微流控生物芯片下基于混合离散粒子群优化的控制模式分配算法,主要贡献如下:

(1)针对连续微流控生物芯片多通道切换机制下的控制模式分配问题,提出了混合社会学习机制、离散

粒子群优化和自适应区域搜索策略的控制模式分配算法，称为SLDPSO-ARS-CPA算法，与同类算法相比，该算法能获得控制阀门数量最少的控制模式分配方案.

(2)为了加快算法的收敛速度，同时避免过早陷入局部最优解，提出了一种更适合本文离散粒子群优化的自适应区域搜索策略，使得算法在有限的迭代次数下，得到一种高质量的控制模式分配方案.

(3)设计了一种基于样例的社会学习机制(Social Learning Discrete Particle Swarm Optimization，SLDPSO).该机制通过降低粒子学习的波动性，从而有效提高了算法的稳定性，保证所求控制模式分配方案的质量.

(4)采用了等距抽值的方式对算法进行参数调整，以期获得一组自适应区域搜索阶段的最佳参数组合，进一步提高了算法的性能.

2 问题描述

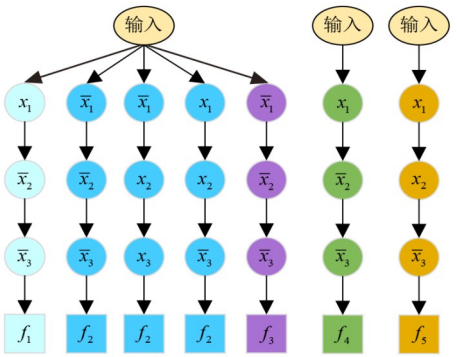
在控制模式分配问题中，假设某多路复用器包含N个控制通道，若此时给定了该多路复用器中用于同时切换这些控制通道的组合(即多路复用组合)，则需要初始化$2^{\lceil \log_2 N \rceil}$个控制端口.此外，这些控制端口能相应生成$2^{\lceil \log_2 N \rceil}$个不同的控制模式.本文将给定的多路复用组合以矩阵的方式呈现，如图1所示.

$$
\begin{array}{cc}
 & \begin{matrix} A & B \end{matrix} \\
\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{pmatrix} & \begin{matrix} \bar{x}_1\bar{x}_2\bar{x}_3 & x_1\bar{x}_2x_3 \\ x_1\bar{x}_2\bar{x}_3 & x_1x_2x_3 \\ \bar{x}_1x_2x_3 & x_1\bar{x}_2\bar{x}_3 \\ x_1x_2\bar{x}_3 & x_1x_2\bar{x}_3 \end{matrix} \\
\begin{matrix} f_1 & f_2 & f_3 & f_4 & f_5 \end{matrix} &
\end{array}
$$

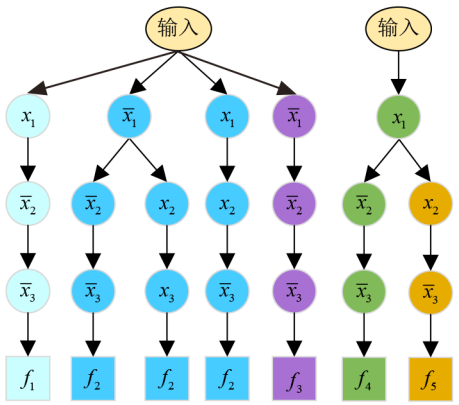
图1 多路复用组合

该矩阵由4个不同的多路复用组合构成，其中每行代表一个多路复用组合，每列代表一个控制通道.根据该矩阵，首先，可以看出在控制逻辑中，总共有5个流阀门(即f_1~f_5)通过控制通道连接到核心输入.其次，总共有4个多路复用组合用来实现控制逻辑中的多通道切换机制(如：矩阵第4行的多路复用组合01001表示与流阀门f_2、f_5连接的控制通道能够同时进行状态切换).此外，右侧的控制模式分配方案A和B中，分别有4种不同的控制模式被选中并分配给矩阵中的4个多路复用组合.例如在分配方案A中，控制模式$x_1\bar{x}_2\bar{x}_3$被分配给了多通道组合10010.如果控制模式$x_1\bar{x}_2\bar{x}_3$被激活，那么与流阀门f_1、f_4连接的控制通道状态将同时发生变化，从而来驱动流阀门f_1、f_4的状态为打开或关闭.

为了评估控制模式分配方案的质量，本文首先采用文献[14]中的方式将分配方案A和B转化为逻辑树，其次通过逻辑化简和逻辑森林构造得到相应分配方案的资源使用情况.例如，图2(a)是分配方案A中流阀门的逻辑树结构，其中根节点与叶子节点分别代表核心输入和流阀门，每个内部节点代表一个控制阀门，每条边代表一条连接两个控制阀门的控制路径，叶子节点是流阀门.对图2(a)所示的逻辑树结构进行逻辑化简、逻辑森林构造后得到如图2(b)所示的结果，该逻辑森林最终包含19个控制阀门、26条控制路径.相反，分配方案B中的逻辑森林最后仅需10个控制阀门以及16条控制路径，如图3(b)所示.



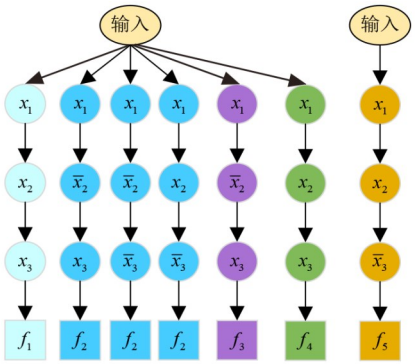
(a) 流阀门初始的逻辑树



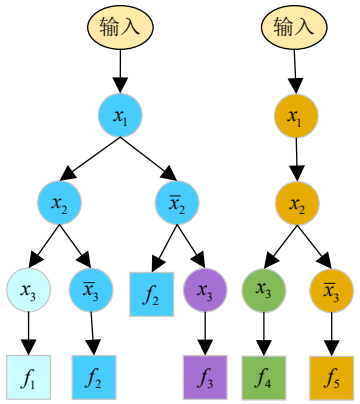
(b) 逻辑树化简、逻辑森林构造的结果

图2 分配方案A对应的流阀门的逻辑简化

上述分配方案A和方案B分别构造的逻辑森林布线情况如图4(a)和图4(b)所示，其中6个控制端口x_1、$\bar{x}_1$、x_2、$\bar{x}_2$、x_3、$\bar{x}_3$连接到压力源，以此来控制5个流阀门的状态为打开或关闭.通过图中可以看出，将不同的控制模式分配给不同的多路复用组合会导致在控制逻辑中使用的控制阀门数量有较大差异.控制阀门的使用数量越少意味着形成控制阀门所需要的通道长度越短，从而使得芯片的制造成本越低.同时，控制阀门数量和通道长度的减少能够有效降低后续布局布线阶段的设计复杂度，使得最终的芯片设计总成本降低.因此，需要设计一种方法来获得使多路复用器中所需阀门数量



(*a*) 流阀门初始的逻辑树



(*b*) 逻辑树化简、逻辑森林构造的结果

图3　分配方案B对应的流阀门的逻辑简化

更少的控制模式分配方案.

