目录

[文献一 2](#_Toc145513714)

[文献二 4](#_Toc145513715)

[文献三 6](#_Toc145513716)

[文献四 8](#_Toc145513717)

[文献五 12](#_Toc145513718)

# 文献一

**文章名**：A Capacity-Aware Wash Optimization for Contamination Removal in Programmable Microfluidic Biochip Devices

**作者**：Piyali Datta$, Arpan Chakraborty, and Rajat Kumar Pal

**会议**：2019 32nd International Conference on VLSI Design and 2019 18th International Conference on Embedded Systems

**年份**：2019

**针对问题**：洗涤时间和缓冲液的容量浪费

**问题表述及约束**：清洗液滴和功能液滴的同步问题，导致清洗路径边长（清洗液滴避开正在工作的功能液滴）和清洗液滴无路可走。应有以下约束：

1、静态流体约束：任何两种流体流必须随时保持安全分离，以避免意外混合。

2、动态流体约束：如果发生任何阀门危险，则不允许同时进行两种流体运动。

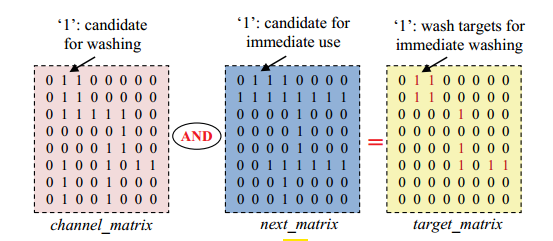
3、阀门约束：PMD 网格上的节点不能同时输入和输出流体

4、容量限制：访问清洗目标的每种清洗流体的容量必须大于或等于清洗该位置所需的容量。

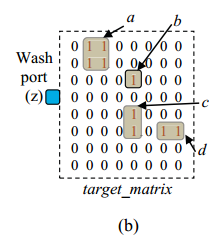
**解决方案**：

1、计算出清洗路线

维护属于子问题Pi的两个矩阵，一个通道矩阵（1表示该节点被污染），一个next矩阵（1表示接下来要用到的节点），两个矩阵中均为1的节点视为子问题Pi清洗的目标，两矩阵相与后得到目标矩阵（1为清洗目标）



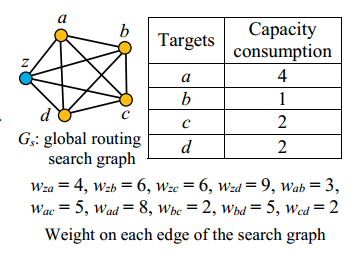
2、最小化总体容量消耗

收集容量消耗的信息：先从1中的目标矩阵提取容量消耗的信息，如下图

将连续的节点分块，每块看作清洗需求为m的一个单位，若液滴到该单位清洗能力小于m则清洗失败，要重新输入清洗液滴

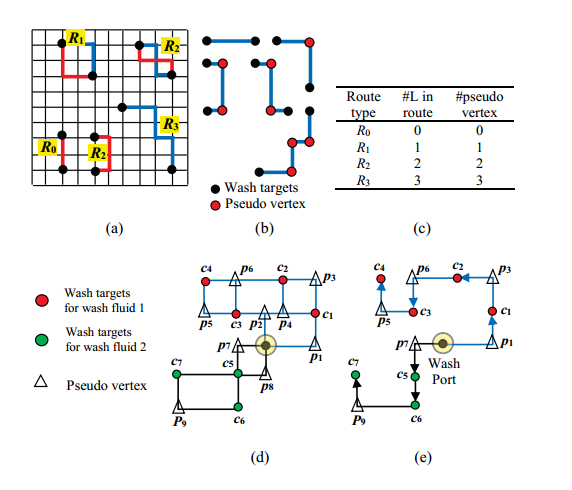
3、最小化总清洗时间

先将Pi的目标矩阵中提取的块和缓冲液源端口以及它们之间的布线可视化一个搜索图，如下所示，abcd为清洗目标，z为缓冲液源端口，将各边以曼哈顿距离为基础赋予权重



再解决CWRO问题（容量损耗布线顺序）：将该问题表述为对于多个清洗缓冲流，搜索图中是否存在一组路径，从缓冲液源端口开始，访问所有每个目标恰好一次，使得每条路径的总容量消耗不超过总容量、清洗缓冲流的洗涤能力和总遍历成本小于一个正整数K？针对该问题，本文提出MILP公式（详见文献）

