计 算 机 研 究 与 发 展
Journal of Computer Research and Development
https://crad.ict.ac.cn
62(4): 950−962, 2025

本期推荐

基于深度强化学习的连续微流控生物芯片控制逻辑布线

蔡华洋[1,2,3] 黄 兴[4] 刘耿耿[1,2,3]

[1](福州大学计算机与大数据学院 福州 350116)

[2](大数据智能教育部工程研究中心(福州大学) 福州 350116)

[3](福建省网络计算与智能信息处理重点实验室(福州大学) 福州 350116)

[4](西北工业大学计算机学院 西安 710072)

(c_huayang@163.com)

Control Logic Routing for Continuous-Flow Microfluidic Biochips Based on Deep Reinforcement Learning

Cai Huayang[1,2,3], Huang Xing[4], and Liu Genggeng[1,2,3]

[1](*College of Computer and Data Science*, *Fuzhou University*, *Fuzhou* 350116)

[2](*Engineering Research Center of Big Data Intelligence* (*Fuzhou University*), *Ministry of Education*, *Fuzhou* 350116)

[3](*Fujian Key Laboratory of Network Computing and Intelligent Information Processing* (*Fuzhou University*), *Fuzhou* 350116)

[4](*School of Computer Science*, *Northwestern Polytechnical University*, *Xi'an* 710072)

Abstract With the advancement of electronic design automation, continuous-flow microfluidic biochips have become one of the most promising platforms for biochemical experiments. This chip manipulates fluid samples in milliliters or nanoliters by utilizing internal microvalves and microchannels, and thus automatically performs basic biochemical experiments, such as mixing and detection. To achieve the correct bioassay function, the microvalves deployed inside the chip are usually managed by a multiplexer-based control logic, and valves receive control signals from a core input through the control channel for accurate switching. Since biochemical reactions typically require high sensitivity, the length of control paths connecting each valve needs to be reduced to ensure immediate signal propagation, and thus to reduce the signal propagation delay. In addition, to reduce the fabrication cost of chips, a vital issue to be addressed in the logic architecture design is how to effectively reduce the total channel length within the control logic. To address the above issues, we propose a deep reinforcement learning-based control logic routing algorithm to minimize the signal propagation delay and total control channel length, thereby automatically constructing an efficient control channel network. The algorithm employs the dueling deep Q-network architecture as the agent of the deep reinforcement learning framework to evaluate the tradeoff between signal propagation delay and total channel length. Besides, the diagonal channel routing is implemented for the first time for control logic, thus fundamentally improving the efficiency of valve switching operations and reducing the fabrication cost of the chip. The experimental results demonstrate that the proposed algorithm can effectively construct a high-performance and low-cost control logic architecture.

Key words continuous-flow microfluidic biochips; deep reinforcement learning; control logic; control channel network; diagonal channel routing

收稿日期: 2024-01-18; **修回日期**: 2024-08-12

基金项目: 国家自然科学基金项目(62372109); 福建省杰出青年科学基金项目(2023J06017)

This work was supported by the National Natural Science Foundation of China (62372109) and the Fujian Science Foundation for Distinguished Young Scholars (2023J06017).

通信作者: 刘耿耿(liugenggeng@fzu.edu.cn)

摘　要　随着电子设计自动化技术的迅速发展，连续微流控生物芯片成为了目前最具前景的生化实验平台之一. 该芯片通过采用内部的微阀门以及微通道来操纵体积仅为毫升或纳升的流体样品，从而自动执行混合和检测等基本的生化实验操作. 为了实现正确的生化测定功能，部署于芯片内部的微阀门通常需要由基于多路复用器的控制逻辑进行管控，其通过控制通道获得来自核心输入的控制信号以实现精确切换. 由于生化反应通常需要非常高的灵敏度，因此为了保证信号的即时传输，需要尽可能地减少连接每个阀门的控制路径长度，以降低信号传输的时延. 此外，为了降低芯片的制造成本，如何有效减少控制逻辑中通道的总长度也是逻辑架构设计需要解决的关键问题之一. 针对上述问题，提出了一种基于深度强化学习的控制逻辑布线算法以最小化信号传输时延以及控制通道总长度，从而自动构建高效的控制通道网络. 该算法采用竞争深度 Q 网络架构作为深度强化学习框架的智能体，从而对信号传输时延和通道总长度进行权衡评估. 此外，针对控制逻辑首次实现了对角型的通道布线，从根本上提高了阀门切换操作的效率并降低了芯片的制造成本. 实验结果表明，所提出的算法能够有效构建高性能、低成本的控制逻辑架构.

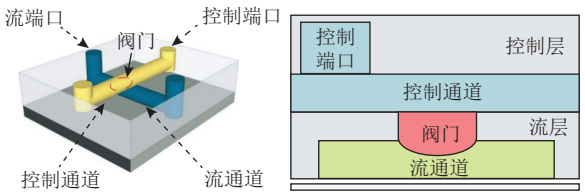
关键词　连续微流控生物芯片；深度强化学习；控制逻辑；控制通道网络；对角通道布线

中图法分类号　TP391

DOI: 10.7544/issn1000-1239.202440034　　**CSTR:** 32373.14.issn1000-1239.202440034

连续微流控生物芯片（continuous-flow microfluidic biochips, CFMBs），也称为片上实验设备，由于其具有高效率、高精度、低成本等优点，在过去的 20 年里已经成为了一个越来越有吸引力的生物化学实验平台[1-2]. 微流控技术通过将流体样品制备、反应、分离、检测等生物化学和医学分析过程的基本操作单元集成到微米尺度的芯片上，从而自动执行生化测定. 与需要人为干预的传统生化实验流程相比，避免了由于手动干预而产生的实验误差，显著地提升了实验结果的准确率和成功率. 如今，这种片上实验设备已经应用于生物化学与生物医学的许多领域[3]，如药物研发[4]和 DNA 分析[5]等.

CFMBs 的基本结构如图 1 所示，其由 2 层逻辑层构成，分别为流层和控制层，其中这 2 个层都由弹性体材料，即聚二甲基硅氧烷（polydimethylsiloxane, PDMS）[6]采用多层软光刻（multilayer soft lithography, MSL）技术制造而成. 每一个逻辑层都包含了一组属于该层的微通道，分别为控制层中的控制通道以及流层中的流通道，其中控制通道构建于流通道的顶部. 芯片上的设备，如混合器和检测器，通过流通道进行连接以实现设备间流体样品的输送. 如图 1（a）所示，阀门是建立在流通道和控制通道交叉处的微小开关，且需要由外部的空气或流体压力模式驱动以改变其状态. 图 1（b）展示了 CFMBs 的横截面示意图. 具体来说，外部的空气在施压下将从控制端口注入到控制通道进行传输，从而驱使阀门向下挤压流通道以阻止流动样品的移动. 当释放控制通道中的空气/流体压力时，阀门将恢复原样，同时来自于流端口的外部压力将促使流体继续沿着流通道进行传输. 换句话说，阀门的工作原理如同微型开关. 以阀门作为基本的控制单元，可以构建具有复杂功能的 CFMBs[7-8].



(a) CFMBs 的3维结构示意图　(b) CFMBs横截面示意图

Fig. 1　Schematic diagram of the basic structure of CFMBs

图 1　CFMBs 基本结构示意图

随着制造技术的发展，成千上万的阀门已经可以被集成到一个只有几平方厘米的单一芯片中，密度可以达到每平方厘米 100 万个[9]. 然而，由于受到芯片面积以及组件尺寸的限制，为每个阀门分配 1 个独立的控制端口是不切实际的，其将会导致芯片的制造复杂性以及成本大大增加. 为了克服上述挑战，采用基于多路复用器的控制逻辑来控制芯片中的阀门已经成为了一种必不可少的选择.

