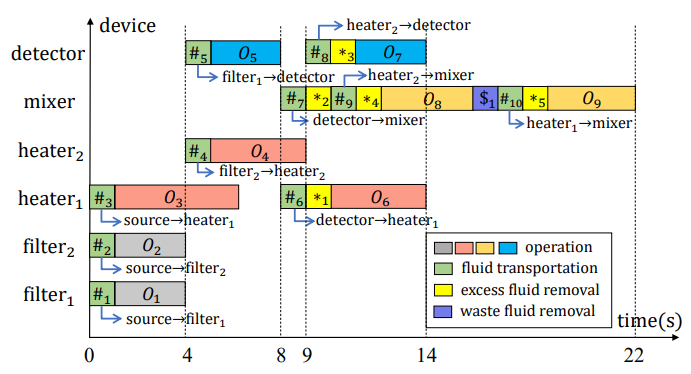
Three-Stage Rapid Physical Design Algorithm for Continuous-Flow Microfluidic Biochips Considering Actual Fluid Manipulations

文章简要介绍了对PathDriver进行了介绍，提出其不能为NP完全问题提供多项式复杂度解。在解决大规模问题时，求解时间可能很长，甚至可能存在无法在可接受的时间范围内完成计算的情况。而改进的PathDriver+尽管进一步考虑了对角线布线的实现和流体端口数量的最小化，但仍然用ILP方法解决物理设计阶段，物理设计算法时间复杂度高的问题尚未得到解决。

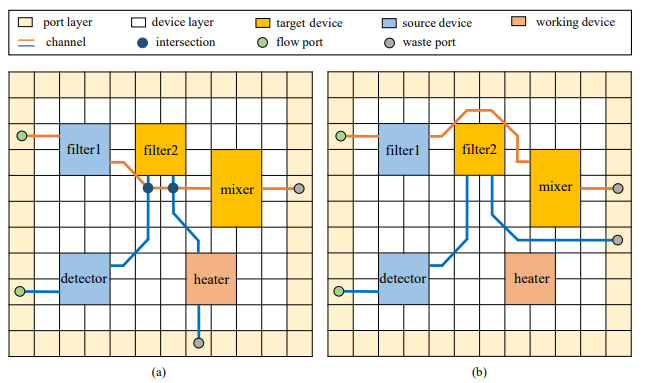
在执行过程中，每个流体输送/去除任务都需要在流路的一端连接一个流口，以提供压力源来驱动流体输送。同时，流路的另一端需要连接到废物口以释放气压。以图2所示的生物测定的结合和调度方案为例。这是一组生物测定和使用的装置库的高级合成阶段的结合和调度方案。*O*1-*O*9代表九种生物测定操作，每一种操作都与特定设备结合以执行。例如，操作*O*1绑定到过滤器1，操作*O*3和*O*6绑定到加热器1，操作*O*8和*O*9绑定到混合器。装订和调度方案共包括16个流体运输/去除任务，包括10个流体运输任务（#1–#10）、5个过量液体去除任务（∗1–∗5）和1个废液去除任务（$1）。这16项流体运输/去除任务需要构建包含流口和废液口的流路。



由于使用从外部压力源注入气压的方法来驱动流体在流道中的输送，因此在流体输送过程中必须避免空气进入目标设备。因此，在流体输送过程中应严格满足设备的容量限制，以便将空气完全推出设备，生物测定只能在去除过量液体后进行。图3显示了混合器的工作过程。首先，将需要混合的绿色液体输入混合器中，然后清除混合器两端多余的液体。接下来，将黄色液体输入混合器进行混合，同时从黄色液体中排出多余的液体。因此，完整的多余液体去除任务必须包括多余液体通道的流路、流口和废液口。当没有后续操作的操作完成时，液体样品和试剂将占据与该操作绑定的设备。此时，需要将液体样品和试剂从设备输出到芯片外部。因此，废物清除任务的流路必须包括流口、废物所在的装置和废物口。

在生物芯片的绑定调度方案中，提供了流体运输任务中设备之间的连接关系以及超额液体和废液的位置，但液体运输/去除任务所绑定的端口尚未确定，端口与设备之间的连接关系也尚未确定。在物理设计阶段，需要获取流体传输/移除任务与端口之间的绑定关系，以及端口与器件之间的连接关系，以实现最佳的放置和布线方案。当只考虑设备之间的连接关系而不考虑端口和设备之间的连接关系时，很容易陷入局部最优解中。因此，在解决物理 设计问题时，需要分析所有流体传输/移除任务，将它们与端口绑定，并确定端口的数量以及端口与设备之间的连接关系。然后可以根据这些信息进行布局和布线设计。