随后是详细的布线阶段，需要注意清洗缓冲流的转向问题，也就是为了避免缓冲流清洗到了计划外的污染点导致清洗能力下降，影响整个清洗的过程。上述的MILP生成了一组路径，它实际上保存了多条可行路径。合并“L”形的子路径，生成伪顶点，再考虑各边的权重赋为：通过候选清洗路径的微通道单元数，从而得到了一个加权图。再以Floyd-Warshal 的最短路径算法找到成本最小的路径，该路径为清洗路径，如下图：



**该算法结果**：

与 H-Scan（水平扫描）、V-Scan（垂直扫描） 和 NTF（最近目标优先） 相比，它完成所有清洗目标污染物的时间（平均）分别减少了 75%、71.8% 和 43.2%。所需的清洗液流量（平均）分别比上述布线方法少 42.1%、41.4% 和 47.3%。

**该算法缺点**：没有考虑污染物降解。

# 文献二

**文献名**：Block-Flushing: A Block-based Washing Algorithm for Programmable Microfluidic Devices

**作者**：Yu-Huei Lin, Tsung-Yi Ho, Bing Li, Ulf Schlichtmann

**会议**：Design, Automation, and Test in Europe

**年份**：2019

**针对问题**：清洗时间优化

**问题模型及约束**：

输入：PMD 架构；清洗块的位置；缓冲液源端口和缓冲液槽的位置

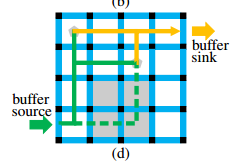
输出：清洗块组合方案；总清洗时间

目标：减少清洗时间；减少清洗作业次数；避免干净的缓冲液流与受污染的缓冲液流交汇

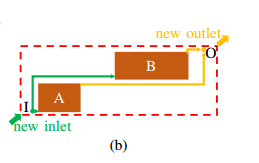
**解决方案**：

核心思想是通过打开相应阀门，合并各区域形成块，以块为清洗单位，缓冲流的路径为源端口到该块的最短路径。

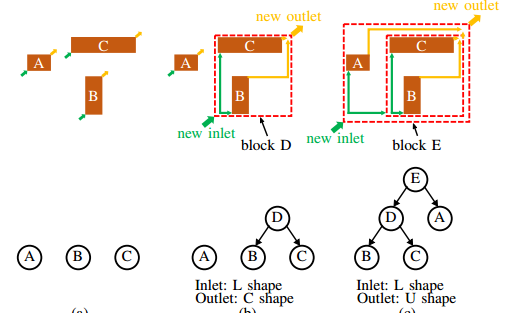
在阐述解决方案前需要合并缓冲流的路径从而提高清洗效率（如果清洗缓冲流连续经过两个污染点的话，会导致清洗第二个污染点低效），如下图所示，也就是分流再汇合。这样做也能避免缓冲流经过不在清洗计划内的节点（如图中虚线路径）



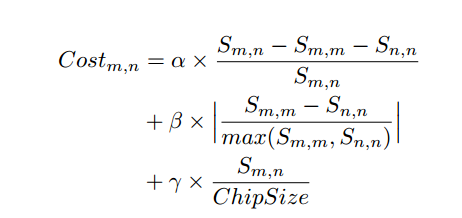
合并成块之后，可以继续递归地把块合并成更大的块（即封装成包，块的入口连接起来，对外体现为一个入口），如下图。递归地进行，直到所有清洗块连接成一颗清洗树