图 2 展示了一个具有控制逻辑的完整生物芯片平台[10]，其中平台的中心为包含了 116 个阀门的连续微流控生物芯片，底部的核心输入则提供指导阀门打开或关闭的控制信号. 围绕在芯片周围的控制通道、多路复用器、核心输入以及右侧的控制端口共同形成一个控制逻辑，从而生成用于切换芯片中阀门状态的控制模式. 当执行生化测定时，该平台不是直接使用 116 个控制端口，而是部署了一个仅包含 14 个控制端口的多路复用器来生成控制模式以实现

对所有阀门的状态切换. 通常来说，$2^{n/2}$个阀门仅需要被具有 n 个控制端口的多路复用器进行控制[10]. 因此，基于多路复用器的控制逻辑在一定程度上减少了连续微流控生物芯片中控制端口的使用数量. 由于在执行生化测定的过程中，芯片内部的阀门需要频繁切换，因此在控制逻辑的设计过程中需要仔细规划和构建大量的控制路径. 这些路径在微小的区域内相互交叉以形成一个高度集成的控制通道网络，且主导着生物芯片的性能.

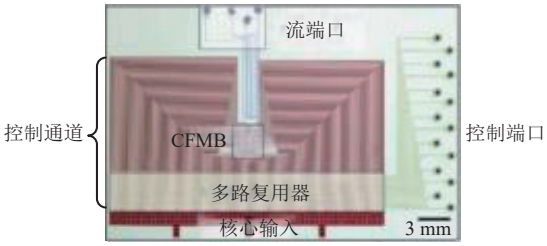


Fig. 2 General platform of a complete biochip

图 2 完整的生物芯片通用平台

为了从根本上提高生物芯片的性能，同时降低芯片的制造成本，针对控制逻辑的通道布线设计应当考虑以下问题：1)应尽量减少连接芯片内每个阀门的控制路径长度以降低信号传输时延，从而缩短切换阀门的响应时间. 由于控制通道是采用 PDMS 材料制造而成的，受到此材料物理特性的影响，压力信号在通道中的传播速度较低[11]. 然而，CFMBs 中的生化反应通常需要非常高的灵敏度，因此在一些生化实验操作中，阀门需要做出快速响应. 通常来说，连接阀门的控制路径增长将会导致该路径下信号的传输时延增大，那么阀门对信号的响应速度也会相应降低，且较慢的响应速度将增加最终生化反应的完成时间. 因此，信号传输时延是评估控制逻辑架构设计的一个重要指标. 2)需要尽可能减少控制逻辑中所需的控制通道总长度以进一步提高 CFMBs 的执行效率，同时降低芯片的制造成本. 由于现有的控制逻辑设计仍采用仅有水平和垂直布线方向的曼哈顿布线方式来实现控制通道网络构建，因此在一定程度上增加了控制逻辑所需的通道总长度. 此外，控制通道总长度的增加同样也导致了信号传输时延的增大，从而降低了生物芯片的执行效率. 因此，本文在原有曼哈顿布线方式的基础上引入对角型的布线方式以实现控制逻辑中通道网络的构建. 需要注意的是，当前的制造技术已经支持对 PDMS 微通道实现对角型布线[12-13].

目前为止，已经出现了一些针对基于多路复用器控制逻辑的相关工作. 此类控制逻辑架构首先是由 Thorsen 等人[7] 所提出的，该架构由二进制阀门的组合阵列所构成，且通过 $2N$ 个控制通道以分别驱动 $2N$ 个流通道的状态. Fidalgo 等人[10] 进一步提出采用控制通道来对阀门进行定位的控制逻辑架构，该架构通过部署 $2N$ 个基于二进制编码的控制端口以实现对 $2N$ 个阀门的单独控制. Wang 等人[14] 提出了一种基于汉明距离的阀门切换优化方法，通过调整控制逻辑内部阀门所需的模式顺序来优化阀门的切换顺序，从而提高控制逻辑的可靠性. 在此基础上，Wang 等人[15] 提出一种基于异或的压力刷新方法，从而克服控制逻辑内部阀门切换时产生的压力偏差问题. Zhu 等人[16] 提出了一种称为多通道控制的合成方法来生成特定的控制逻辑，该控制逻辑可以同时对一些控制通道进行寻址，然后利用剩余的编码容量引入备用通道，从而提高控制逻辑的容错能力. Huang 等人[17] 提出了一种基于强化学习的控制逻辑合成方法以进一步提高阀门的切换效率并减少了架构设计的成本. 此外，Liang 等人[18] 提出了一种基于斯珀纳定理的组合编码策略以减少控制逻辑的阀门和控制通道的数量，从而进一步提高了控制逻辑的可靠性.

尽管上述方法在一定程度上优化了现有的控制逻辑架构，但尚未对控制逻辑通道布线方面存在的问题进行考虑，且这些方法中逻辑架构的控制通道仍采用曼哈顿的布线方式，因此一定程度上限制了控制通道网络构建的灵活性. 为了从根本上解决上述问题，本文研究了非曼哈顿布线方式在 CFMBs 控制逻辑布线中的应用，并提出了一种基于深度强化学习的控制逻辑布线算法.

本文的主要贡献包括 4 个方面：

1)为 CFMBs 的控制逻辑提出了第 1 个同时考虑信号传输时延和通道总长度的控制通道网络构建问题模型，这对于提高控制逻辑的布线设计自动化以及性能优化具有重要意义.

2)针对 CFMBs 提出了一种基于深度强化学习(deep reinforcement learning, DRL)的控制逻辑布线算法. 该算法采用竞争深度 Q 网络(dueling deep Q-network, DuelingDQN)架构对信号传输时延和通道总线长进行权衡评估，通过减少阀门的控制路径长度以实现有效的通道复用，从而同时降低信号传输时延和控制设计成本.

3)首次将具有对角型通道的控制通道网络集成到生物芯片的控制逻辑中，从而进一步提高了阀门切换操作的效率，同时降低了逻辑架构的整体成本.

4)使用 6 个测试用例来评估所提出的布线算法. 实验结果表明，所提出的算法能够为 CFMBs 生成具有较短信号传输时延以及较低芯片设计成本的优化控制逻辑架构.

1 问题描述

1.1 控制逻辑的工作原理

如前所述，在 CFMBs 上执行生化测定时，需要一个控制逻辑生成控制模式，从而更新流层中的阀门(后续称为流阀门)的状态，以便生化操作能够根据用户预先制定的测定计划有效执行[19]. 如图 3 所示，本文以一个用于驱动 3 个流阀门的逻辑架构示意图为例来说明控制逻辑的工作原理. 在该架构中，核心输入部分和各流阀门由控制通道进行连接，从而传输驱动流阀门状态切换的控制信号. 4 个连接到外部压力源的控制端口 x_1，$\bar{x}_1$，x_2，$\bar{x}_2$ 被部署在右侧以引导构建在控制通道上方的控制阀门状态. 需要注意的是，控制端口的压力值通常是互补的[10]，这意味着一对互补的控制端口，例如 x_1 和 $\bar{x}_1$，仅有 1 个端口可被设置为高压. 因此，通过控制端口的组合可以生成指定各控制通道连接情况的控制模式，从而实现对应流阀门的状态更新.

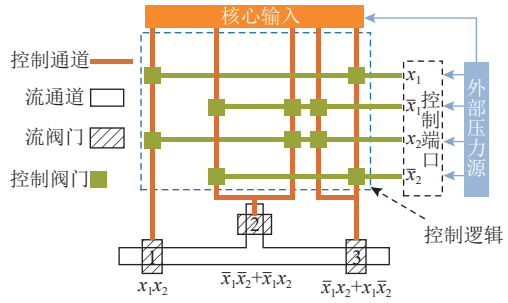


Fig. 3 Control logic architecture for driving three flow valves

图 3 驱动 3 个流阀门的控制逻辑架构

例如，图 3 的控制逻辑中流阀门 1 连接了 1 条控制通道，而流阀门 2, 3 则分别连接了 2 条控制通道，其中连接流阀门 1 的通道被分配了控制模式 x_1x_2. 一旦控制端口 x_1 和 x_2 被设置为高压时，表示为逻辑值 1，那么这 2 个端口所连接的控制阀门被打开，且控制逻辑左侧的通道将被连通，使得流阀门 1 的状态能够更新为与核心输入的值相同. 相反，流阀门 2, 3 的状态则保持不变，因为剩余 2 个控制端口 $\bar{x}_1$ 和 $\bar{x}_2$ 都为低压，表示为逻辑值 0. 此时连接流阀门 2, 3 的控制通道都未被连通，那么来自于核心输入的控制信号则未能传输至这 2 个流阀门以实现状态切换. 需要注意的是，连接流阀门 2 的 2 条控制通道被分别分配了控制模式 $\bar{x}_1x_2$ 和 $\bar{x}_1\bar{x}_2$，其中控制模式 $\bar{x}_1x_2$ 同样也分配给了连接流阀门 3 的控制通道. 一旦控制模式 $\bar{x}_1x_2$ 被激活，流阀门 2 和流阀门 3 将同时进行状态切换. 此外，该控制逻辑仍然保留了流阀门 2 和流阀门 3 单独切换的功能. 这 2 个流阀门能够通过分别激活控制模式 $\bar{x}_1\bar{x}_2$ 和 $x_1\bar{x}_2$ 以实现独立的状态切换. 基于上述方式，所有流阀门的状态都需要激活对应的控制模式进行更新，从而保证用户能够编程一系列由多个控制模式组成的模式序列以自动执行生化测定.