控制模式分配方案对应的逻辑树在经过逻辑化简和逻辑森林构造后即可获得控制阀门的使用量. 用$n_{i,j}$表示逻辑森林中第i棵树的第j层中代表控制阀门的节点个数. 因此,为了最小化控制阀门的数量,该问题优化目标的表示如下:

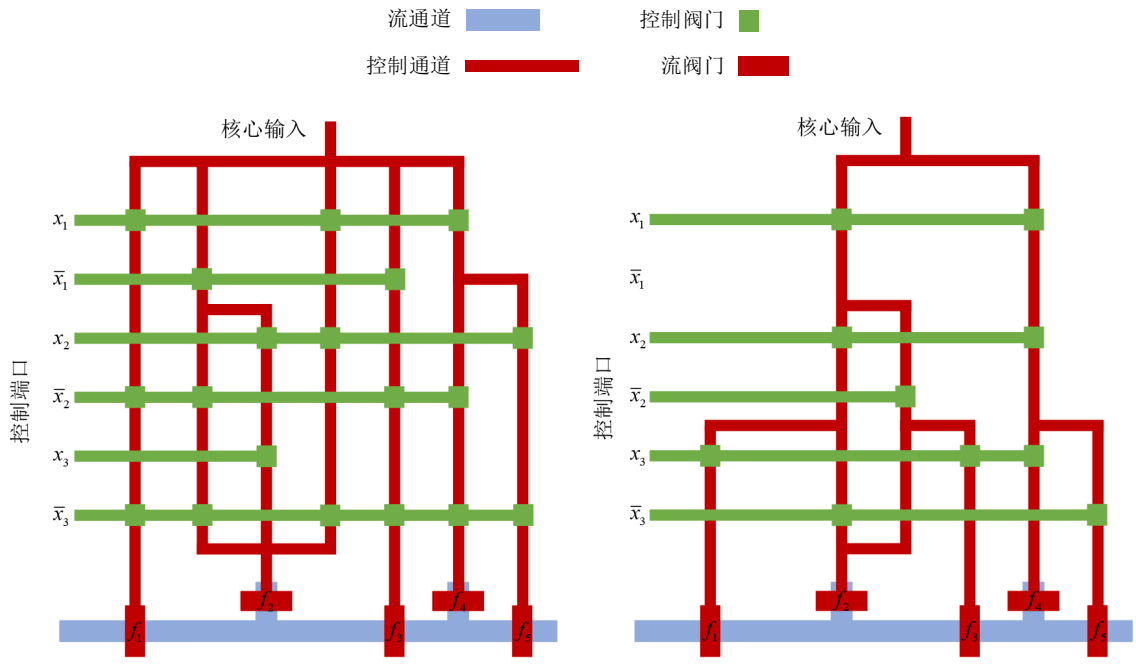
$$\text{minmize}\sum n_{i,j},\ 1\leqslant i\leqslant I \text{ and } 1\leqslant j\leqslant J_i$$

其中,I是逻辑森林中逻辑树的数量,J_i是第i棵逻辑树的层数.

3　本文算法

3.1　算法流程

本文提出的SLDPSO-ARS-CPA算法由社会学习离散粒子群优化算法SLDPSO和自适应区域搜搜策略ARS两个阶段交替进行. 算法整体流程如图5所示. 首先以上述的多路复用矩阵作为输入. 其次在每轮迭代过程中,通过SLDPSO阶段更新所有粒子的位置. 然后将粒子按适应度值进行排序,位置较好的P个粒子依次在ARS阶段执行T次区域搜索. 由于这P个粒子接近全局最优解的概率较大,因此对位置较好的P个粒子执行区域搜索能使算法快速收敛,且能有较大概率跳出局部最优解. 在多次迭代后最终得到需要控制阀门数量更少的控制模式分配方案. SLDPSO-ARS-CPA算法的伪代码见算法1.



(*a*) 分配方案A逻辑森林对应的布线图　　(*b*) 分配方案B逻辑森林对应的布线图

图4　逻辑森林对应的布线

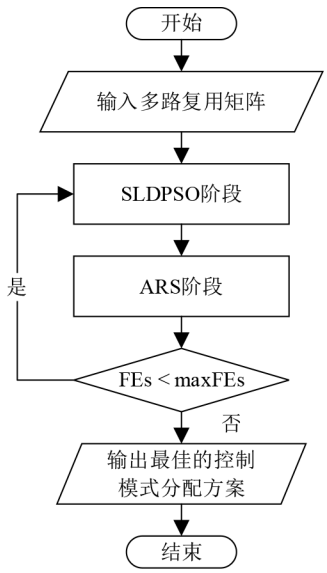


图5 SLDPSO-ARS-CPA算法流程

算法1 SLDPSO-ARS-CPA算法

输入: 多路复用矩阵

输出: 控制模式分配方案

1. 初始化规模为N的种群
2. 计算粒子适应度值, 并将粒子按适应度值升序排序
3. WHILE FEs<maxFEs DO
4. FOR i=1 TO N-1 DO
5. 粒子p_i根据其速度更新位置
6. 粒子p_i根据个体或社会认知更新位置
7. 计算粒子p_i的适应度值
8. END FOR
9. 将粒子按适应度值升序排序
10. FOR i=1 TO P DO
11. FOR t=1 TO T DO
12. 粒子p_i进行区域搜索
13. IF 搜索到更好位置 THEN
14. 更新粒子p_i的位置
15. END IF
16. END FOR
17. 修改粒子p_i搜索半径
18. END FOR
19. END WHILE

3.2 SLDPSO阶段

3.2.1 粒子编码方式

粒子p_i的位置编码为

$$\mathrm{cp}(1),\mathrm{cp}(2),\cdots,\mathrm{cp}(i),\cdots,\mathrm{cp}(j),\quad 0\leqslant i\leqslant j\leqslant M \tag{1}$$

$$\text{s.t.}\forall i,j\quad \mathrm{cp}(i)\neq\mathrm{cp}(j) \tag{2}$$

其中,针对包含N个控制通道的多路复用器,$\mathrm{cp}(i)$的值设为$k(0\leqslant k<2^{\lceil\log_2 N\rceil})$,表示编号为$k$的控制模式被分配给第$i$个多路复用组合,$M$表示多路复用组合的最大编号. 以图1中的多路复用矩阵为例,与流阀门(f_1~f_5)相连的控制通道有5个,因此初始化6个控制端口($x_1,\bar{x}_1,x_2,\bar{x}_2,x_3,\bar{x}_3$),这些控制端口能相应生成的控制模式及模式编号如下:

$$0:\ \overline{x}_1\overline{x}_2\overline{x}_3\,1:\overline{x}_1\overline{x}_2x_3\,2:\overline{x}_1x_2\overline{x}_3\,3:\overline{x}_1x_2x_3$$

$$4:\ x_1\overline{x}_2\overline{x}_3\,5:x_1\overline{x}_2x_3\,6:x_1x_2\overline{x}_3\,7:x_1x_2\overline{x}_3$$

若用粒子p_i表示图1中的控制模式分配方案A,则该粒子的编码为p_i:0,4,3,6.

3.2.2 粒子评估方式

不同的控制模式分配方案会导致后续布线环节需要不同的控制阀门、控制路径数量,为了在种群搜索期间评估粒子p_i的质量,粒子p_i相应的评价函数如下:

$$F_n(S_i)=\alpha\cdot C_v+\beta\cdot E_l \tag{3}$$

其中,α和β是两个权重因子,C_v是控制阀门的数量,E_l是控制路径的数量. 由于本文的目标是尽可能地减少多路复用器中所需的资源数,因此适应度值越小的粒子所表示的控制模式分配方案越好.

3.2.3 粒子更新方式

在SLDPSO阶段,粒子的更新公式如下:

$$P_i^t=\mathrm{SF}_3\left(\mathrm{SF}_2\left(\mathrm{SF}_1\left(p_i^{t-1},\omega\right),c_1\right),c_2\right) \tag{4}$$

其中,ω是惯性权重,代表粒子进行变异操作的概率,c_1和c_2是加速度因子,代表粒子进行交叉操作的概率. SF_1为惯性分量,用于在一定程度上保持粒子原有状态. SF_2为个体认知分量,使粒子向其历史最优状态学习. SF_3为社会认知分量,使粒子向其样例池中某个优秀粒子进行学习.

(1)惯性分量

算法采用SF_1表示粒子的惯性分量,通过变异操作实现,表示为

$$W_i^t=\mathrm{SF}_1\left(P_i^{t-1},\omega\right)=\begin{cases}\mathrm{swap}\left(x_1,x_2\right), & r_1<\omega\\ \mathrm{rep}\left(x_1\right), & r_1\geqslant\omega\wedge C_u\neq\varnothing\\ P_i^{t-1}, & \text{否则}\end{cases} \tag{5}$$

其中,W_i^t是速度更新后的粒子,r_1是[0, 1)之间的随机数,C_u是没有被分配过的控制模式编号集合. 针对粒子p_i,swap(x_1,x_2)是通过交换x_1和x_2所表示的两个控制模式来更新粒子;rep(x_1)则是用C_u中任意一个控制模式来替换x_1所表示的控制模式.