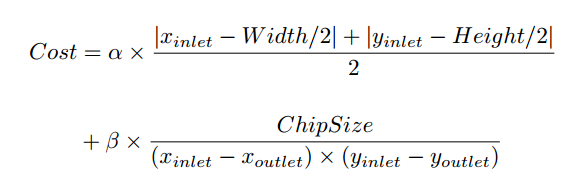
清洗树为满二叉树，叶子节点是清洗块，非叶子节点是覆盖其子树的清洗块，下面是生成清洗树的一个实例



随后，根据优先级构造清洗树对的队列，按该队列顺序地合并两个清洗树，然后删除原来的树，保证整科树中每个清洗块只出现一次。随机地选择两棵清洗树作为清洗对会造成不良的结果。先按x轴方向或者y轴方向组合清洗树对，从而可以得到由清洗树对组成的队列，再按以下成本函数计算清洗树对的成本，从而确定清洗树对队列内各清洗树对的优先级。



依次检查优先级低的对是否可以合并，根据相对位置，检查是否能为“L”、“C”或“U”中的一种情况，这也叫模式布线。若非这三种情况，说明存在障碍，根据以下成本函数，选择成本最低的清洗树先对其清洗。选择合适的模式后，就确定缓冲流的出入口。



迭代地进行上述步骤，最终只剩下一个清洗块

**实验结果**：与顺序清洗和随机合并成块这两种布线方式相比，清洗时间开销更小，在大规模的情况下更加优秀

# 文献三

**文献名**：Design Optimization for Programmable Microfluidic Devices Integrating Contamination Removal and Capacity-Wastage-Aware Washing

**作者**：Piyali Datta , Arpan Chakraborty & Rajat Kumar Pal

**期刊**：IETE Journal of Research

**年份**：2020

**针对问题**：清洗时间优化和容量浪费

**问题模型及约束**：

输入：生物测定图；芯片尺寸；缓冲液源端口和缓冲液槽的位置

目标：开发一种与容量浪费感知清洗方案相结合的流动层综合方案

约束：

1、进行生物测定所需资源阀门的总数在任何时候都不得超过其可用限制

2、所有生物测定操作的执行必须保持其顺序。

3、避免功能液滴和清洗液滴的意外混合。

4、避免阀门危险。

5、每种清洗缓冲液在清洗目标时，必须具有足以清洗该位置的污染物的能力

**解决方案**：

该方案包括两个阶段：

（1）找到最少部署阀门数量、最少污染点的流动层解决方案；

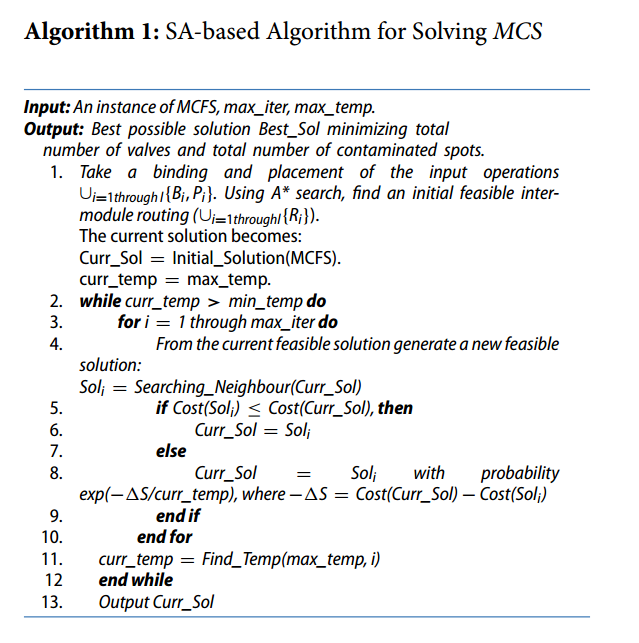
（2）最小化清洗时间并减少容量消耗。

首先要获取清洗子问题Pi的污染点清洗顺序，其次，为了找到一条从清洗缓冲流源端口到目标污染点的详细路线，应用基于A\*搜索的算法。

文章在介绍该方案时，是分成五个环节介绍：流动层综合、目标污染点检测、减少容量浪费、容量浪费感知的全局路径查找、路径质量感知详细布线。

流动层综合：

该问题可视为MCS（最小成本综合）问题。针对该问题，首先要设计高效的模块布局方案，这是一个NP完备问题，用启发式方法来解决该问题。根据输入的生物测定图找到与相关模块绑定的操作，用基于SA的算法布局后，用基于A\*搜索的算法来进行布线，文中给出了三种生成不同新解的情况。基于SA的算法如下：