1.2 控制逻辑的通道布线

与传统电子系统中的布线问题不同，例如在印刷电路和超大规模集成电路设计中，信号传输是电子化的，而 CFMBs 中的控制信号是由外部压力产生的，该信号受到 PDMS 材料的影响，在控制通道中的传输速度相对较低(通常为 10 mm/s)[20]. 当控制信号从核心输入处注入时，CFMBs 中的流阀门不能立即关闭，因此信号传输延迟的现象在流阀门的切换过程中普遍发生. 为了评估延迟的程度，需要对信号传输时延 T_p 进行计算，其定义为：

$$T_p = L/v_p, \tag{1}$$

其中 L 为控制路径的长度，v_p 为控制信号在控制通道中的传输速度(在本文实验中 v_p 同样设置为 10 mm/s).

然而，较长的信号传输时延将导致 2 个问题：1)流阀门切换的响应时间较长. 由于压力信号在控制通道中存在传输延迟，阀门不能对信号做出即时反应，缓慢的阀门切换不仅增加了生化反应的完成时间，而且对于时间灵敏度要求高的生化反应而言，较慢的阀门响应可能会影响最终实验结果的质量. 2)较长的控制模式设置时间. 为了确保所有的阀门完全切换到所需的状态，在执行 2 个连续的阀门控制模式之间必须插入 1 个模式设置时间，该时间必须大于最长控制路径所造成的延迟，那么控制模式引导的所有阀门才能被设置到合适的状态. 由于该时间间隙发生在激活每个控制模式之前，因此同样需要通过优化信号传输时延以减少控制模式设置时间. 综上所述，如何有效降低控制信号的传输时延已成为 CFMBs 控制逻辑设计中的一个关键问题.

另一方面，在 CFMBs 控制逻辑的架构设计中，应合理地规划和构建控制通道网络，以便所有的流阀门能够按实际需求连接至控制逻辑，从而实现控制信号的有效传输. 此外，由于控制通道网络的构建直接决定了最终的控制逻辑架构，因此在通道布线

阶段还应优化控制逻辑的通道总长度,从而有效降低芯片的制造成本并进一步提高其执行效率. 为此,可以在原有逻辑架构中加入对角型的通道布线方式,同时通过有效规划控制通道以实现不同控制路径之间的资源共享,从而为 CFMBs 控制逻辑构建一个高效的控制通道网络.

需要注意的是,虽然通过减少控制通道总长度能够一定程度上优化信号传输时延,但并不能仅从该方面进行优化以实现传输时延的减小. 实际上信号传输时延还受到核心输入与流阀门之间控制路径长度的影响. 此外,所有控制路径的长度总和并不等同于控制通道的总长度. 这是因为不同的控制路径会存在复用同一段控制通道的情况,复用的控制通道越多,那么所需的控制通道总长度也会相应地减少. 然而,如果某条控制路径仅通过尽可能复用其他路径使用的控制通道段(即,减少控制通道总长度)以优化信号传输时延,那么可能会导致该路径并不会采用最短路径的布线方式连接到流阀门,而是为了实现更多的通道复用采取绕路的布线方式,从而导致该路径的信号传输时延增大. 因此本文算法需要从信号传输时延和通道总线长 2 个方面进行权衡评估,以生成一个合适的控制逻辑布线方案.

1.3 问题模型

基于上述分析,针对 CFMBs 控制逻辑的控制通道网络构建问题模型表述如下:

输入:控制逻辑中控制阀门的位置,CFMBs 中流阀门的位置,驱动所有流阀门的模式序列.

输出:考虑信号传输时延的优化控制逻辑架构.

目标:最小化控制信号传输时延,最小化控制通道总长度.

2 算法设计

2.1 算法的提出

DRL 是一种将强化学习与深度学习相结合的机器学习方法. 在深度强化学习中,智能体通过观察环境的状态,实现动作与环境的交互,从而获得奖励信号[21]. 智能体的目标是通过与环境的不断交互来学习如何在给定任务中做出正确的决策以最大化累积奖励.

上述过程通常被表述成一个马尔可夫决策过程(Markov decision process, MDP)[22],通常定义为(S, A, T, R),其中 S 是一个包含了智能体所有可感知状态的集合,A 是一个包含了智能体所有可采取动作的集合,S 和 A 分别称为状态空间和动作空间;T 是一个状态转移概率,表示在给定状态和动作下,智能体在下一时刻转移到某个状态的概率;R 是一个奖励函数,该函数定义了智能体在特定状态下采取特定动作后所获得的奖励. DRL 的基本流程如下所述:在时间步 t 下,智能体首先从环境中感知当前状态 s_t 并从 A 中选取一个动作 a_t. 然后,智能体通过执行动作 a_t 后从环境中获得奖励 r_t,并转移到下一个状态 s_{t+1}. 之后,智能体再根据新状态 s_{t+1} 选择新的动作. 这个过程会不断地迭代,直至智能体学习到一个有效的策略. 到目前为止,DRL 在许多领域都取得了重要的突破[23],如机器人控制[24] 和自动驾驶[25] 等.

为了构建一个高效的控制通道网络,本文提出了一种基于 DRL 的连续微流控生物芯片控制逻辑布线算法. 图 4 展示了该算法的整体设计流程,其主要由 DRL 框架构建以及网络架构训练 2 部分组成. 本文将所有控制阀门、流阀门的位置以及模式序列作为输入,并根据输入信息初始化布线网格. 图 5(a)(b)分别展示了流阀门 f_1 至 f_5 对应的模式序列以及根据阀门位置初始化后的布线网格,其中图 5(a)中从顶部核心输入沿着箭头方向到达底部某个流阀门所经

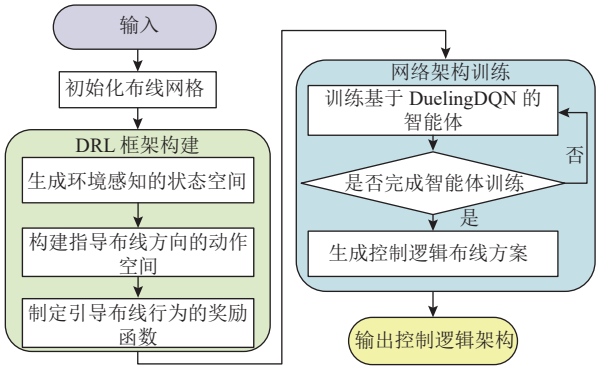
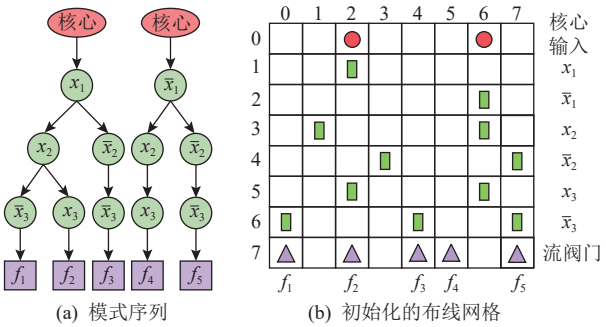


Fig. 4 Flow chart of DRL-based control logic routing algorithm

图 4 基于 DRL 的控制逻辑布线算法流程图



(a) 模式序列 (b) 初始化的布线网格

Fig. 5 Pattern sequences of flow valves f_1 to f_5 and initialized routing grid

图 5 流阀门 f_1 至 f_5 的模式序列以及初始化的布线网格

过的中间节点组成该流阀门的控制模式. 例如，流阀门 f_1 的控制模式为 $x_1x_2\bar{x}_3$.