图6展示了粒子p_i在迭代过程中进行速度更新的两种方式. 例如,图6(a)采用rep(·)方式,从集合C_u中取出编号为7的控制模式来替换第一个多路复用组合(01100)原先已分配的控制模式,将粒子更新为W_i^t:7,4,3,6. 图6(b)则采用swap(·, ·)方式,将第一个和第三个多路复用组合原先已分配的控制模式进行交换,将

粒子更新为 W_i^t:0,6,3,4.

(2)个体认知分量

算法使用 SF_2 表示粒子的个体认知分量,通过交叉操作实现,表示为

$$I_i^t=\mathrm{SF}_2\left(W_i^t,c_1\right)=\begin{cases}C_p\left(W_i^t,p_i^{\mathrm{pbest}}\right), & r_2<c_1\\ W_i^t, & \text{否则}\end{cases}\tag{6}$$

其中,I_i^t 是交叉后的粒子;r_2 是[0, 1)之间的随机数;c_1 是学习因子,决定粒子 W_i^t 与其历史最优位置 p_i^{pbest} 进行交叉操作的概率. 如图7所示,在进行交叉操作时,首先逐个维度地扫描粒子 W_i^t 及其学习.

$$\begin{pmatrix}0&1&1&0&0\\1&0&0&1&0\\0&1&0&0&0\\0&1&0&1&0\end{pmatrix}\begin{matrix}\bar{x}_1\bar{x}_2\bar{x}_3\\x_1\bar{x}_2\bar{x}_3\\\bar{x}_1x_2x_3\\x_1x_2\bar{x}_3\end{matrix}\qquad \begin{matrix}f_1&f_2&f_3&f_4&f_5\end{matrix}$$

$$\begin{pmatrix}0&1&1&0&0\\1&0&0&1&0\\0&1&0&0&0\\0&1&0&1&0\end{pmatrix}\begin{matrix}\bar{x}_1\bar{x}_2\bar{x}_3\\x_1\bar{x}_2\bar{x}_3\\\bar{x}_1x_2x_3\\x_1x_2\bar{x}_3\end{matrix}\qquad \begin{matrix}f_1&f_2&f_3&f_4&f_5\end{matrix}$$

C_u: 1, 2, 5, 7 p_i^{t-1}: 0, 4, 3, 6　　C_u: 1, 2, 5, 7 p_i^{t-1}: 0, 4, 3, 6

⇓rep(0)　　⇓swap(1, 3)

$$\begin{pmatrix}0&1&1&0&0\\1&0&0&1&0\\0&1&0&0&0\\0&1&0&1&0\end{pmatrix}\begin{matrix}x_1x_2x_3\\x_1\bar{x}_2\bar{x}_3\\\bar{x}_1x_2x_3\\x_1x_2\bar{x}_3\end{matrix}\qquad \begin{matrix}f_1&f_2&f_3&f_4&f_5\end{matrix}$$

$$\begin{pmatrix}0&1&1&0&0\\1&0&0&1&0\\0&1&0&0&0\\0&1&0&1&0\end{pmatrix}\begin{matrix}\bar{x}_1\bar{x}_2\bar{x}_3\\x_1\bar{x}_2\bar{x}_3\\\bar{x}_1x_2x_3\\x_1\bar{x}_2\bar{x}_3\end{matrix}\qquad \begin{matrix}f_1&f_2&f_3&f_4&f_5\end{matrix}$$

C_u: 1, 2, 5 W_i^t: 7, 4, 3, 6　　C_u: 1, 2, 5, 7 W_i^t: 0, 6, 3, 4

(a) 变异操作　　(b) 交叉操作

图6　SLDPSO阶段的惯性分量

目标 p_i^{pbest},若 W_i^t 和 p_i^{pbest} 在某个多路复用组合所分配到的控制模式相同,则在粒子的新位置中保持该多路复用组合所分配的控制模式不变(如图7中 $x_1x_2x_3$)若在分配过程中出现了冲突,即将同一个控制模式同时分配给了不同的多路复用组合(如图7中 $x_1\bar{x}_2\bar{x}_3$),则需要重新进行交叉操作,最终得到更新后的粒子 I_i^t:7, 4, 0, 6. 经过上述交叉操作,粒子在种群进化过程中能够保留住一部分较好的基因.

(3)社会认知分量

算法使用 SF_3 表示粒子的社会认知分量,通过交叉操作实现,表示为

$$G_i^t=\mathrm{SF}_3\left(I_i^t,c_2\right)=\begin{cases}C_p\left(I_i^t,p_k^{\mathrm{pbest}}\right), & r_3<c_2\\ I_i^t, & \text{否则}\end{cases}\tag{7}$$

其中,G_i^t 是交叉后的粒子,r_3 是[0, 1)之间的随机数,c_2 是学习因子. 当 $r_3<c_2$ 时,粒子 I_i^t 与其样例池中粒子 p_k^{pbest} 进行交叉操作. 社会认知分量与个体认知分量中交叉操作相同. 通过交叉操作,粒子能够不断地在种群中学习到更好的基因,在多次迭代后就能使得所有粒子向全局最优位置靠拢.

$$\begin{pmatrix}0&1&1&0&0\\1&0&0&1&0\\0&1&0&0&0\\0&1&0&1&0\end{pmatrix}\begin{matrix}x_1x_2x_3\\x_1\bar{x}_2\bar{x}_3\\\bar{x}_1x_2x_3\\x_1x_2\bar{x}_3\end{matrix}\qquad \begin{matrix}f_1&f_2&f_3&f_4&f_5\end{matrix}$$

$$\begin{pmatrix}0&1&1&0&0\\1&0&0&1&0\\0&1&0&0&0\\0&1&0&1&0\end{pmatrix}\begin{matrix}x_1x_2x_3\\x_1\bar{x}_2\bar{x}_3\\\bar{x}_1x_2x_3\\x_1\bar{x}_2\bar{x}_3\end{matrix}\qquad \begin{matrix}f_1&f_2&f_3&f_4&f_5\end{matrix}$$

W_i^t: 7, 4, 3, 6　　p_i^{pbest}: 7, 5, 4, 0

⇓

$$\begin{pmatrix}0&1&1&0&0\\1&0&0&1&0\\0&1&0&0&0\\0&1&0&1&0\end{pmatrix}\begin{matrix}x_1x_2x_3\\x_1\bar{x}_2\bar{x}_3\\\bar{x}_1\bar{x}_2\bar{x}_3\\x_1x_2\bar{x}_3\end{matrix}\qquad \begin{matrix}f_1&f_2&f_3&f_4&f_5\end{matrix}$$

I_i^t: 7, 4, 4, 6

⇓解决冲突

$$\begin{pmatrix}0&1&1&0&0\\1&0&0&1&0\\0&1&0&0&0\\0&1&0&1&0\end{pmatrix}\begin{matrix}x_1x_2x_3\\x_1\bar{x}_2\bar{x}_3\\\bar{x}_1\bar{x}_2\bar{x}_3\\x_1x_2\bar{x}_3\end{matrix}\qquad \begin{matrix}f_1&f_2&f_3&f_4&f_5\end{matrix}$$

I_i^t: 7, 4, 0, 6

图7　SLDPSO阶段的个体认知分量

3.3　ARS阶段

3.3.1　区域编码策略

在DPSO算法中,粒子 p_i 当前所处的位置只能表示一个解,如图8(a)所示. 相反,如图8(b)所示,针对区域编码策略而言,其基本思想是使用一个粒子 p_i 来表示以粒子 p_i 为圆心、r_i 为半径的圆形区域内的所有位置,从而使粒子能覆盖更多的解空间.

在该编码方式下,粒子 p_i 在其覆盖的区域内进行局部搜索,快速地在该区域内找到一个更好的位置 p_i^{new},并移动到 p_i^{new},从而使得种群能以较快地速度向全局最优解靠近.

