DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics: Robotics, Cognition, Intelligence

Routing Strategy for Multi-Layer Microfluidic Devices Using ILP

Minxing Wang

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics: Robotics, Cognition, Intelligence

Routing Strategy for Multi-Layer Microfluidic Devices Using ILP

Routing-Strategie für mehrschichtige Mikrofluidikchips mit ganzzahliger linearer Programmierung

Author:	Minxing Wang
Supervisor:	Prof. Dr.-Ing. Ulf Schlichtmann
Advisor:	Yushen Zhang
Submission Date:	2023.11.10

I confirm that this master's thesis in informatics: robotics, cognition, intelligence is my own work and I have documented all sources and material used.

Munich, 2023.11.10 Minxing Wang

Acknowledgments

I would like to express my sincere gratitude to all those who have supported me throughout my journey to attain my Master's degree. I am deeply thankful for the unwavering encouragement, guidance, and assistance I have received from my professors, advisors, and mentors. Their expertise and commitment to education have been invaluable in shaping my academic growth.

I am also grateful for the camaraderie and collaboration I have experienced with my fellow students. Together, we have shared knowledge, faced challenges, and celebrated successes, creating a rich and inspiring learning environment.

Furthermore, I extend my appreciation to my family and friends for their unwavering support, patience, and understanding during this demanding academic endeavor. Their love and encouragement have been my pillars of strength.

To all of you, I extend my heartfelt thanks. Your contributions have played a crucial role in my academic and personal development, and I am grateful beyond words for your belief in me and my aspirations.

Abstract

Nowadays, most microfluidic chips are manufactured by soft lithography technology. However, with the development of 3D printing technology and the emergence of new materials, microfluidic chips can also be produced by 3D printing. Compared with microfluidic chips produced by soft lithography, 3D-printed microfluidic chips allow for the rapid prototyping and production of complex, multi-layered microfluidic architectures, which would be laborious and time-consuming to create using soft lithography. In the future, 3D-printed microfluidic chips will be used in a broader range of fields.

To simplify the 3D-printed chip design process, this thesis designs a program that can generate layouts of channels automatically if given the chip's size parameters, the placement information of modules inside the chip, and the connection relationship between modules. In addition, users can specify the pressure difference between the two modules, and the program can design the layouts of channels according to the required pressure loss. Besides that, to ensure the response speed of the program, the thesis uses the A* algorithm, scaling, and channel merging to avoid introducing too many parameters that need to be calculated.

After testing, the program can generate placement information of channels for a three-layer microfluidic chip with six modules within 1 second. In the future, the accuracy and application scope of the program's output results will be further improved.

Contents

1 Introduction

1.1 Research background

Microfluidic chips are devices that incorporate micro-scale channels and chambers, designed to handle and analyze fluids in extremely small volumes. The core principle of these chips is the manipulation of fluid behavior at a micro-scale through tiny fluidic channels. Microfluidic technology allows experiments to be conducted in a minuscule space, which saves reagents, reduces costs, enhances reaction speeds, and improves efficiency. Commonly used in fields like chemical analysis, biological assays, and pharmaceutical development, microfluidic chips enable precise control and manipulation of minute samples such as cells, DNA molecules, or chemical reagents, providing a high level of automation and integration for experiments. By precisely controlling the flow, temperature, and chemical environment of fluids, microfluidic chips can simulate and analyze complex biochemical processes. For example, microfluidic chips can produce chemicals because the velocity and the property of reaction fluid can be precisely controlled. With the help of microfluidic chips, reactions will be carried out on a small scale and not require hands-on, saving raw materials and guaranteeing safety. For instance, a microfluidic chip can synthesize a common pharmaceutical, aspirin. The chip's channels are designed to mix acetic anhydride with salicylic acid, and an acid catalyzes the reaction within a controlled temperature environment. Due to the microscale environment, the reaction is highly efficient, and the aspirin produced can be collected continuously from the chip output. [1].

Currently, the majority of microfluidic chips are fabricated using soft lithography techniques. However, with the development of 3D printing technology and a more broader range of available printing materials, microfluidic chips can also be produced by 3D printing. Currently, the most commonly used method for 3D printing microfluidic chips is stereolithography(SLA). This is a widely used 3D printing technology for microfluidics, which uses a laser to cure liquid resin into solid plastic. SLA can produce high-resolution features suitable for the tiny channels and structures often required in microfluidic devices [2] [3].