然后，将 CFMBs 控制逻辑布线问题构建成符合 MDP 求解的 DRL 框架. 具体来说，首先为 DRL 智能体生成一个可感知控制逻辑布线环境的状态空间. 其次根据可用的控制通道布线角度构建动作空间，从而为智能体提供布线方向. 当动作空间构建完成后，制定能够引导智能体布线行为的奖励函数. 最后根据构建完成的 DRL 框架，对基于 DuelingDQN 的 DRL 智能体进行训练以获得一个有效的控制逻辑布线方案，从而输出优化的控制逻辑架构. 本文算法所涉及的更多技术细节，如上述的状态空间、动作空间、奖励函数设计、网络架构训练将会在后续进行介绍.

2.2 DRL 框架构建

2.2.1 状态空间设计

为了使 DRL 智能体能够准确感知到环境的信息，本文将时刻 t 下的状态表示为 s_t，该状态不仅取决于当前控制路径到达的位置，而且需要对各条控制路径进行区分. 原因是针对处于相同位置的不同控制路径，智能体选择的合理动作也可能不同. 如果不对控制路径进行区分，那么智能体将无法识别控制路径所适用的布线策略，从而导致最终布线策略失效. 因此，整个状态空间可以表述为

$$S=\left\{s_t|s_t=[p_t^{\mathrm{n}},p_t^{\mathrm{v}},p_t^{\mathrm{e}},p_t^{\mathrm{i}}],p_t^{\mathrm{v}}=(v_t^x,v_t^y),p_t^{\mathrm{e}}=(e_t^x,e_t^y),p_t^{\mathrm{i}}=(i_t^x,i_t^y)\right\},\tag{2}$$

其中 p_t^{n} 表示时刻 t 下正在规划的控制路径序号，p_t^{v} 表示时刻 t 下控制路径到达的位置，p_t^{e} 为时刻 t 下控制路径需要到达的目标点位置，p_t^{i} 为时刻 t 下控制路径需要到达的必经点位置. 上述位置信息由布线网格中各点的 x, y 坐标构成.

根据给定的模式序列，各条控制路径将会依次经过对应控制模式指定的必经点，并最终到达目标点，其中，必经点为控制逻辑中的控制阀门，目标点为 CFMBs 中的流阀门. 需要注意的是，状态 s_t 中的必经点位置 p_t^{i} 将会根据实际的布线情况进行更新. 如果智能体在时刻 t 下采取动作到达当前需要经过的必经点，那么在下一时刻，即时刻 t+1，状态 s_{t+1} 中的 p_{t+1}^{i} 将更新为下一个需要经过的必经点位置. 最终，当前规划的控制路径在到达流阀门后结束布线，同时智能体开始为下一条控制路径的布线做准备，直到控制逻辑中所有的控制路径都完成布线.

2.2.2 动作空间设计

本文将时刻 t 下所选择并执行的动作表示为 a_t，其中动作 a_t 表示控制路径的布线行为. 针对控制逻辑的控制通道布线阶段，由于本文在原有曼哈顿布线方式的基础上引入了对角型的布线方式，因此动作空间可以表述成

$$A=\left\{a_t|a_t\in\{a_{\mathrm{r}},a_{\mathrm{dr}},a_{\mathrm{d}},a_{\mathrm{dl}},a_{\mathrm{l}}\}\right\},\tag{3}$$

其中 a_{r}, a_{dr}, a_{d}, a_{dl}, a_{l} 则分别表示智能体的 5 个布线方向，分别为 0°, 45°, 90°, 135°, 180°, 如图 6 所示. 当智能体在时刻 t 执行动作 a_t 后，控制路径将依据动作 a_t 所表示的布线方向通过 1 个网格. 此外，如果布线方向为非对角型，那么控制路径的长度将增加 1 个单位长度，即 1 mm，否则控制路径长度将增加 1.4 mm.

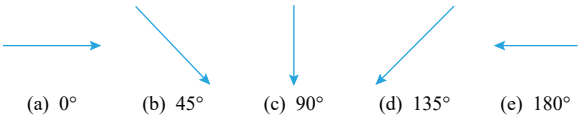


Fig. 6 Schematic diagram of routing angles for control logic

图 6 控制逻辑布线角度示意图

2.2.3 奖励函数设计

奖励是评价智能体所选择动作质量的直接信号. 为了引导 DRL 智能体探索并学习到有效的布线策略，本文的奖励函数将基于智能体执行动作后控制路径所到达的点的类型进行构建. 具体来说，控制路径到达的点主要分为 5 类，分别为上述的必经点 p^{i}、目标点 p^{e}、障碍点 p^{o}、共享点 p^{s}、普通点 p^{c}. 这 5 类节点的定义分别如下：

1）必经点. 控制路径根据给定的模式序列在布线过程中必须经过的特定位置，这类点通常为控制阀门.

2）目标点. 控制路径在布线过程中需要到达的终点，这类点为 CFMBs 中的流阀门.

3）障碍点. 控制路径在布线过程中需要避开的位置，这类点通常为其他路径的必经点以及目标点.

4）共享点. 多条控制路径汇聚或相交的位置，且该位置不能存在控制阀门.

5）普通点. 为布线网格内除了必经点、目标点、障碍点等特殊点之外的其他普通节点.

本文将时刻 t 下智能体在状态 s_t 执行动作 a_t 后获得的即时奖励表示为 r_t，因此总体的奖励函数可以表示为

$$r_t=\begin{cases}c_{\mathrm{p}}^1,\ p_{t+1}^{\mathrm{v}}=p^{\mathrm{i}},\\ c_{\mathrm{n}}^1,\ p_{t+1}^{\mathrm{v}}=p^{\mathrm{o}},\\ c_{\mathrm{p}}^2,\ p_{t+1}^{\mathrm{v}}=p^{\mathrm{e}},\\ R_{\mathrm{p}}(p_{t+1}^{\mathrm{v}}),\ p_{t+1}^{\mathrm{v}}=p^{\mathrm{c}},\\ R_{\mathrm{p}}(p_{t+1}^{\mathrm{v}})+c_{\mathrm{p}}^3,\ \text{其他},\end{cases}\tag{4}$$

其中p_{t+1}^{v}为时刻t下智能体执行动作a_t后控制路径到达的位置，c_{p}^1，c_{p}^2，c_{p}^3分别为3个正的常数，c_{n}^1为负的常数，$R_{\mathrm{p}}(p_{t+1}^{\mathrm{v}})$是$p_{t+1}^{\mathrm{v}}$为普通点$p^{\mathrm{c}}$的奖励函数.

图7展示了图5(b)中流阀门f_1至f_5对应的控制路径，本文以图中流阀门f_5的控制路径(虚线)为例来说明奖励的计算，其中该路径的序号为50，且对应的控制模式为$\bar{x}_1\bar{x}_2\bar{x}_3$. 在该例子中，假设时刻$t$下路径50的必经点为控制阀门$\bar{x}_2$(上对角线网纹方格). 如果执行$a_t$后$p_{t+1}^{\mathrm{v}}$到达控制阀门$\bar{x}_2$，正向奖励$c_{\mathrm{p}}^1$将被反馈给智能体以鼓励其经过该必经点.

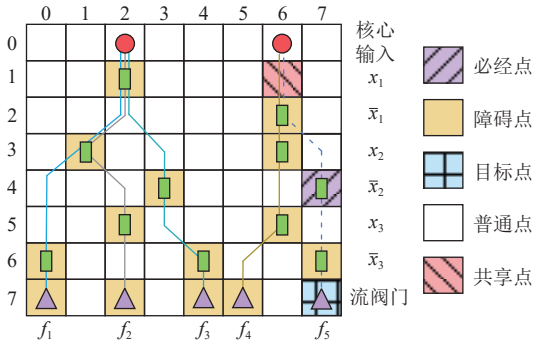


Fig. 7 Schematic diagram of control paths of flow valves f_1 to f_5

图7 流阀门f_1至f_5的控制路径示意图

同时，除了当前属于必经点的控制阀门$\bar{x}_2$外，网格中其余的控制阀门以及其他路径需要经过的流阀门都不可被控制路径50经过，因此这些点将被视为障碍点p^{o}(灰底方格). 如果执行a_t后p_{t+1}^{v}为障碍点p^{o}，那么当前状态s_t将在下一时刻转移成终止状态，且智能体将获得惩罚值c_{n}^1以引导其在布线过程中避开这类点.