3.3.2　自适应区域搜索

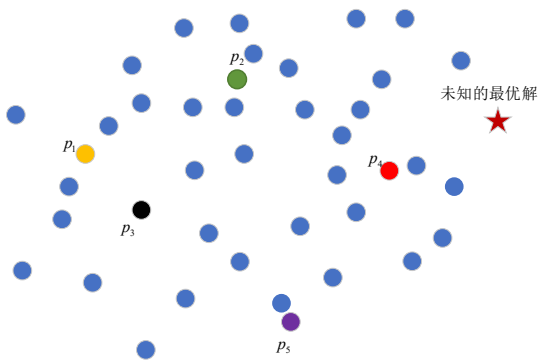
自适应区域搜索,即粒子 p_i 在其覆盖区域内进行局部搜索时,该区域的半径 r_i 会根据搜索的结果进行自适应的调整. 以图8(c)为例,当粒子 p_4 在进行某次区域搜索时找到了其覆盖范围内更好的位置 p_4^{new},此是本文将 p_4 更新为 p_4^{new}. 针对这种情况,说明在粒子 p_4 的覆盖区域内有较大希望找到更优解,那么通过扩大搜索半径则能够引导种群快速向全局最优解移动. 因此,粒子 p_4 在执行完 T 次区域搜索后将原来的搜索半径 r_4 扩大为 r_4^{new}. 相反,粒子 p_1 在 T 次区域搜索过程中没有找到更好的位置,说明粒子 p_1 可能已经找到了近似的局部最优解或全局最优解,太大的搜索半径可能导致在搜索阶段跳过该区域的最优解,此时缩小 p_1 的搜索半径

为 r_1^{new}. 此外，每次迭代过程中，只有种群中最优秀的 P 个粒子会执行区域搜索.

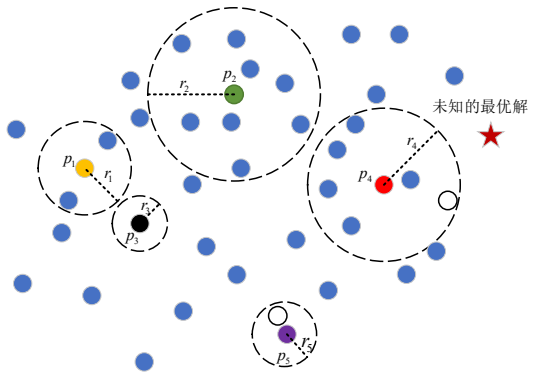
根据上述搜索策略，在 SLDPSO-ARS-CPA 算法的 ARS 阶段，粒子的位置更新公式为

$$p_i^{\text{new}}=\begin{cases}\left\{\text{swap}\left(x_1,x_2\right)\right\}_{\left\lceil\frac{r_i}{2}\right\rceil}, & \text{rand}\left(0,1\right)<\rho\\ \left\{\text{rep}\left(x_1\right)\right\}r_i, & \text{否则}\end{cases}\tag{8}$$

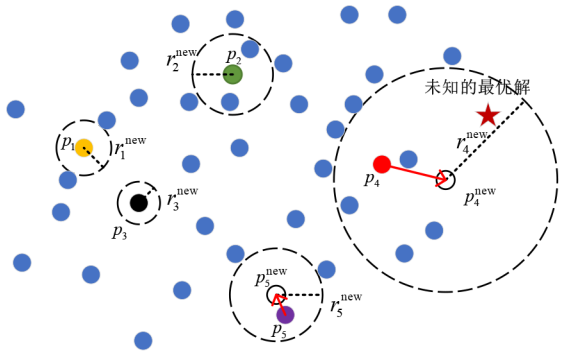
其中，$\text{swap}(x_1,x_2)$ 是通过交换 x_1 和 x_2 所表示的两个控制模式来更新粒子，$\{\text{swap}(x_1,x_2)\}_{\lceil r_i/2\rceil}$ 表示执行 $\text{swap}(x_1,x_2)$ 操作 $\lceil r_i/2\rceil$ 次. $\text{rep}(x_1)$ 表示从未被分配的控制模式编号集合中任意选择一个控制模式替换粒子 p_i 中 x_1 所表示的控制模式，$\{\text{rep}(x_1)\}_{r_i}$ 代表执行 $\text{rep}(x_1)$ 操作 r_i 次，如果粒子新位置 p_i^{new} 的适应度值小于 p_i 当前的适应度值，说明 p_i^{new} 所表示的控制模式分配方案所需的资源数量更少，则用 p_i^{new} 代替原来的粒子 p_i.



(a) 基于点的编码方式



(b) 基于区域的编码方式



(c) ARS 策略

图 8 编码策略

在 ARS 阶段，粒子经过 T 次区域搜索后，半径的自适应更新公式为

$$r_i^{\text{new}}=\begin{cases}r_i-1, & \text{flag}=0\\ r_i+1, & \text{flag}=1\end{cases}\tag{9}$$

其中，flag 为 0 表示粒子 p_i 在 T 次区域搜索过程中没有找到更好的位置，否则 flag 标记为 1. 此外，粒子 p_i 覆盖范围的半径为 r_i，$r_i\in[0,r_{\max}]$. 如果 r_i^{new} 的值超过了其覆盖范围区间，则把 r_i^{new} 设为 0 或 $r_{\max}$. 在该问题中，$r_{\max}$ 以及初始半径 r_0 的计算方式如式(10)和式(11)所示：

$$r_0=\left\lceil\frac{D}{2}\right\rceil\tag{10}$$

$$r_{\max}=\left\lceil\frac{D}{2}\right\rceil\tag{11}$$

其中，D 是粒子 p_i 的维度(即多路复用组合的数量).

4 实验结果

本文实验采用文献[14]使用的 5 个测试用例 RA30、CPA、R0、R1、R2 对所提算法进行测试. 本文展示了该算法在 2 个现实生化应用上的执行结果，其中有在 RA30 芯片上执行的比色蛋白质测定以及在 CPA 芯片上执行的体外诊断[14]. 此外，随机生成 3 个规模更大的测试用例 case1、case2、case3 来进一步验证算法的有效性. 上述测试用例的详细信息如表 1 所示，其中，A_c 是控制逻辑中使用到的控制模式数量，N_c 是控制通道的数量. 本文算法的参数设置如下：种群大小 $N=50$，惯性分量中的惯性权重 $\omega=0.5$，个体认知分量中学习因子 $c_1=1$，社会认知分量中学习因子 $c_2=1$，粒子评价函数中权重因子 α 和 β 的取值分别为 0.7 和 0.3，ARS 阶段决定粒子更新方式的决策因子 $\rho=0.5$，执行区域搜索的粒子数 $P=5$，粒子每次迭代过程中进行区域的次数 $T=15$，算法迭代过程中评价函数的最大调用次数 maxFEs = 30 000. 考虑到算法的随机性，本文的实验结果取 20 次独立运行的平均值. 本文在 4.1 节分析了 ARS 策略中相关参数对算法的影响. 在 4.2~4.4 节分析了 SLDPSO-ARS-CPA 算法中 SLDPSO 和 ARS 策略分别对算法的影响，验证了这两个阶段在控制模式分配问题中的有效性，并与群智能算法中具有代表性的麻雀搜索算法(Sparrow Search Algorithm，SSA)[31]作对比，进

表1　测试用例详细信息

测试用例	RA30	CPA	R0	R1	R2	case1	case2	case3
A_c	17	22	26	28	51	45	100	89
N_c	19	25	22	27	48	47	90	88

一步体现了本文算法的优越性.

4.1　对参数P、T、$r_{\max}$和r_0的设置

ARS是本文所提SLDPSO-ARS-CPA算法中的一个重要阶段,其有效性将直接影响SLDPSO-ARS-CPA算法解的质量. 在ARS阶段,执行区域搜索的粒子个数P和执行区域搜索的次数T在一定程度上决定了算法的收敛速度和解的质量. 若P和T的值过小会使较少的粒子执行局部搜索,且会使最好的P个粒子在进行局部搜索时搜索不充分,导致在ARS阶段错过该区域的局部最优解. 若P和T的值过大,会导致在ARS阶段浪费较多的FEs,使SLDPSO阶段执行次数减少. 为了选取适当的P值和T值,本文将P和T的可能取值设为5、10、15.