3D printing technology stands to broaden the application of microfluidic chips beyond the reach of traditional photolithography due to several compelling reasons. Firstly, 3D printing allows for rapid prototyping, enabling researchers and developers to iterate design modifications swiftly and cost-effectively. This agility in design and testing significantly accelerates the development process. Secondly, 3D printing is not constrained by the layer-by-layer fabrication method inherent to photolithography, allowing for more complex, three-dimensional structures that can integrate multiple functions within a single chip. This capability can

lead to more sophisticated and multifaceted microfluidic devices. Thirdly, the materials available for 3D printing are diversifying, offering a range of properties such as flexibility, transparency, and chemical resistance that can be tailored to specific applications. Moreover, the accessibility of 3D printers increases the democratization of microfluidic chip fabrication, making it available to a broader audience and potentially spurring innovation. Lastly, the reduction in cost and time, coupled with the increase in design complexity, positions 3D-printed microfluidic chips as a versatile and attractive option for various applications in medical diagnostics, environmental monitoring, and educational purposes, among others.

Specifically, 3D-printed microfluidic chips have the following three adavantages compared to those produced by soft lithography.

First, 3D-printed microfluidic chips have reliable connection ports. Sometimes, it is necessary to connect the microfluidic chip to other devices. For PDMS-based chips, the channels can be sealed to chips by inserting holes or using adhesive. However, this kind of channel intervention may lead to leakage or back pressure due to the lack of reliable connection protocols [4]. 3D-printed microfluidic chips can solve this problem by implementing connection mechanisms such as threaded porters that fit commercial finger-tight adapters, which soft lithography cannot achieve [5].

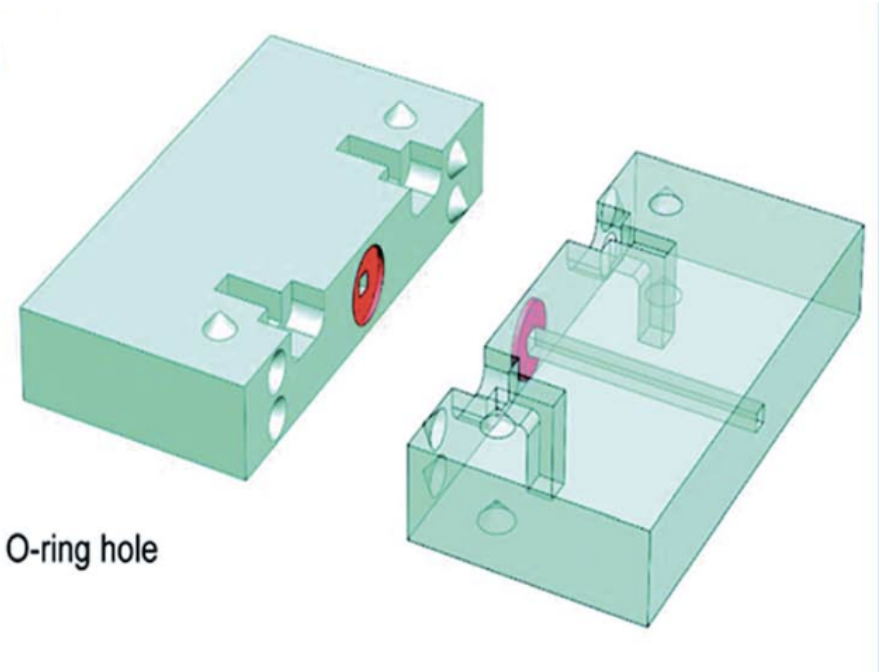


Figure 1.1: Example of connection - O-ring hole [6]

Second, 3D-printed microfluidic chips enable modular installation and disassembly of detectors and sensors. Electrochemical detection is widely used in microfluidic chips for

molecule detection. Nowadays, there are many PDMS-based microfluidic chips integrating electrodes inside. However, this kind of chip is disposable because, once the electrode inside fouls, the chip cannot work. In contrast, if people use 3D-printed microfluidic chips, they can assemble and disassemble the electrodes through threads, clean them, and reuse them [6].