此外，如果智能体在状态s_t执行a_t后可以到达目标点p^{e}(灰底十字网纹方格)，其将会获得一个正向奖励c_{p}^2以鼓励到达目标点的行为，即完成该条路径的构建. 需要注意的是，c_{p}^2小于上述的c_{p}^1，这是因为c_{p}^2设置方式能够引导控制路径根据控制模式优先经过属于必经点的控制阀门，到达属于目标点的流阀门，从而保证路径正确的控制功能.

另一方面，针对路径50中p_{t+1}^{v}为普通点(白底方格)的情况，引入关键点p^{k}以实现奖励计算，其中关键点p^{k}包括了必经点p^{i}以及目标点p^{e}. 相关的奖励函数表示为

$$R_{\mathrm{p}}(p_{t+1}^{\mathrm{v}})=\begin{cases}c_{\mathrm{n}}^2, md(p_{t+1}^{\mathrm{v}},p_t^{\mathrm{k}})<md(p_t^{\mathrm{v}},p_t^{\mathrm{k}}),\\ c_{\mathrm{n}}^3, md(p_{t+1}^{\mathrm{v}},p_t^{\mathrm{k}})=md(p_t^{\mathrm{v}},p_t^{\mathrm{k}}),\\ c_{\mathrm{n}}^4, \text{其他},\end{cases}\tag{5}$$

其中c_{n}^2，c_{n}^3，c_{n}^4为3个负的常数，且$c_{\mathrm{n}}^2>c_{\mathrm{n}}^3>c_{\mathrm{n}}^4$，$md(p_{t+1}^{\mathrm{v}},p_t^{\mathrm{k}})$和$md(p_t^{\mathrm{v}},p_t^{\mathrm{k}})$分别为智能体在时刻$t$执行动作前后控制路径到达的位置与当前关键点之间的曼哈顿距离. 根据布线情况，关键点p^{k}将随着必经点p^{e}同步更新. 需要注意的是，当控制路径到达最后一个必经点时，关键点p^{k}将更新为目标点p^{e}.

从上述奖励函数可以看出，智能体在执行a_t后p_{t+1}^{v}到达普通点p^{c}时存在3种情况：1)如果执行a_t后使得控制路径更接近关键点，那么智能体将会获得一个较小的惩罚值c_{n}^2；2)如果执行a_t后，控制路径到达的位置与关键点的曼哈顿距离保持不变，此时智能体会获得一个较大的惩罚值c_{n}^3，因为该行为没有带来更好的布线结果；3)如果执行a_t后使得控制路径更远离关键点，此时智能体将会获得一个更大的惩罚值c_{n}^4，因为该行为导致了更差的布线结果. 在控制逻辑的布线过程中，该惩罚机制将为智能体提供有效信号以指引控制路径靠近关键点，且保证了每条控制路径以最短的布线方式连接对应的流阀门，从而有效降低了控制信号的传输时延.

由于布线网格上的2个控制阀门之间可能存在多条路径，为了减少控制逻辑的控制通道总长度，本文基于式(4)附加一个正向奖励c_{p}^3，从而鼓励智能体引导控制路径在靠近目标点的同时尽可能地使用网格上先前其他控制路径所占用的区域，即共享点p^{s}. 例如，控制路径50的共享点为图7中的右下对角线灰底网纹方格. 通过该方式，智能体将有效规划控制通道以实现不同控制路径之间的资源共享，从而降低控制逻辑的总成本. 此外，从该奖励函数可能获得的奖励值可以看出，如果智能体选择了一个远离关键点的共享点，那么其获得的惩罚(“−4”)甚至大于智能体选择一个靠近关键点的普通点所获得的惩罚(“−2”)，因此智能体在整个奖励函数的引导下更偏向于首先确保控制路径长度的减少，然后再考虑通道资源的复用. 换句话说，优化信号传输时延的优先级整体上是大于优化通道总长度的.

2.3 网络架构训练

基于上述所设计的状态空间、动作空间以及奖励函数，本文采用了DRL方法中的DuelingDQN网络架构[26]作为智能体对信号传输时延和通道总长度进行权衡评估，从而实现控制通道的自动化布线. 为了学习到一种有效的布线策略，智能体需要在训练过程中计算时刻t下状态动作对(s_t,a_t)的Q值，并对所有状态动作对的Q值进行迭代更新. 最终，智能体对

各时刻下所感知的状态分别选择具有最大 Q 值的动作来组成布线策略.

表 1 展示了本文所采用 DuelingDQN 架构的基本配置，包括网络层数、网络类型、神经元数量以及各网络层使用的激活函数，其中输入层与输出层的神经元数量即为各自对应的维度. 输入层的 7 个维度分别对应智能体状态中包含的 7 个元素. 此外，输出层的 5 个维度则分别对应智能体可选择的 5 个布线方向. DuelingDQN 通过改变架构中称为目标网络的神经网络结构来优化算法，它将用于计算时刻 t 下状态动作对 (s_t,a_t) 的 Q 值动作值函数拆分成了与状态相关的价值函数 V，以及与动作相关的优势函数 D. 具体来说，其主要是在目标网络的全连接层使用 2 个分离流，从而对状态值函数和优势函数进行单独计算，最后将 2 个分离流进行汇总以输出单一 Q 值. 因此针对 Q 值的计算方式可以表示成

$$Q(s_t,a_t,\theta_t,\alpha_t,\beta_t)=V(s_t,\theta_t,\beta_t)+D(s_t,a_t,\theta_t,\alpha_t), \tag{6}$$

其中 $V(s_t,\theta_t,\beta_t)$ 和 $D(s_t,a_t,\theta_t,\alpha_t)$ 分别是价值函数和优势函数的输出. θ_t 指的是时刻 t 下目标网络的参数，β_t 是价值函数部分的参数，α_t 是优势函数部分的参数. 可以看出 $V(s_t,\theta_t,\beta_t)$ 仅与状态有关，而 $D(s_t,a_t,\theta_t,\alpha_t)$ 与状态和动作都有关. 但是式(6)的 Q 值计算方式无法辨识最终输出里面 $V(s_t,\theta_t,\beta_t)$ 和 $D(s_t,a_t,\theta_t,\alpha_t)$ 各自的作用，从而严重地降低了神经网络的性能.

Table 1 Configuration of DuelingDQN Architecture

表 1 DuelingDQN 架构配置

网络层数	网络类型	神经元数量	激活函数
1	输入层	7	
2	隐层（全连接）	64	ReLu
3	隐层（全连接）	128	ReLu
4	隐层（全连接）	128	ReLu
5	输出层	5	

为了解决上述不可辨识性的问题，将优势函数部分引入平均操作，因此最终 Q 值的计算方式为

$$Q(s_t,a_t,\theta_t,\alpha_t,\beta_t)=V(s_t,\theta_t,\beta_t)+\left(D(s_t,a_t,\theta_t,\alpha_t)-\sum_{a\in A}D(s,a,\theta_t,\alpha_t)/|A|\right), \tag{7}$$

其中 $|A|$ 表示动作空间 A 中的动作总数，且该 Q 值也称为目标网络的预测 Q 值.

然后，目标网络利用动作 a_t 以及下一个时刻状态 s_{t+1} 来计算状态动作对 (s_t,a_t) 的目标 Q 值 G_t. 需要注意的是，智能体在状态 s_t 下采取动作 a_t 就能够获得下一时刻的状态 s_{t+1}，即使当前状态 s_t 并未进行转移. 因此，状态动作对 (s_t,a_t) 的目标 Q 值计算方式为

$$G_t=r_t+\gamma Q(s_{t+1},a_t,\theta_t,\alpha_t,\beta_t), \tag{8}$$

其中 r_t 是在状态 s_t 下采取动作 a_t 后获得的奖励，γ 表示折扣因子，G_t 用于评估所采取动作 a_t 的质量. 通常来说，γ 用于权衡当前和未来获得的奖励. 当 γ 接近 1 时，智能体更关注长期回报；否则，智能体更关注即时奖励[22].