清洗目标检测：

维护Mch（保存通道利用的详细信息）和Mnext（常规流体操作的下一步的通道部署信息）两个二维数组，这两个数组根据文中给出的三种规则进行更新。更新后，如果节点在两个二维数组中都为“1”，则该节点为目标污染点

减少容量浪费：

核心思想就是如果一个目标污染点附近有相邻的目标污染点，则把它们看成一个整体，最后得到该整体所需的清洗液的量，从而判断是否会发生清洗能力不足导致清洗失败的情况。

容量浪费感知的全局路径查找：

再解决CWRO问题（容量损耗布线顺序）：将该问题表述为对于多个清洗缓冲流，搜索图中是否存在一组路径，从缓冲液源端口开始，访问所有每个目标恰好一次，使得每条路径的总容量消耗不超过总容量、清洗缓冲流的洗涤能力和总遍历成本小于一个正整数K？针对该问题，本文提出MILP公式（详见文献），MILP生成了一组路径，它实际上保存了多条可行路径。

路径质量感知详细布线：

“L”形的布线是最合适的，然而有多种“L”形的节点间布线，MILP中生成了一些可行路径，通过最小化这些路径上的dirty-denial节点来找到最终的清洗路线，由MILP计算出的子问题Pi的可行路径可获得伪顶点集（伪顶点是 PMD 网格上的交汇点），与所有顶点间的布线为边，可得到一个带权图，通过计算dirty-denial节点来获得图的各边的权重，用Floyd-Warshal 的算法从单一来源找到最短路径，以找到每个清洗目标之间成本最小化的路径。