在ARS策略中,每个粒子所覆盖的区域都有一个半径r,半径的大小会随着粒子区域搜索的结果进行自适应变化. 在SLDPSO-ARS-CPA算法中,半径采用线性递增或递减的方式进行变化. 如果搜索半径的初值r_0设置过大,则在有限的迭代次数中,随着半径r线性变化,会使粒子所覆盖的搜索区域始终较大. 粒子在较大的搜索区域中进行T次区域搜索难以搜索到该区域高质量的解方案,使区域搜索的效果极大地降低. 若搜索半径的初值r_0设置过小,则会导致在区域搜索过程中难以跳出局部最优解,同样无法发挥区域搜索的效果. 在粒子半径r线性增大过程中,本文设置$r_{\max}$来防止r无限增大,若$r_{\max}$的值过大,会导致r增大到一定程度时使粒子所覆盖的搜索区域过大,产生和r_0设置过大时相同的弊端. 而$r_{\max}$的值过小会使r的值限制在很小的范围内,导致难以跳出局部最优解. 为了选取适当的$r_{\max}$和r_0,在满足约束$r_0 \leqslant r_{\max}$的同时,本文将$r_{\max}$和r_0可能的取值设置为$\lceil D/4\rceil$、$\lceil D/2\rceil$、$\lceil 3D/4\rceil$.

为了探究P、T、$r_{\max}$以及r_0对ARS策略的影响,本节对P、T、$r_{\max}$以及r_0的54种组合进行了实验,最终选出该控制模式分配问题下的最佳参数组合,并且参数的组合情况使用SLDPSO-ARS-CPA(w,x,y,z)来表示. 如SLDPSO-ARS-CPA$(5,10,\lceil D/4\rceil,\lceil D/2\rceil)$表示在SLDPSO-ARS-CPA算法每轮迭代的ARS阶段,较好的5个粒子依次执行10次区域搜索,在区域搜索的过程中该粒子初始半径r_0的取值为$\lceil D/4\rceil$,且其半径取值区间为$[0,\lceil D/2\rceil]$. 实验结果如表2所示,表中呈现了8个测试用例在算法的54种参数组合下求得的控制阀门数量以及各组参数中控制阀门数量的最小值. 该实验中采用指标index来判断参数组合的质量,index的计算方式为

$$\text{index}=\sqrt{\frac{\sum_{i\in T}\left(C_{vi}-C'_{vi}\right)^2}{8}} \tag{12}$$

其中,T表示8个测试用例的集合,C_{vi}表示测试用例i在当前参数组合下的阀门数量,C'_{vi}表示测试用例i在所有参数组合中的最小阀门数量. 综合考虑各个测试用例上的表现结果,index的值越小,说明基于该参数组合,算法求得的解方案有机会进一步接近最优值. 实验结果验证了参数P、T、$r_{\max}$以及r_0取值的不同组合对ARS阶段的重要影响,结果表明,在参数组合SLDPSO-ARS-CPA$(5,15,\lceil D/2\rceil,\lceil D/2\rceil)$下的index取值最小,为8.27. 即在ARS阶段,当$T=15$,$p=5$,$r_0=\lceil D/2\rceil$,$r\in[0,\lceil D/2\rceil]$时算法所求控制模式分配方案所需阀门数最少.

4.2　SLDPSO策略的有效性验证

为了验证SLDPSO策略的有效性,本节将使用该策略的SLDPSO算法与文献[14]中改进的DPSO算法进行对比,使用最佳阀门数、最差阀门数、平均阀门数以及标准差这4个指标来衡量策略的优化程度,结果如表3所示. 在8个测试用例中,本文算法虽然在最佳阀门数量、最差阀门数量以及平均阀门数量无明显优化,但使标准差平均优化了32.21%. 其中,在case1上算法的标准差优化了64.29%. 值得注意的是,SLDPSO相对于文献[14]中的DPSO方法,标准差得到了较大的优化,说明算法稳定性得到了提高. 在DPSO方法中,粒子总是通过向种群最优粒子学习的方式实现社会学习,当较差的粒子向最优粒子学习时移动过快,易跳过全局最优解,导致算法波定性较大. 而SLDPSO采用基于样例的社会学习机制,每个粒子维持一个用于记录比其更优粒子的样例池. 当较差的粒子进行社会学习时,该粒子向其样例池中任意一个粒子进行学习,从而能减小学习过程中的波动性. 结果表明,在引入基于样例的社会学习机制后,能大幅度提高算法的稳定性,证明了SLDPSO策略的有效性.

表 2 对参数 P、T、r_{max} 和 r_0 的设置

参数组合				RA30	CPA	R0	R1	R2	case1	case2	case3	index
P	T	r_0	r_{max}									
5	5	$\lceil D/4 \rceil$	$\lceil D/4 \rceil$	62	73	117	101	237	299	762	674	15.33
			$\lceil D/2 \rceil$	61	71	115	99	238	304	769	670	16.10
			$\lceil 3D/4 \rceil$	60	74	114	100	231	299	770	670	14.22
		$\lceil D/2 \rceil$	$\lceil D/2 \rceil$	58	75	113	101	240	306	768	667	16.31
			$\lceil 3D/4 \rceil$	60	72	113	104	233	306	760	664	13.56
		$\lceil 3D/4 \rceil$	$\lceil 3D/4 \rceil$	60	74	114	102	242	298	770	669	16.48
	10	$\lceil D/4 \rceil$	$\lceil D/4 \rceil$	64	72	113	100	222	288	770	651	9.13
			$\lceil D/2 \rceil$	59	72	112	99	222	289	756	676	11.43
			$\lceil 3D/4 \rceil$	60	73	113	98	226	295	758	660	8.74
		$\lceil D/2 \rceil$	$\lceil D/2 \rceil$	59	74	121	98	233	299	758	654	10.94
			$\lceil 3D/4 \rceil$	60	74	113	97	220	292	757	665	8.45
		$\lceil 3D/4 \rceil$	$\lceil 3D/4 \rceil$	61	71	118	94	229	301	775	669	14.88
	15	$\lceil D/4 \rceil$	$\lceil D/4 \rceil$	61	73	113	100	221	297	764	665	10.21
			$\lceil D/2 \rceil$	59	72	114	99	219	291	762	676	11.84
			$\lceil 3D/4 \rceil$	61	72	114	101	224	285	765	659	8.91
		$\lceil D/2 \rceil$	$\lceil D/2 \rceil$	59	73	114	95	210	297	754	666	8.27
			$\lceil 3D/4 \rceil$	60	71	115	98	223	296	759	667	10.11
		$\lceil 3D/4 \rceil$	$\lceil 3D/4 \rceil$	61	72	116	99	228	290	769	662	11.01
10	5	$\lceil D/4 \rceil$	$\lceil D/4 \rceil$	60	70	117	100	239	305	764	674	16.66
			$\lceil D/2 \rceil$	60	74	114	99	240	303	773	675	18.04
			$\lceil 3D/4 \rceil$	63	73	115	98	241	300	768	665	15.40
		$\lceil D/2 \rceil$	$\lceil D/2 \rceil$	63	72	117	97	248	311	760	665	18.09
			$\lceil 3D/4 \rceil$	61	72	119	101	241	307	754	673	16.83
		$\lceil 3D/4 \rceil$	$\lceil 3D/4 \rceil$	60	74	121	103	243	307	773	658	17.24
	10	$\lceil D/4 \rceil$	$\lceil D/4 \rceil$	58	71	118	98	229	300	765	663	11.81
			$\lceil D/2 \rceil$	63	73	116	101	233	301	779	667	16.36
			$\lceil 3D/4 \rceil$	64	74	115	97	235	296	766	662	12.73
		$\lceil D/2 \rceil$	$\lceil D/2 \rceil$	58	71	116	99	239	305	771	650	14.69
			$\lceil 3D/4 \rceil$	59	74	117	97	233	308	760	657	12.83
		$\lceil 3D/4 \rceil$	$\lceil 3D/4 \rceil$	63	71	118	101	246	306	779	665	19.40
	15	$\lceil D/4 \rceil$	$\lceil D/4 \rceil$	60	72	117	101	236	305	762	666	14.49
			$\lceil D/2 \rceil$	61	74	117	101	235	306	764	663	14.12
			$\lceil 3D/4 \rceil$	57	70	118	100	240	306	750	669	15.47
		$\lceil D/2 \rceil$	$\lceil D/2 \rceil$	59	74	115	98	237	306	755	665	13.95
			$\lceil 3D/4 \rceil$	61	72	119	98	232	298	773	669	14.87
		$\lceil 3D/4 \rceil$	$\lceil 3D/4 \rceil$	60	73	114	101	234	297	762	673	14.22
15	5	$\lceil D/4 \rceil$	$\lceil D/4 \rceil$	59	72	122	100	253	308	765	667	19.78
			$\lceil D/2 \rceil$	61	72	119	98	249	306	759	670	18.27
			$\lceil 3D/4 \rceil$	59	72	121	102	238	316	759	661	16.55
		$\lceil D/2 \rceil$	$\lceil D/2 \rceil$	60	73	118	100	242	313	768	671	18.79
			$\lceil 3D/4 \rceil$	59	73	116	100	257	308	774	665	21.45
		$\lceil 3D/4 \rceil$	$\lceil 3D/4 \rceil$	60	70	122	102	250	308	762	661	18.18