为了使智能体在学习过程中不断向正确的策略调整，需要基于上述预测 Q 值以及目标 Q 值对目标网络的参数进行更新，因此需要为神经网络构建损失函数 $L(\theta)$ 为：

$$L(\theta)=E[(G_t-Q(s_t,a_t,\theta_t,\alpha_t,\beta_t))^2]. \tag{9}$$

然后，通过采用梯度下降法来最小化该损失函数[25]，从而实现目标网络中参数的更新.

基于上述技术细节，具体的算法如算法 1 所示.

算法 1. 基于深度强化学习的控制逻辑布线算法.

输入：控制阀门位置，流阀门位置，模式序列；

输出：最小化信号传输时延以及通道总长度的控制逻辑布线方案.

① 根据输入信息构建并初始化布线网格 G_r；
② 初始化目标网络参数 θ；
③ 初始化训练批次 B_S，训练回合 E_P，学习率 ∂，
　动作选择概率 ε 以及目标网络更新周期 U_c；
④ for 每一个训练回合 E_P
⑤ 　重置环境参数；
⑥ 　if 状态 s_t 不是终止状态
⑦ 　　随机生成概率值 V_P；
⑧ 　　if $V_P>\varepsilon$
⑨ 　　从动作空间 A 中随机选择动作 a_t；
⑩ 　　else
⑪ 　　选择 Q 值最大的动作 a_t；
⑫ 　　end if
⑬ 　　执行 a_t 并根据式(4)计算对应奖励 r_t；
⑭ 　　获取下一时刻状态 s_{t+1}；
⑮ 　　元组 $(s_t,\ a_t,\ r_t,\ s_{t+1})$ 存入经验回放区；
⑯ 　　if $E_P \bmod U_c=0$
⑰ 　　从经验回放区随机选取 B_S 个元组；
⑱ 　　采用 ∂ 以最小化式(9)来更新 θ；
⑲ 　　end if
⑳ 　　将状态 s_t 更新为下一时刻状态 s_{t+1}；
㉑ 　end if
㉒ end for

从上述算法步骤中可知, 首先需要根据模式序列以及所有阀门的位置来构建并初始化布线网格G_r. 其次, 需要对 DRL 布线方法中的网络参数以及使用的超参数进行初始化. 然后, 智能体感知状态s_t, 并采用ε贪婪策略来选取动作. 具体来说, 智能体将以ε的概率选择具有最大 Q 值的布线方向, 否则将从动作空间 A 随机选择 1 个布线方向. 利用该策略, 智能体能够更好地权衡动作的开发和探索, 从而更有机会探索到最优策略. 当执行动作a_t后, 智能体将根据式(4)获得对应的奖励信号, 并相应获得下一时刻状态s_{t+1}. 基于上述结果, 元组(s_t, a_t, r_t, s_{t+1})将被存入经验回放区. 当智能体需要学习新策略时, 即此时目标网络的参数需要更新, 那么将从经验回放区中随机选取B_S个元组作为学习样本以最小化式(9)表示的损失函数. 之后, 智能体感知下一时刻状态s_{t+1}以开始下一轮的环境交互. 当完成所有指定的学习迭代后, 智能体将生成一个最小化信号传输时延以及通道总长度的控制逻辑布线方案.

3 实验结果及分析

本文所提出的基于深度强化学习的连续微流控生物芯片控制逻辑布线算法采用 TensorFlow 1.13.1 框架实现, 并在具有 2.40 GHz 和 8 GB 内存的设备上进行测试. 6 个测试用例用于验证本文算法的有效性, 其中 RC01 是用于实现 RA30 芯片中比色蛋白分析[27]的布局测试用例, RC02 是用于实现 CPA 芯片中体外诊断分析[27]的布局测试用例, RC03 是用于实现 mRNA 芯片中体外诊断分析[28]的布局测试用例, 而 RC04, RC05, RC06 则分别对应 3 种随机合成分析的布局测试用例. 表 2 列出了测试用例的相关信息, 其中从左到右分别给出了控制逻辑占用的网格大小、CFMBs 中流阀门数量、控制逻辑中控制阀门数量. 此外, 表 3 给出了本文算法提及的相关参数取值.

Table 2 Details of Test Cases Used in Our Paper

表 2 本文所用测试用例的详细信息

测试用例	网格大小	流阀门数量	控制阀门数量
RC01	18×34	20	75
RC02	18×43	26	92
RC03	18×53	31	123
RC04	18×59	28	139
RC05	18×82	50	229
RC06	19×98	52	327

Table 3 Setting of Parameters Used in Our Paper

表 3 本文使用的参数设置

参数	取值	参数	取值
c_p^1	20	c_n^4	−10
c_p^2	15	B_S	128
c_p^3	8	E_P	45 000
c_n^1	−40	ε	0.9
c_n^2	−2	U_c	200
c_n^3	−5	∂	10^{-3}

3.1 本文算法有效性验证

由于当前还未有针对 CFMBs 控制逻辑通道布线的相关工作, 因此本文实现了另一种控制逻辑布线算法, 称为 DRA, 以验证本文算法的有效性. 具体来说, DRA 是基于 A*算法来实现逻辑架构中控制通道网络的构建, 且该算法同样以控制信号传输时延以及控制通道总长度作为优化目标. 需要注意的是, 后续实验所评估的控制信号传输时延是由最长控制路径引起的延迟, 即最大信号传输时延. 本文在上述测试用例上分别运行这 2 种算法, 表 4 和表 5 分别展示了本文算法与 DRA 算法在控制通道总长度与信号传输时延指标上的对比结果, 其中“优化率”栏展示了

Table 4 Comparison of the Total Control Channel Length

表 4 控制通道总长度的比较

测试用例	通道总长度/mm		优化率/%
	DRA	本文算法	DRA
RC01	268.2	231.6	13.6
RC02	332.8	296.6	10.9
RC03	455.8	405.2	12.5
RC04	483.0	415.8	13.9
RC05	699.6	618.4	11.6
RC06	903.4	804.0	11.0
平均			12.3

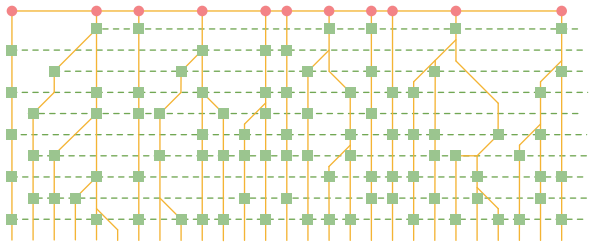
Table 5 Comparison of the Signal Transmission Delay

表 5 信号传输时延的比较

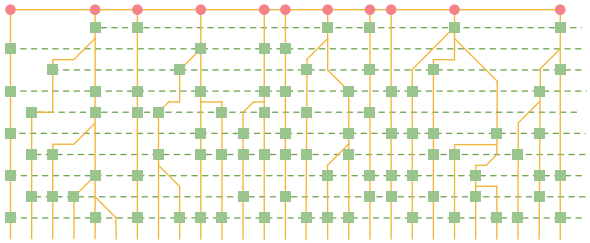
测试用例	时延/s		优化率/%
	DRA	本文算法	DRA
RC01	2.0	1.6	20.0
RC02	1.6	1.4	12.5
RC03	1.8	1.4	22.2
RC04	1.8	1.6	11.1
RC05	2.2	1.8	18.2
RC06	2.0	1.6	20.0
平均			17.3

本文算法相对于 DRA 算法在各指标上的优化程度.

从表 4 看出，相对于 DRA 算法，本文算法在控制通道总长度方面实现了 10.9%~13.9% 的长度缩减，平均减少了 12.3%. 此外，从表 5 可以看出，本文算法相对于 DRA 算法具有更好的时延优化效果，在各测试用例上本文算法实现了 11.1%~22.2% 的时延降低，平均降低了 17.3%. 另外，图 8 展示了本文算法与 DRA 针对测试用例 RC02 分别生成的局部控制逻辑布线结果，从而更直观地表现 2 种方法在布线效果上的差别. 上述对比实验结果表明了本文所提布线算法的有效性，这主要得益于本文算法中的 DRL 智能体倾向于以长远的眼光在布线过程中自适应地调整布线策略，并根据环境的奖励反馈以有效地规划各控制路径的长度，从而降低了控制信号的传输时延. 同时该智能体能够通过先前的布线经验来最大程度地对不同控制路径的通道资源进行资源共享，从而有效地减少了控制通道的总长度，一定程度上保证了最终布线方案的质量.