**实验结果**：与 H-Scan（水平扫描）、V-Scan（垂直扫描） 和 NTF（最近目标优先） 相比，它完成所有清洗目标污染物的时间（平均）分别减少了 21.1%、31% 和 32.5%。所需的清洗液流量（平均）也更少。

# 文献四

**文献名**：Contamination-Aware Synthesis for Programmable Microfluidic Devices

**作者**：Hui-Chieh Yu , Yu-Huei Lin, Zhiyang Chen, Bing Li , Xing Huang, Ulf Schlichtmann，Tsung-Yi Ho , and Hailong Yao

**期刊**：IEEE Transactions On Computer-aided Design Of Integrated Circuits And Systems

**年份**：2022

**针对问题**：针对PMD的高层次综合及物理设计以及清洗优化

**创新点：**以块为清洗单位，且高效地合并清洗块，将问题建模为3D模块布局问题

**痛点：**模拟结果对比不够全面

**问题模型及约束**：

输入：由操作及其之间的关联的有向无环图；组件库及其相关信息；

输出：PMD 架构以及芯片尺寸和总计算时间；清洗操作和生化操作的列表；阀门的控制顺序；

目标：减少清洗时间；减少生化操作总时间；

约束：

1、流体约束

2、操作顺序约束

3、操作依赖约束

4、污染约束

5、不可重叠约束

6、不可阻塞约束

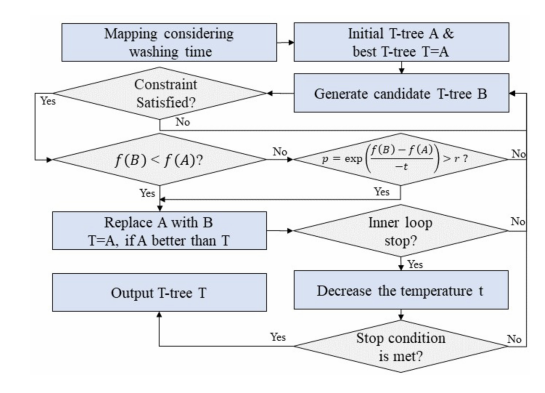
**解决方案**：

相较于2019发表于DATE的工作，本文提出了一种将布局、绑定和调度建模成3D模块布局问题的方法，提出了一种基于T树的3D模块布局方法来进行物理设计和绑定，其用基于模拟退火的方法进行优化，此外还考虑了存储的问题。清洗方面还是沿用了其以块为清洗单位的思想。

定义1：T树

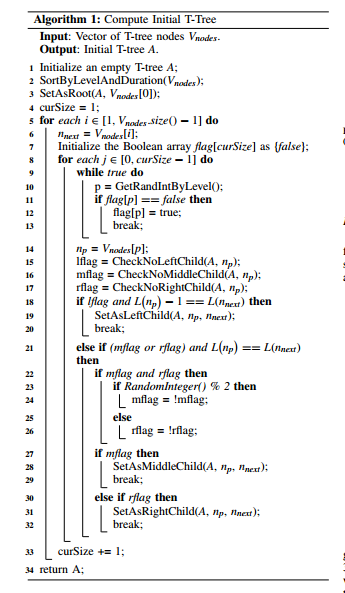
其是一个三叉树，每个节点代表一个3D模块，T树的每条边代表两个模块之间的几何关系。

基于T树的3D模块布局方法和调度方法：



其流程如上图所示，先用文中定义的策略生成初始的满足约束的T树，进行扰动操作，每次扰动后若不符合约束则要重建树，最后通过筛选物理位置、大小和时间得到符合预期的T树，其对应一个布局方案。

T树的初始生成算法：

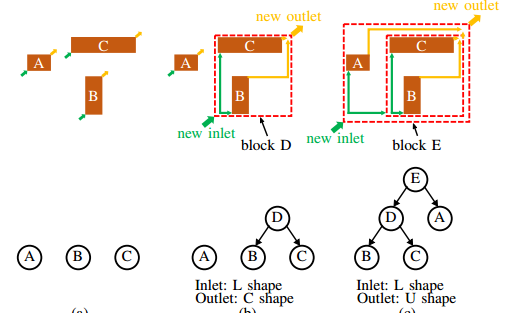


生成初始T树后，用基于模拟退火的算法进行更新，更新的步骤总结为：移动、交换和旋转（移动和交换是对于3D模块节点的，旋转是旋转模块，即X轴方向和Y轴方向交换）。

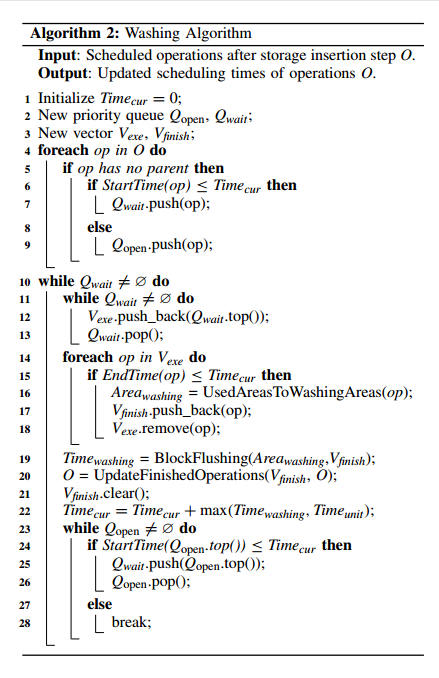
清洗块的核心思想是通过打开相应阀门，合并各区域形成块，这样形成的清洗块需要约束为三种类型，分别为“L”、“C”和“U”型。以块为清洗单位，缓冲流的路径为源端口到该块的最短路径。