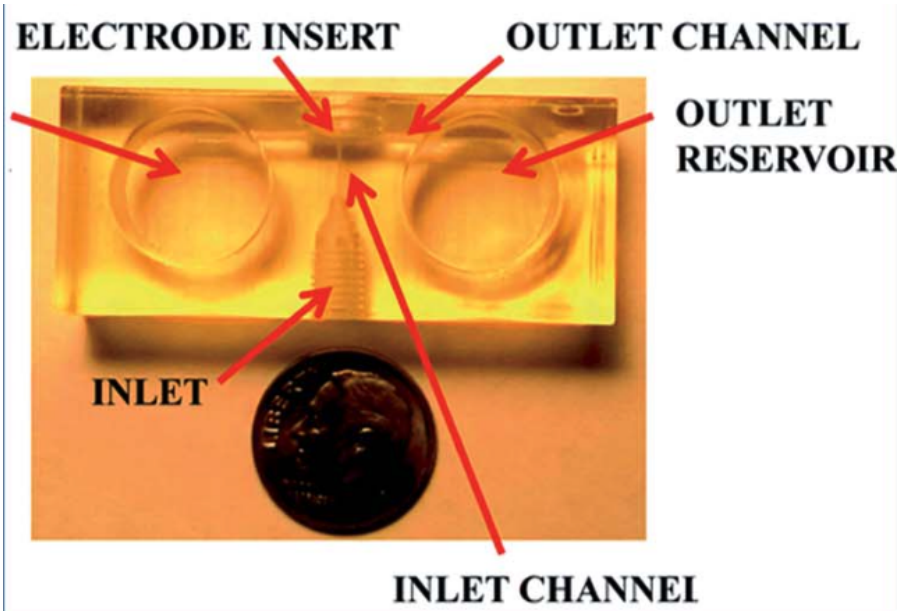


Figure 1.2: A 3D-printed microfluidic device with wall-jet electrochemical configuration. [6]

Last, 3D-printed microfluidic chips support a robust on-chip cell integration. At present, on-chip cell culture is a trendy technology. By controlling the flow of liquid, people can not only provide cells with a continuous supply of nutrients but also take away the waste produced by the cells from the culture environment. In addition, by controlling the speed of the fluid, people can also simulate the biological forces received by cells in the natural environment. Although there are already many PDMS-based microfluidic chips that can serve as a cell culture, 3D-printed microfluidic chips serving as a cell culture is more advantageous. For example, 3D-printed microfluidic chips can implement interface modularization, which is difficult for PDMS-based microfluidic chips to achieve. People can insert the cell culture dish into the chip as a module. If the culture dish is contaminated, people only need to replace the cell culture dish module instead of the entire equipment [6].

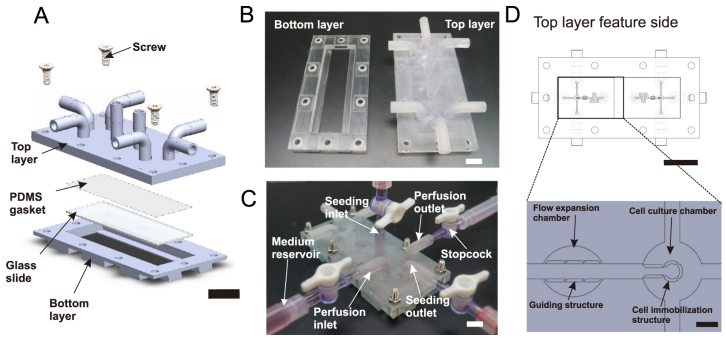


Figure 1.3: Design and instance of the 3D printed microfluidic spheroid culture system. [7]

Since 3D-printed microfluidic chips have many advantages over those manufactured by soft lithography technology, more and more industries will apply 3D-printed microfluidic chips. In this thesis, we mainly focus on the assisted design tool for 3D-printed microfluidic chips, aiming to make the design process of 3D-printed microfluidic chips less time-consuming and labor-intensive.