(a) 由本文算法生成的局部控制逻辑布线结果



(b) 由 DRA 生成的局部控制逻辑布线结果

Fig. 8 Routing results generated by our proposed algorithm and DRA for RC02

图 8 本文算法与 DRA 针对 RC02 生成的布线结果

为了验证本文算法具有一定的泛化能力，我们选择测试用例 RC01~RC04 对单个模型进行训练，并使用训练完成的模型对 RC05 和 RC06 进行测试，然后将生成的布线结果与 DRA 在控制通道总长度以及信号传输时延方面进行对比，对比结果如表 6 所示. 从表 6 看出，相对于 DRA 算法，本文算法在这 2 个测试用例上分别实现了 10.5% 以及 9.1% 的通道长度缩减. 此外，本文算法相对于 DRA 算法仍具有更好的时延优化效果，在这 2 个测试用例上分别实现了 18.2% 以及 20.0% 的时延降低. 上述实验结果表明了本文算法具有一定的泛化能力，其针对未知的测试用例仍能够生成有效的布线方案.

Table 6 Generalization Verification of Our Proposed Algorithm

表 6 本文算法的泛化性验证

测试用例	通道总长度/mm		优化率/%	时延/s		优化率/%
	DRA	本文算法	DRA	DRA	本文算法	DRA
RC05	699.6	625.8	10.5	2.2	1.8	18.2
RC06	903.4	821.0	9.1	2.0	1.6	20.0

3.2 DuelingDQN 架构有效性验证

在所提出的基于 DRL 的控制逻辑布线算法中，采用了一种对末层全连接层进行拆分的 DuelingDQN 架构作为 DRL 智能体，从而高效地实现智能体的学习过程. 为了验证 DuelingDQN 架构的优越性，本文引入了另外 2 种 DRL 架构以代替 DuelingDQN，即深度双 Q 网络（double deep Q-network, DoubleDQN）[29] 和深度 Q 网络（deep Q-network, DQN）[30]，其中 DoubleDQN 是在 DQN 的基础上提出的一种改进模型. 具体来说，DoubleDQN 通过使用 2 个神经网络来减少 Q 值的过高估计问题，其中策略网络用于选择动作，而目标网络则用于评价所选动作的质量. 本文分别比较了 3 种网络架构在控制通道总长度和信号传输时延的实验结果，并在表 7 和表 8 进行展示，其中“优化率”栏分别展示了 DuelingDQN 架构相对于其他 2 种 DRL 架构在各指标上的优化程度.

Table 7 Effectiveness Validation of DuelingDQN with Respect to the Total Control Channel Length

表 7 DuelingDQN 在控制通道总长度方面的有效性验证

测试用例	通道总长度/mm			优化率/%	
	DQN	DoubleDQN	DuelingDQN	DQN	DoubleDQN
RC01	249.8	241.4	231.6	7.3	4.1
RC02	328.7	315.4	296.6	9.8	6.0
RC03	456.8	426.2	405.2	11.3	4.9
RC04	480.2	448.0	415.8	13.4	7.2
RC05	687.2	642.2	618.4	10.0	3.7
RC06	890.4	844.6	804.0	9.7	4.8
平均				10.3	5.1

从表 7 可以看出，DuelingDQN 优于基础的 DQN 架构，在控制通道总长度方面实现了 7.3%~13.4% 的长度缩减，平均缩减了 10.3%. 此外，与改进后的网络

Table 8 Effectiveness Validation of DuelingDQN with Respect to the Signal Transmission Delay

表 8 DuelingDQN 在信号传输时延方面的有效性验证

测试用例	时延/s			优化率/%	
	DQN	DoubleDQN	DuelingDQN	DQN	DoubleDQN
RC01	1.8	1.8	1.6	11.1	11.1
RC02	1.6	1.4	1.4	12.5	0.0
RC03	1.8	1.6	1.4	22.2	12.5
RC04	1.8	1.6	1.6	11.1	0.0
RC05	2.2	1.8	1.8	18.2	0.0
RC06	2.0	1.8	1.6	20.0	11.1
平均				15.6	5.8

架构 DoubleDQN 相比, DuelingDQN 在控制通道总长度方面仍然实现了平均 5.1% 的优化. 另一方面, 如表 8 所示, DuelingDQN 在信号传输时延方面同样优于其余 2 个 DRL 架构, 尤其是与基础的 DQN 架构相比, 本文算法所采用的 DuelingDQN 架构能够实现 11.1%~22.2% 的信号传输时延降低, 平均降低了 15.6%. 而与 DoubleDQN 相比, DuelingDQN 同样实现了信号传输时延的优化, 且在所有测试用例上能够实现平均 5.8% 的传输时延降低. 上述结果表明了所采用的 DuelingDQN 架构具有更优异的性能, 这主要是因为 DuelingDQN 架构相比于另外 2 种架构能够分别对状态价值和动作价值进行估计以计算更为准确的 Q 值, 使得 DRL 智能体的学习过程更加稳定高效, 从而获得高质量的布线方案.

3.3 对角型布线方式的有效性验证

为了验证在控制逻辑中采用对角型通道布线的有效性, 本文基于所提出的 DRL 框架实现了另一种未采用对角型布线方式的算法 MHR(仅采用曼哈顿布线方式), 并与本文算法(采用对角型布线方式的算法)分别在控制通道总长度以及信号传输时延 2 方面进行对比. 对比结果如表 9 和表 10 所示, 其中“优化率”栏展示了采用对角型布线方式相对于未采用对角型布线方式在上述各指标上的优化程度.

从表 9 可以看出, 在相同的 DRL 框架下, 采用对角型布线方式相对于未采用对角型布线方式在各测试用例上能够实现 19.3%~22.6% 的控制通道总长度缩减, 平均缩减了 20.8%. 此外, 如表 10 所示, 采用对角型的布线方式在信号传输时延方面仍优于未采用对角型的布线方式, 实现了 22.2%~36.0% 的传输时延降低, 平均降低了 28.9%. 上述实验结果表明了在控制逻辑中采用对角型布线方式的有效性. 需要注意的是, 虽然对角型布线方式能够有效地优化控制通道总长度以及信号传输时延, 但是如何指导该布线方式的实施本质上是基于本文所提出的深度强化学习框架, 从而能够生成高质量的控制逻辑布线方案. 此外, 在 3.1 节也证明了所提出的深度强化学习方法的有效性. 因此本文中所采用的深度强化学习方法以及对角型布线方式都是使得最终实验结果质量提升的关键因素, 二者缺一不可, 前者基于后者以进一步提升实验结果的质量, 而后者需要依赖于前者才能够有效发挥其自身作用.

Table 9 Effectiveness Validation of Diagonal Routing with Respect to the Total Control Channel Length

表 9 对角型布线在控制通道总长度方面的有效性验证

测试用例	通道总长度/mm		优化率/%
	MHR	本文算法	MHR
RC01	292.0	231.6	20.7
RC02	383.0	296.6	22.6
RC03	509.0	405.2	20.4
RC04	534.0	415.8	22.2
RC05	768.0	618.4	19.5
RC06	996.0	804.0	19.3
平均			20.8

Table 10 Effectiveness Validation of Diagonal Routing with Respect to the Signal Transmission Delay

表 10 对角型布线在信号传输时延方面的有效性验证

测试用例	时延/s		优化率/%
	MHR	本文算法	MHR
RC01	2.1	1.6	23.8
RC02	1.8	1.4	22.2
RC03	2.1	1.4	33.3
RC04	2.2	1.6	27.3
RC05	2.6	1.8	30.8
RC06	2.5	1.6	36.0
平均			28.9

另一方面, 表 11 列出了本文算法针对每个测试用例所需的训练时间和推理时间, 以及 2 种基线算法 DRA 和 MHR 的运行时间, 其中推理时间指的是训练完成后的模型对各测试用例生成有效布线方案所需的时间. 需要注意的是, 表 11 中基线算法 MHR 的运行时间为训练完成后模型的推理时间. 从表 11 中可以看出, 虽然本文算法在模型训练阶段需要花费较多的时间, 但训练好的模型能够在较短的计算时间内针对各测试用例生成高质量的布线方案, 从而体现了本文算法在执行效率方面的有效性.