续表

参数组合				RA30	CPA	R0	R1	R2	case1	case2	case3	index
P	T	r_0	r_{max}									
15	10	$\lceil D/4 \rceil$	$\lceil D/4 \rceil$	58	72	119	100	251	306	757	670	18.68
			$\lceil D/2 \rceil$	59	74	118	101	239	305	773	668	17.08
			$\lceil 3D/4 \rceil$	60	71	113	97	242	295	762	662	13.80
		$\lceil D/2 \rceil$	$\lceil D/2 \rceil$	61	72	117	100	245	302	752	677	17.71
			$\lceil 3D/4 \rceil$	60	71	119	100	247	308	768	669	18.74
		$\lceil 3D/4 \rceil$	$\lceil 3D/4 \rceil$	59	72	117	104	259	305	753	647	19.17
	15	$\lceil D/4 \rceil$	$\lceil D/4 \rceil$	57	72	121	101	241	295	766	669	15.52
			$\lceil D/2 \rceil$	61	70	119	100	241	300	767	663	15.00
			$\lceil 3D/4 \rceil$	61	74	119	99	245	315	769	661	18.73
		$\lceil D/2 \rceil$	$\lceil D/2 \rceil$	59	73	116	104	252	308	754	676	20.17
			$\lceil 3D/4 \rceil$	62	73	121	103	251	310	758	662	18.64
		$\lceil 3D/4 \rceil$	$\lceil 3D/4 \rceil$	60	74	119	102	263	313	760	656	22.09
最小值				57	70	112	94	210	285	750	647	8.27

4.3　ARS策略的有效性验证

为了验证ARS策略的有效性,本节将使用该策略的DPSO-ARS算法与文献[14]中的DPSO算法进行对比,使用与4.2节相同的4个指标来衡量策略的优化程度,结果如表4所示. 可以发现,DPSO-ARS算法虽然在标准差上无优化,但在最佳阀门数量、最差阀门数量以及平均阀门数量上分别优化了11.51%、12.03%、12.80%. 在文献[14]中,粒子总是向种群最优粒子学习,因此所有粒子的移动方向一致,导致过早陷入局部最优解,且一旦陷入局部最优解就难以再跳出. 而在种群中,较好的粒子接近全局最优解的概率更大. 若这些较好的粒子在每轮迭代结束时在一定范围内进行区域搜索,则有机会移动到全局最优解处,且能有效跳出局部最优解. 同时,根据当前区域搜索的结果动态地调整下一次区域搜索的半径,能提高区域搜索的效率,从而提高ARS策略的性能. 结果表明,使用ARS策略能提高解的质量,求得一个所需阀门数量更少的控制模式分配方案,验证了该策略的有效性.

4.4　SLDPSO-ARS-CPA算法的有效性验证

通过4.2节和4.3节的分析可以看出,在单独使用SLDPSO策略时,算法的稳定性能得到较好的提高,但解的质量可能会有所下降. 相反,在单独使用ARS策略时,算法能得到较高质量的解,但算法稳定性可能会降低.

为了探究在SLDPSO策略和ARS策略共同作用下的效果,本节将SLDPSO-ARS-CPA算法与文献[14]中改进的DPSO算法进行对比. 如表5所示,可以发现本文算法在最佳阀门数量、最差阀门数量、平均阀门数量以及标准差分别优化了16.74%、20.35%、19.01%、29.18%. 其中,测试用例R2在上述4个指标分别优化了39.75%、35.68%、36.57%、15.38%. 值得关注的是,本文算法在这4个指标上的平均优化率比单独使用SLDPSO策略和单独使用ARS策略时都要高. 实验结果表明,在SLDPSO策略和ARS策略的共同作用下,本文算法能以较高的稳定性求得所需阀门数更少的控制模式分配方案.

为了进一步验证SLDPSO-ARS-CPA算法在控制模式分配问题中的优越性,将本文算法与群体智能算法中具有代表性的SSA算法[31]在8个测试用例上进行对比. 如表6所示,尽管本文算法在标准差上与SSA算法仍有一定的差距,但在最佳阀门数量、最差阀门数量以及平均阀门数量等重要的性能指标方面分别优化了23.71%、23.97%、23.86%. 实验结果表明,本文算法相较于SSA算法在阀门数量这一重要性能指标具有很大优势的优化效果,从而说明了本文算法的优越性.

此外,本节对比了本文算法SLDPSO-ARS-CPA与SSA算法[31]以及文献[14]算法在8个测试用例上的收敛速度和收敛精度,如图9所示. 在RA30和CPA上的收敛曲线表明,在算法前期的ARS阶段,粒子在局部搜索时没有找到其覆盖区域内的局部最优解,因此在算法前期没有起到加速收敛的效果. 而在其他6个规模更大的测试用例上,ARS一开始就起到了明显的加速收敛的效果. 同时,相较于文献[14]算法和SSA算法,本文算法在所有测试用例上得到的解方案精度都较高. 可以发现,随着测试用例规模的增大,本文算法的优势更加明显.

表3 SLDPSO算法与文献[14]算法的比较

用例	最佳阀门数量			最差阀门数量			平均阀门数量			标准差		
	文献[14]	SLDPSO	优化率	文献[14]	SLDPSO	优化率	文献[14]	SLDPSO	优化率	文献[14]	SLDPSO	优化率
RA30	58	59	−1.72%	75	76	−1.33%	67	65	2.99%	4	4	0.00%
CPA	70	70	0.00%	86	87	−1.16%	80	79	1.25%	4	4	0.00%
R0	121	112	7.44%	146	144	1.37%	131	131	0.00%	8	9	−12.50%
R1	100	116	−16.00%	126	129	−2.38%	113	123	−8.85%	7	3	57.14%
R2	322	316	1.86%	370	343	7.30%	350	332	5.14%	13	7	46.15%
case1	337	360	−6.82%	402	377	6.22%	359	370	−3.06%	14	5	64.29%
case2	992	999	−0.71%	1065	1036	2.72%	1038	1017	2.02%	21	11	47.62%
case3	842	860	−2.14%	927	891	3.88%	898	878	2.23%	20	9	55.00%
均值	—	—	−2.26%	—	—	2.08%	—	—	0.21%	—	—	32.21%