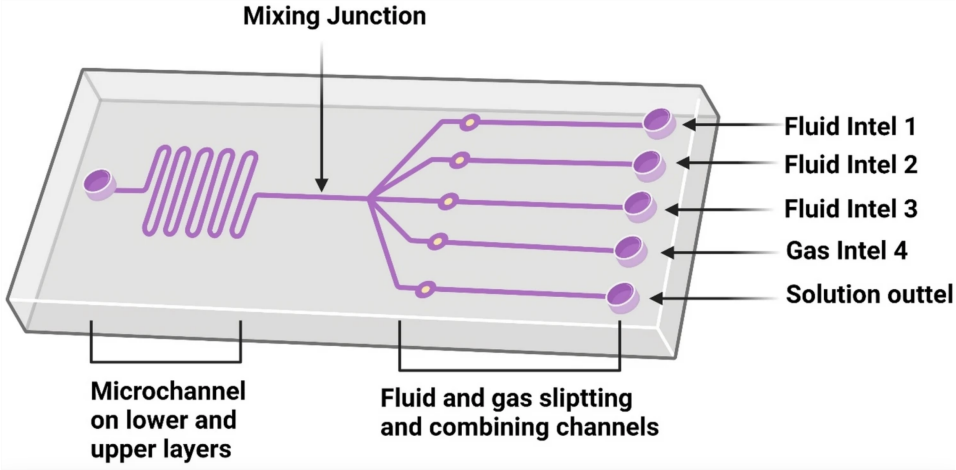


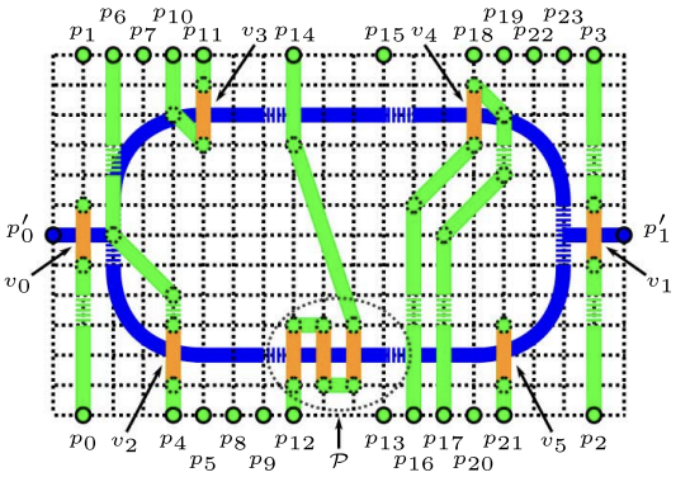
Figure 1.4: Example of microfluidic chip implementing simple mixing function [8]

1.2 Research status

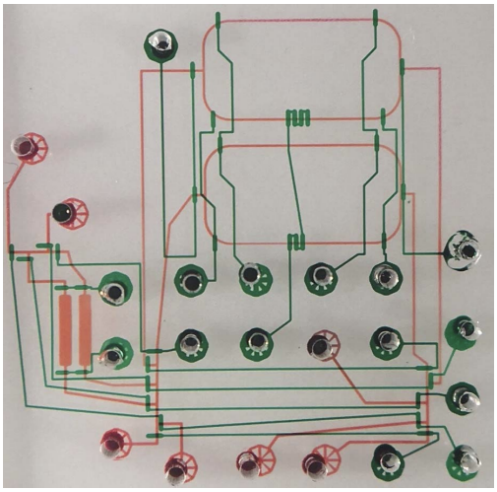
Currently, there are already two famous assisted design tools for 2D microfluidic chips, which can help us automatically generate the layout of modules and channels inside the chip. These two tools are first introduced below.

In 2017, the first automated design tool, which can be seamlessly connected with the microfluidic chip manufacturing process, was invented, called Columba. Columba constructs a module model library to simulate microfluidic elements involved in interactions precisely

and uses linear constraint programming to construct the overall layout of each module on the chip and the channels connecting each module with the help of Gurobi. Regarding the working process of this software, Columba takes a plain-text netlist as input, and its output is compatible with AutoCAD, which means the output can be directly used for mask fabrication [9].



(a) Model example output by Columba 2.0.