合并成块之后，可以继续递归地把块合并成更大的块（即封装成包，块的入口连接起来，对外体现为一个入口），递归地进行，直到所有清洗块连接成一颗清洗树

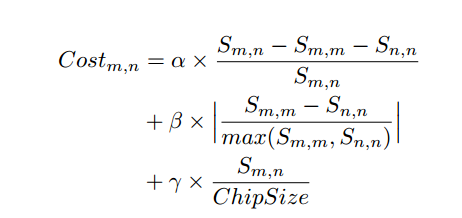
清洗树为满二叉树，叶子节点是清洗块，非叶子节点是覆盖其子树的清洗块，下面是生成清洗树的一个实例



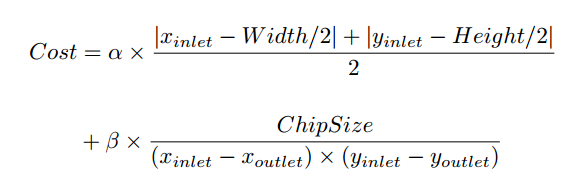
清洗的算法如下：



随后，根据优先级构造清洗树对的队列，按该队列顺序地合并两个清洗树，然后删除原来的树，保证整科树中每个清洗块只出现一次。随机地选择两棵清洗树作为清洗对会造成不良的结果。先按x轴方向或者y轴方向组合清洗树对，从而可以得到由清洗树对组成的队列，再按以下成本函数计算清洗树对的成本，从而确定清洗树对队列内各清洗树对的优先级。



依次检查优先级低的对是否可以合并，根据相对位置，检查是否能为“L”、“C”或“U”中的一种情况。若非这三种情况，说明存在障碍，根据以下成本函数，选择成本最低的清洗树先对其清洗。选择合适的模式后，就确定缓冲流的出入口。



迭代地进行上述步骤，最终只剩下一个清洗块。

**实验结果**：

清洗优化方面，与顺序清洗和随机合并成块这两种布线方式相比，清洗时间开销更小，在大规模的情况下更加优秀。

综合方法方面，与基于贪心算法的ASAP相比，生物检测的时间明显降低，

# 文献五

**文献名**：Mixer-based Washing Methods for Programmable Microfluidic Devices

**作者**：Yifang BAO, Shigeru YAMASHITA, Bing LI,and Tsung-Yi HO

**期刊**：IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences

**年份**：2022

**针对问题**：针对Block-Flushing需长时间清洗和较多的缓冲液以及可能无法将阀门清洗干净

**创新点：**首次提出用混合器清洗

**痛点：**文章的混合器的选用是基于矩形的，复杂形状的混合器没有进行探究

**问题模型及约束**：

输入：一组受污染的阀门；

输出：

目标：确定应用哪些混合器来清洗所有受污染的阀门；

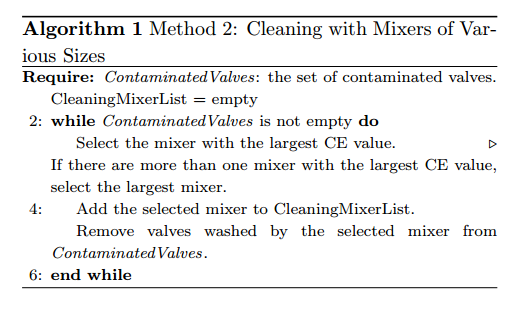
约束：

**解决方案**：

文中提出了两种利用混合器进行清洗的方法。

第一种是用2×2的混合器进行清洗操作，首先定义了四种类型的阀门以便叙述，然后按顺序递增的顺序依次从输入的受污染的阀门中选择对应混合器进行清洗。

第二种方法是出于多次打开/关闭阀门会损耗阀门的考虑，首先定义了一种衡量清洗效率的参数—清洗效率CE，其值为清洗操作清洗了多少阀门比上用了多少阀门，显然该值越大，对阀门损耗越小。对此，文中提出了一种基于贪心算法的启发式算法来选用不同尺寸的混合器进行清洗。算法如下：



**实验结果**：

清洗时间方面，方法一与方法二接近；CE值方面，方法二优于方法一。