表4 DPSO-ARS算法与文献[14]算法的比较

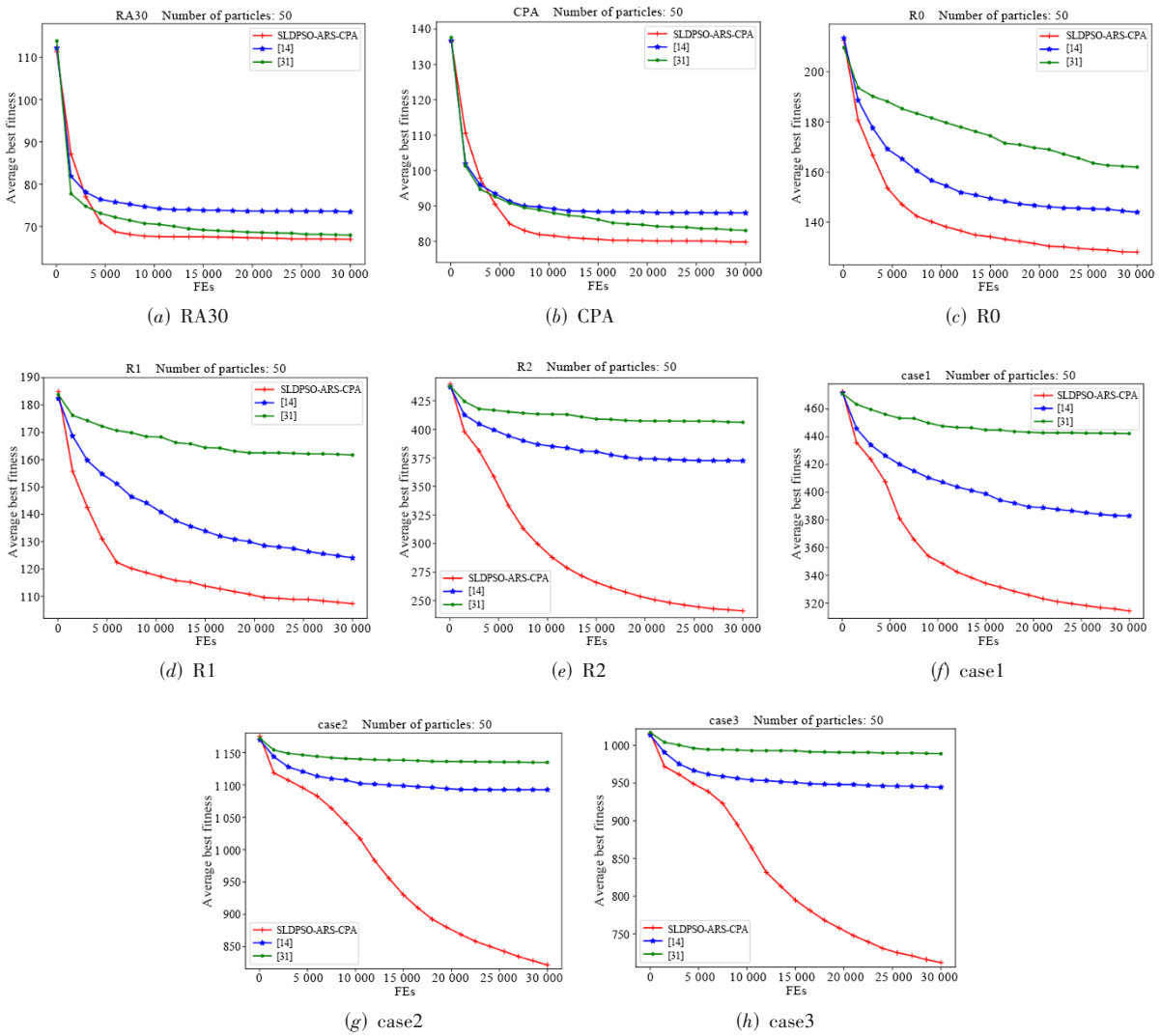
用例	最佳阀门数量			最差阀门数量			平均阀门数量			标准差		
	文献[14]	DPSO-ARS	优化率	文献[14]	DPSO-ARS	优化率	文献[14]	DPSO-ARS	优化率	文献[14]	DPSO-ARS	优化率
RA30	58	57	1.72%	75	66	12.00%	67	60	10.45%	4	3	25.00%
CPA	70	66	5.71%	86	80	6.98%	80	73	8.75%	4	4	0.00%
R0	121	107	11.57%	146	127	13.01%	131	114	12.98%	8	5	37.50%
R1	100	93	7.00%	126	112	11.11%	113	102	9.73%	7	6	14.29%
R2	322	233	27.64%	370	310	16.22%	350	270	22.86%	13	18	−38.46%
case1	337	293	13.06%	402	346	13.93%	359	314	12.53%	14	15	−7.14%
case2	992	864	12.90%	1065	952	10.61%	1038	915	11.85%	21	26	−23.81%
case3	842	737	12.47%	927	812	12.41%	898	779	13.25%	20	23	−15.00%
均值	—	—	11.51%	—	—	12.03%	—	—	12.80%	—	—	−0.95%

表5 SLDPSO-ARS-CPA算法与文献[14]算法的比较

用例	最佳阀门数量			最差阀门数量			平均阀门数量			标准差		
	文献[14]	SLDPSO-ARS-CPA	优化率	文献[14]	SLDPSO-ARS-CPA	优化率	文献[14]	SLDPSO-ARS-CPA	优化率	文献[14]	SLDPSO-ARS-CPA	优化率
RA30	58	56	3.45%	75	66	12.00%	67	60	10.45%	4	3	25.00%
CPA	70	68	2.86%	86	75	12.79%	80	72	10.00%	4	2	50.00%
R0	121	105	13.22%	146	125	14.38%	131	115	12.21%	8	6	25.00%
R1	100	90	10.00%	126	105	16.67%	113	97	14.16%	7	5	28.57%
R2	322	194	39.75%	370	238	35.68%	350	222	36.57%	13	11	15.38%
case1	337	278	17.51%	402	311	22.64%	359	293	18.38%	14	9	35.71%
case2	992	751	24.29%	1065	817	23.29%	1038	776	25.24%	21	16	23.81%
case3	842	650	22.80%	927	692	25.35%	898	673	25.06%	20	14	30.00%
均值	—	—	16.74%	—	—	20.35%	—	—	19.01%	—	—	29.18%

表6　SLDPSO-ARS-CPA算法与文献[31]的比较

用例	最佳阀门数量			最差阀门数量			平均阀门数量			标准差		
	文献[31]	SLDPSO-ARS-CPA	优化率	文献[31]	SLDPSO-ARS-CPA	优化率	文献[31]	SLDPSO-ARS-CPA	优化率	文献[31]	SLDPSO-ARS-CPA	优化率
RA30	56	56	0.00%	73	66	9.59%	61	60	1.64%	4	3	25.00%
CPA	68	68	0.00%	80	75	6.25%	75	72	4.00%	3	2	33.33%
R0	131	105	19.85%	162	125	22.84%	148	115	22.30%	7	6	14.29%
R1	130	90	30.77%	158	105	33.54%	149	97	34.90%	7	5	28.57%
R2	373	194	47.99%	397	238	40.05%	382	222	41.88%	9	11	–22.22%
case1	404	278	31.19%	426	311	27.00%	416	293	29.57%	8	9	–12.50%
case2	1071	751	29.88%	1092	817	25.18%	1079	776	28.08%	6	16	–166.67%
case3	929	650	30.03%	952	692	27.31%	941	673	28.48%	6	14	–133.33%
均值	—	—	23.71%	—	—	23.97%	—	—	23.86%	—	—	–29.19%



(*a*) RA30　(*b*) CPA　(*c*) R0

(*d*) R1　(*e*) R2　(*f*) case1

(*g*) case2　(*h*) case3

图9　SLDPSO-ARS-CPA算法与文献[14]算法和文献[31]算法的收敛曲线对比

5 结语

针对连续微流控生物芯片下的控制模式分配问题,以优化多路复用器中的控制阀门数量为目标,提出了SLDPSO-ARS-CPA算法. 首先,为了避免过早陷入局部最优解,同时加快算法的收敛速度,提出了一种在离散问题下的自适应区域搜索策略. 其次,结合基于样例的社会学习机制,使种群在进化时降低粒子学习过程中的波动性,提高了算法稳定性. 最后,提出了等距抽值的方式对自适应区域搜索阶段重要参数的组合进行筛选,以进一步提高算法的收敛精度. 与同类算法对比,本文所提算法在保证稳定性的同时能以较快的速度收敛,并且能获得阀门数最少的控制模式分配方案. 经实验证明,本文算法在体外诊断以及比色蛋白测定生化应用中表现出良好的性能.