并行约束任务：当并行执行8多个流体输送/去除任务时，流路中的流口、废液口、源和目标设备不能共享，并且它们的流道不能有交点。同时，由于执行操作而处于关闭状态的设备不能出现在平行流体输送/去除任务的流路上。在生物芯片的物理设计阶段，应严格遵循并行任务约束，以确保生物测定的结合和调度方案的实施。



考虑实际流体操作的CFMB的物理设计问题可以表述如下：

输入：（1）考虑实际流体操作的绑定和调度方案，包括所有操作的开始和结束时间以及流体运输/移除任务的开始和结束时间。（2）根据实际流体操作构建一组流路，包括使用的成分信息以及过量液体和废液的位置信息。

输出：流路规划方案，包括：（1）每个流路中使用的端口和设备的放置位置。（2）流道的布线方案。

目标：最小化（1）流体端口数量。（2）流道的总长度。（3）流道交点的数量。

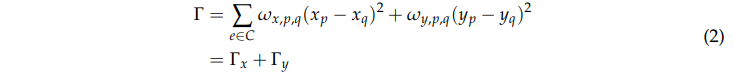
文章提出了一种三阶段快速物理设计算法。该算法包括端口驱动的预处理阶段、力导向的二次放置阶段和基于协商的路由阶段。

在端口驱动的预处理阶段，提出一种端口驱动的预处理算法，在端口和设备之间生成连接矩阵，以减少引入的端口数量。使用最大并行传输时隙优先级策略来确定所有流路中连接的设备和端口，优先分析具有最多并行流体运输/去除任务的时隙，并将端口分配给该时隙中的每个流体运输/去除任务。然后在重复利用端口与创建新端口两个方案中选择代价最小的方案执行。第一种解决方案解决了平行运输流路之间的需求，但代价是增加了端口数量。第二种解决方案仅增加现有端口和设备之间的连接数，而不添加新端口。但是，第二种方案可能很容易增加端口和设备之间的最大连接数，其中端口的最大连接数是指所有端口中连接次数最多的端口连接的设备数，设备的最大连接数是指所有设备中连接数最多的设备与其他设备和端口的连接数。当端口和设备之间的最大连接数过大时，端口和设备周围的路由资源可能会紧张，严重时，可能会因端口和设备周围的路由资源不足而导致路由失败。因此损失函数综合了两个方案的优缺点：



在力导向二次放置阶段，将放置问题建模为二次代价函数的极值问题，这在数学上显著减少了搜索空间，并显著缩短了算法的运行时间。

在二次布局中，连接矩阵*C*用于表示模块之间的连接关系，其中***C*中的每个非零元素都表示一条边*e***，行和列分别表示两个模块。端口和设备之间的连接矩阵*C*中的边*e*=（*p*，*q*）由一对设备或端口*p，q*表示。设备或端口的位置（例如*p*）由坐标（*xp*，*yp*）表示，二次成本函数*Γ*是*e*∈*C*所有边的欧几里得长度平方和。因此，*Γ*的表示形式如下：

其中*Γ*取决于连接点的位置，本文假设连接点位于它所代表的端口或设备的中心。成本函数可分为*x*方向和*y*方向，*Γx*表示*x*方向的开销，而*ωx,p,q*，*ωy,p,q*分别表示*x*方向和*y*方向的连接权重。在本文提出的问题模型中，在*x*方向和*y*方向建立流道的成本相同，因此*ωx,p,q*=*ωy,p,q*=1。*Γx*的计算方法主要后面讨论，*Γy*与它类似。

在布局过程中，设备和端口分为可移动模块（总数*M*）和固定模块（总数*F*）。由于端口和器件的放置区域不同，端口位于芯片的边缘，而器件位于芯片的中心。因此，本文将端口放置与设备放置分开。在进行端口配置时，端口被视为可移动模块，而设备是固定模块。在进行设备布局时，设备被视为可移动模块，而端口是固定模块。由于只需要确定可移动模块的位置，因此*M*个可移动模块的*x*坐标可以用向量*x*表示：

根据（3），（2）中的二次成本函数*Γx*可以表示为二次矩阵：

其中，矩阵*CM*表示可移动模块之间的连接，而矢量*dx*表示可移动模块和固定模块之间的连接。*CM*的维度为*M*×*M*，在第*i*行、第*j*列中具有元素*ci*，*j*。*dx*的维度为*M*，在第*i*行中具有条目*di*。**确定*CM*和*****dx*的方法如下**。例如，在端口和设备之间的连接矩阵*C*中取一条任意边*e*=（*i，j*），**如果模块*i*和*j*都是可移动的**，则将*ωx,p,q*添加到对角矩阵项*ci，i*和*cj，j*中，并从非对角线项*c**i，j*和*cj，i*中减去*ωx,i,j*。**如果一个组件是固定的**，例如组件*j*，则*ωx,i,j*被添加到*ci，i*，并且从向量项*di*中减去*xj*×*ωx,i,j*。**如果两个模块都是固定的**，则*CM*和*dx*不会更改。