Table 11 Training Time and Inference Time of Our Proposed Algorithms and Running Time of DRA and MHR

表 11 本文算法的训练和推理时间以及 DRA 和 MHR 的运行时间

测试用例	训练时间/min	推理时间/s	运行时间/s	
	本文算法	本文算法	DRA	MHR
RC01	90.3	0.1	0.1	0.1
RC02	112.5	0.1	0.2	0.1
RC03	150.2	0.1	0.3	0.1
RC04	172.5	0.1	0.3	0.2
RC05	225.2	0.2	0.3	0.2
RC06	337.5	0.2	0.4	0.2

4 结　论

本文针对 CFMBs 控制逻辑研究了同时考虑信号传输时延和通道总长度的控制通道网络构建问题，并提出了第 1 个基于深度强化学习的连续微流控生物芯片控制逻辑布线算法. 该算法采用 DuelingDQN 架构实现了对信号传输时延和通道总线长的权衡评估，旨在自动构建一个高效的控制通道网络. 此外，通过引入对角型的控制通道布线，为控制逻辑的通道布线阶段提供了巨大的灵活性，从而进一步降低了芯片的制造成本并提高其执行效率. 6 个测试用例被用于验证本文算法的有效性. 实验结果表明，本文算法能够有效构建高性能、低成本的控制逻辑架构.

作者贡献声明：蔡华洋提出初步方案、实验设计，撰写和修改论文；黄兴负责论文思路构建、理论指导和论文修改；刘耿耿负责论文方案分析、理论指导、论文润色和修改.

参　考　文　献

[1] Huang Xing, Ho T, Guo Wenzhong, et al. Computer-aided design techniques for flow-based microfluidic lab-on-a-chip systems[J]. ACM Computing Surveys, 2021, 54(5): 1−29

[2] Liu Genggeng, Huang Hongbin, Chen Zhisheng, et al. Design automation for continuous-flow microfluidic biochips: A comprehensive review[J]. Integration, 2022, 82: 48−66

[3] Hu Xu, Chen Zhen, Chen Zhisheng, et al. Architectural synthesis of continuous-flow microfluidic biochips with connection pair optimization[J]. Electronics, 2024, 13(2): 247

[4] Einav S, Gerber D, Bryson P D, et al. Discovery of a hepatitis C target and its pharmacological inhibitors by microfluidic affinity analysis[J]. Nature Biotechnology, 2008, 26(9): 1019−1027

[5] Hong J W, Chen Yan, Anderson W F, et al. Molecular biology on a microfluidic chip[J]. Journal of Physics: Condensed Matter, 2006, 18(18): 691−701

[6] 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

[7] Thorsen T, Maerkl S J, Quake S R. Microfluidic large-scale integration[J]. Science, 2002, 298(5593): 580−584

[8] Huang Xing, Ho T, Chakrabarty K, et al. Timing-driven flow-channel network construction for continuous-flow microfluidic biochips[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2019, 39(6): 1314−1327

[9] Araci I E, Quake S R. Microfluidic very large scale integration (mVLSI) with integrated micromechanical valves[J]. Lab on a Chip, 2012, 12(16): 2803−2806

[10] Fidalgo L M, Maerkl S J. A software-programmable microfluidic device for automated biology[J]. Lab on a Chip, 2011, 11(9): 1612−1619

[11] Lim Y C, Kouzani A Z, Duan W. Lab-on-a-chip: A component view[J]. Microsystem Technologies, 2010, 16: 1995−2015

[12] Tseng T M, Li Mengchu, Freitas D N, et al. Columba 2.0: A co-layout synthesis tool for continuous-flow microfluidic biochips[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2017, 37(8): 1588−1601

[13] Yang Kailin, Yao Hailong, Ho T, et al. AARF: Any-angle routing for flow-based microfluidic biochips[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2018, 37(12): 3042−3055

[14] Wang Qin, Zuo Shiliang, Yao Hailong, et al. Hamming-distance-based valve switching optimization for control-layer multiplexing in flow-based microfluidic biochips[C]//Proc of the 22nd Asia and South Pacific Design Automation Conf. Piscataway, NJ: IEEE, 2017: 524−529

[15] Wang Qin, Xu Yue, Zuo Shiliang, 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

[16] Zhu Ying, Huang Xing, Li Bing, 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

[17] Huang Xing, Cai Huayang, Guo Wenzhong, et al. Control-logic synthesis of fully programmable valve array using reinforcement learning[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2024, 43(1): 277−290

[18] Liang Siyuan, Li Mengchu, Tseng T M, et al. CoMUX: Combinatorial-coding-based high-performance microfluidic control multiplexer design[C/OL]//Proc of the 41st ACM Int Conf on Computer-Aided Design. New York: ACM, 2022 [2024-06-19]. https://dl.acm.org/doi/abs/10.1145/3508352.3549353

[19] Chen Zhisheng, Huang Xing, Guo Wenzhong, et al. Physical synthesis of flow-based microfluidic biochips considering distributed channel storage[C]//Proc of Design, Automation & Test in Europe Conf & Exhibition. Piscataway, NJ: IEEE, 2019: 1525−1530

[20] Hu Kai, Dinh T A, Ho T, 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

[21] Zeng Junjie, Qin Long, Xu Haotian, et al. Exploration approaches in deep reinforcement learning based on intrinsic motivation: A review[J]. Journal of Computer Research and Development, 2023, 60(10): 2359−2382 (in Chinese)
(曾俊杰, 秦龙, 徐浩添, 等, 基于内在动机的深度强化学习探索方法综述[J]. 计算机研究与发展, 2023, 60(10): 2359−2382)

[22] Sutton R S, Barto A G. Reinforcement learning: An introduction[M]// Adaptive Computation and Machine Learning. Cambridge, MA: MIT, 2018

[23] Tang Chuzhe, Wang Zhaoguo, Chen Haibo. Empowering system software with machine learning methods: Challenges, practice, and prospects[J]. Journal of Computer Research and Development, 2023, 60(5): 964−973 (in Chinese)
(唐楚哲, 王肇国, 陈海波. 机器学习方法赋能系统软件: 挑战、实践与展望[J]. 计算机研究与发展, 2023, 60(5): 964−973)

[24] Kober J, Bagnell J A, Peters J. Reinforcement learning in robotics: A survey[J]. The International Journal of Robotics Research, 2013, 32(11): 1238−1274

[25] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540): 529−533

[26] Wang Ziyu, Schaul T, Hessel M, et al. Dueling network architectures for deep reinforcement learning[C]//Proc of the 33rd Int Conf on Machine Learning. Cambridge, MA: MIT, 2016: 1995−2003

[27] Liu Chunfeng, Huang Xing, Li Bing, et al. DCSA: Distributed channel-storage architecture for flow-based microfluidic biochips[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020, 40(1): 115−128

[28] Marcus J S, Anderson W F, Quake S R. Microfluidic single-cell mRNA isolation and analysis[J]. Analytical Chemistry, 2006, 78(9): 3084−3089

[29] Van Hasselt H, Guez A, Silver D. Deep reinforcement learning with double q-learning[C]//Proc of the 30th AAAI Conf on Artificial Intelligence. Palo Alto, CA: AAAI, 2016: 2094−2100

[30] Mnih V, Kavukcuoglu K, Silver D, et al. Playing Atari with deep reinforcement learning[J]. arXiv preprint, arXiv:1312.560, 2013

Cai Huayang, born in 1997. PhD candidate. His main research interest includes design automation of microfluidic biochips.

蔡华洋, 1997 年生. 博士研究生. 主要研究方向为微流控生物芯片的设计自动化.

Huang Xing, born in 1991. PhD, professor, PhD supervisor. His main research interest includes design automation for microfluidic biochips and integrated circuits.

黄　兴, 1991 年生. 博士, 教授, 博士生导师. 主要研究方向为微流控生物芯片和集成电路的设计自动化.

Liu Genggeng, born in 1988. PhD, professor, PhD supervisor. Senior member of CCF. His main research interest includes design automation for microfluidic biochips and integrated circuits.

刘耿耿, 1988 年生. 博士, 教授, 博士生导师. CCF 高级会员. 主要研究方向为微流控生物芯片和集成电路的设计自动化.