参考文献

[1] BALAGADDÉ F K, YOU L, HANSEN C L, et al. Long-term monitoring of bacteria undergoing programmed population control in a microchemostat[J]. Science, 2005, 309(5731): 137-140.

[2] WHITESIDES G M. The origins and the future of microfluidics[J]. Nature, 2006, 442(7101): 368-373.

[3] YAGER P, EDWARDS T, FU E, et al. Microfluidic diagnostic technologies for global public health[J]. Nature, 2006, 442(7101): 412-418.

[4] UNGER M A, CHOU H P, THORSEN T, et al. Monolithic microfabricated valves and pumps by multilayer soft lithography[J]. Science, 2000, 288(5463): 113-116.

[5] ROGERS J A, NUZZO R G. Recent progress in soft lithography[J]. Materials Today, 2005, 8(2): 50-56.

[6] YAO H, HO T Y, CAI Y. PACOR: Practical control-layer routing flow with length-matching constraint for flow-based microfluidic biochips[C]//ACM/EDAC/IEEE Design Automation Conference. Piscataway: IEEE, 2015: 1-6.

[7] HU K, DINH T A, HO T Y, et al. Control-layer routing and control-pin minimization for flow-based microfluidic biochips[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2016, 36(1): 55-68.

[8] HONG J W, QUAKE S R. Integrated nanoliter systems[J]. Nature Biotechnology, 2003, 21(10): 1179-1183.

[9] THORSEN T, MAERKL S J, QUAKE S R. Microfluidic large-scale integration[J]. Science, 2002, 298(5593): 580-584.

[10] GROVER W H, IVESTER R H C, JENSEN E C, et al. Development and multiplexed control of latching pneumatic valves using microfluidic logical structures[J]. Lab on a Chip, 2006, 6(5): 623-631.

[11] KIM S J, LAI D, PARK J Y, et al. Microfluidic automation using elastomeric valves and droplets: Reducing reliance on ex-ternal controllers[J]. Small, 2012, 8(19): 2925-2934.

[12] ZHONG J F, CHEN Y, MARCUS J S, et al. A microfluidic processor for gene expression profiling of single human embryonic stem cells[J]. Lab on a Chip, 2008, 8(1): 68-74.

[13] WANG Q, XU Y, ZUO S, et al. Pressure-aware control layer optimization for flow-based microfluidic biochips[J]. IEEE Transactions on Biomedical Circuits and Systems, 2017, 11(6): 1488-1499.

[14] ZHU Y, HUANG X, LI B, et al. Multicontrol: Advanced control-logic synthesis for flow-based microfluidic biochips[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2019, 39(10): 2489-2502.

[15] POLI R, KENNEDY J, BLACKWELL T. Particle swarm optimization[J]. Swarm Intelligence, 2007, 1(1): 33-57.

[16] GONG Y J, ZHANG J, CHUNG H S H, et al. An efficient resource allocation scheme using particle swarm optimization[J]. IEEE Transactions on Evolutionary Computation, 2012, 16(6): 801-816.

[17] LIU G, CHEN Z, ZHUANG Z, et al. A unified algorithm based on HTS and self-adapting PSO for the construction of octagonal and rectilinear SMT[J]. Soft Computing, 2020, 24(6): 3943-3961.

[18] SU B, LIN Y, WANG J, et al. Sewage treatment system for improving energy efficiency based on particle swarm optimization algorithm[J]. Energy Reports, 2022, 8: 8701-8708.

[19] SETAYESH M, ZHANG M, JOHNSTON M. A novel particle swarm optimisation approach to detecting continuous, thin and smooth edges in noisy images[J]. Information Sciences, 2013, 246: 28-51.

[20] CEYLAN Z. Short-term prediction of COVID-19 spread using grey rolling model optimized by particle swarm optimization[J]. Applied Soft Computing, 2021, 109: 107592.

[21] CHEN X, LIU G, XIONG N, et al. A survey of swarm intelligence techniques in VLSI routing problems[J]. IEEE Access, 2020, 8: 26266-26292.

[22] 朱予涵, 黄鸿斌, 林泓星, 等. 连续微流控生物芯片下基于序列对的流层物理设计算法[J]. 计算机辅助设计与

图形学学报, 2022, 34(4): 535-544.
ZHU Y H, HUANG H B, LIN H X, et al. Sequence-pair-based flow-layer physical design algorithm for continuous-flow microfluidic biochips[J]. Journal of Computer-Aided Design and Computer Graphics, 2022, 34(4): 535-544. (in Chinese)

[23] MOLINA D, HERRERA F. Iterative hybridization of DE with local search for the CEC'2015 special session on large scale global optimization[C]//IEEE Congress on Evolutionary Computation. Piscataway: IEEE, 2015: 1974-1978.

[24] LIU G, ZHOU R, XU S, et al. Two-stage competitive particle swarm optimization based timing-driven X-routing for IC design under smart manufacturing[J]. ACM Transactions on Management Information Systems, 2022, 13(4): 1-26.

[25] CHENG R, JIN Y. A social learning particle swarm optimization algorithm for scalable optimization[J]. Information Sciences, 2015, 291: 43-60.

[26] 王毅, 李晓梦, 耿国华, 等. 基于直觉模糊熵的混合粒子群优化算法[J]. 电子学报, 2021, 49(12): 2381-2389.
WANG Y, LI X M, GENG G H, et al. Hybrid particle swarm optimization algorithm based on intuitionistic fuzzy entropy[J]. Acta Electronica Sinica, 2021, 49(12): 2381-2389. (in Chinese)

[27] 刘耿耿, 黄逸飞, 王鑫, 等. 基于混合离散粒子群优化的Slew约束下X结构Steiner最小树算法[J]. 计算机学报, 2021, 44(12): 2542-2559.
LIU G G, HUANG Y F, WANG X, et al. Hybrid discrete particle swarm optimization algorithm for X-architecture Steiner minimal tree construction with Slew constraints[J]. Chinese Journal of Computers, 2021, 44(12): 2542-2559. (in Chinese)

[28] YU Z, SI Z, LI X, et al. A novel hybrid particle swarm optimization algorithm for path planning of UAVs[J]. IEEE Internet of Things Journal, 2022, 9(22): 22547-22558.

[29] ZHAO S Z, LIANG J J, SUGANTHAN P N, et al. Dynamic multi-swarm particle swarm optimizer with local search for large scale global optimization[C]//IEEE Congress on Evolutionary Computation. Piscataway: IEEE, 2008: 3845-3852.

[30] JIAN J R, CHEN Z G, ZHAN Z H, et al. Region encoding helps evolutionary computation evolve faster: A new solution encoding scheme in particle swarm for large-scale optimization[J]. IEEE Transactions on Evolutionary Computation, 2021, 25(4): 779-793.

[31] XUE J, SHEN B, PAN A. An intensified sparrow search algorithm for solving optimization problems[J]. Journal of Ambient Intelligence and Humanized Computing, 2023, 14(7): 9173-9189.

作者简介

曾裕钦　男,2000年出生,四川宜宾人. 硕士研究生. 主要研究方向为EDA算法.
E-mail: 1244934835@qq.com

蔡华洋　男,1997年出生,福建泉州人. 博士研究生. 主要研究方向为EDA算法.
E-mail: c_huayang@163.com

周茹平　女,1998年出生,福建福州人. 硕士研究生. 主要研究方向为EDA算法.
E-mail: zrp08200331@163.com

刘耿耿　男,1988年出生,福建南安人. 博士、副教授. 主要研究方向为EDA算法.
E-mail: liugenggeng@fzu.edu.cn

黄　兴　男,1991年出生,陕西咸阳人. 博士,教授. 主要研究方向为EDA算法.
E-mail: xing.huang@nwpu.edu.cn

徐　宁　男,1968年出生,湖北武汉人. 博士、教授. 主要研究方向为FPGA物理设计、电子设计自动化、大数据分析与人工智能、图像处理. 中国电子学会会员编号:E190002967S.
E-mail: xuning@whut.edu.cn