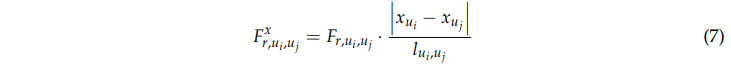
*Γx*的最小值可以通过找到导数等于零的点来获得:

在放置过程中，获得的模块坐标与流道的最小总长度将严重重叠。力导向二次放置算法用于使用额外的排斥力来分散芯片上的模块，以消除重叠。**力导向算法基于物理学中的万有引力和排斥原理，模拟了空间中节点之间的交互，使图中具有连接关系的模块聚集在一起，排斥力的存在保证了模块之间不会重叠**。

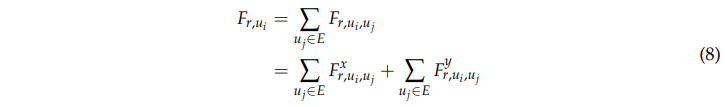
由于设备和端口的放置区域不同，因此设备和端口之间不会重叠。因此，所有相似的模块之间，例如设备和设备之间以及端口与端口之间都存在排斥力。不同模块之间（例如设备和端口之间）没有排斥力。任意两个相同类型的模块之间的排斥力可以表示为：

其中模块*ui*，*uj*∈*P*（端口集）或*ui*，*uj*∈*D*（设备集），*l*是模块*ui*和模块 uj 之间的欧几里得距离，计算公式为 *lui,uj*=根号[（*xui*−*xuj*）^2+（*yui*−*yuj*）^2]，排斥力的方向*Fr,ui,uj*与连接两个模块的线相反，*Kr*是力导向算法中的系数;在文中，*Kr*=1。

由于排斥力*Fr,ui,uj*的方向与连接模块*ui*和模块*uj*的线相反，因此力可以在*x*方向*y*方向上分成分量。以*x*方向为例，排斥力在*x*方向的大小*F xr,ui,uj*的计算公式为：



根据（6）和（7），模块*i*接收到的所有排斥力*Fr,ui*都可以表示为:

其中*E*=*D*（当模块*ui*表示设备时）和*E*=*P*（当模块*ui*表示端口时）。

由于初始位置有明显的重叠，因此在首次计算排斥力时获得了较大的排斥力。如果根据排斥力的大小直接移动模块，可能会导致移动距离过大，降低最终布局方案的质量。因此，在移动模块时，移动距离应乘以折减系数*τ*。以*x*方向为例，模型*ui*的移动距离*△x*表示为：

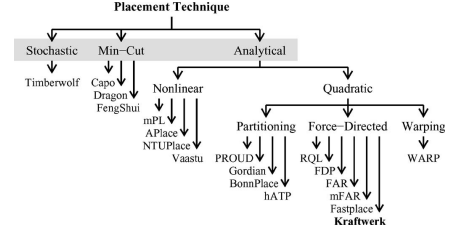


根据（9），模块移动后的新位置表示为：



在布局迭代中，会添加布局成本变化*△*和参数*ε*来决定是否终止迭代。放置成本变化*△*是从当前放置迭代的成本*Γ*中减去上一次放置迭代的*Γpre*放置成本而获得的绝对值。参数*ε*是预先设置的。仅当*△*≤*ε*时，才会输出布局结果。

在文章使用的二次力导向布局算法原文中，作者首先对现有的布局算法进行了归类，如图3所示，现有的布局技术包括随机、最小切割以及解析式，其都有几种比较典型的算法。但是随机和解析式中的非线性算法需要使用大量的CPU资源，因此力导向算法更适合布局。



然后介绍了几种网络模型，从理论说明了算法Kraftwerk2采用的Bound2Bound网络模型以及其有效性。其次列出布局采用的二次成本函数（同上述的布局算法一致）和其能够最小化网络线长的理论依据。尽管通过二次成本函数能够得到最初解，但是模块的重叠程度非常高，因为都聚集在了某一块区域当中，因此引入了移动力、保持力、净力（也就是最初解引脚连接保持线长最短的力）。在移动力中提出了供需系统D，能够解决某区域密度过大的问题，从而减少重叠程度，然后介绍了保持力，它是净力的反作用力，初始时能够与净力抵消，保持线长优化，而在移动力加入后，为了防止过度解决重叠而忽略线长的优化，所以在模块移动过程中添加了保持力，保持力在初始阶段就已经确定，它不依赖于*x，y*，三种力相互作用移动模块，当合力为0时得到一次迭代后的布局解。

在基于协商的路由阶段，提出了一种基于启发式协商的路由算法和一种优先构建并行执行的流通道策略，以减少算法的运行时间，同时保证路由方案中的交叉数接近最优解。