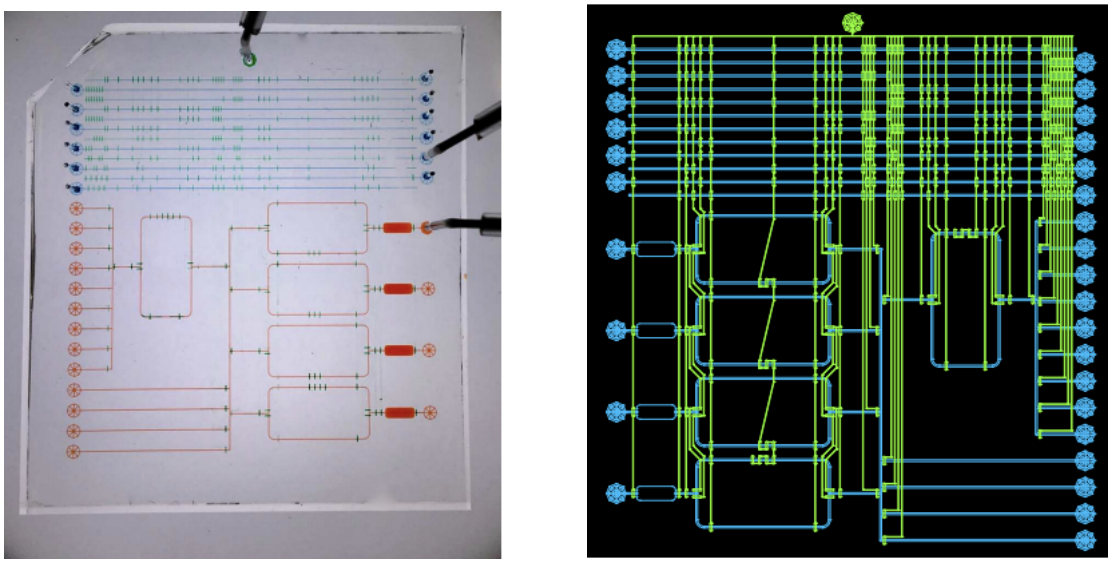


(b) Fabricated chips based on the output model output instance.

Figure 1.5: Output of Columba 2.0 [9].

Furthermore, in 2018, to solve the excessive computational burden caused by large-size chip design, a new architectural framework, and a straight channel routing discipline were applied to Columba. Besides that, the new version of Columba also synthesizes multiplexers for efficient and reconfigurable valve control. With the help of the above methods, the new version of Columba can construct a chip layout design containing 200 modules within 3

minutes [10].



(a) Model example output by Columba S.

(b) Fabricated chips based on the output model output instance.

Figure 1.6: Output of Columba S [10].

As for assisted design tools for 3D-printed microfluidic chips, there are two popular: Flui3d and a program that can automatically design microfluidic chip layouts based on deep reinforcement learning. Through Flui3d, we can drag modules and channels in the visual interface to design the chip, which is intuitive. Besides that, Flui3d can generate an STL format file of the designed chip, which can be directly input into a 3D printer for production [11]. And about the program, users enter the selected modules and their corresponding size parameters into the program. Then, the program can output a chip design plan that includes the placement information about modules and channels [12].

We can notice that neither of the two tools mentioned above has the ability to automatically design a complete 3D-printed chip that has detailed placement information about both modules and channels if users specify the pressure loss between two modules. Existing methodologies for channel layout design that contain constraints on pressure loss are predominantly manual, leading to a high propensity for errors and significant demands on time. To address the current deficit in design automation tools, which lack the capability to determine the precise placement of channels within microfluidic chips based on the interconnectivity of modules and the requirement for pressure loss, this thesis proposes the development of an innovative program.

This thesis endeavor seeks to bridge this gap by introducing a software solution specifically engineered to autonomously generate channel configurations for 3D-printed microfluidic

chips that meet the requirement for pressure loss between modules by users. The program is designed to accept as input the spatial placement of modules, their interconnecting relationships, and desired pressure loss between modules and subsequently outputs the optimal arrangement of channels. This work aims to significantly enhance the efficiency and accuracy of microfluidic chip design.

1.3 Research significance

In this thesis, a program is designed that uses both linear constraint programming and nonlinear constraint programming with the help of Gurobi, a large-scale mathematical programming optimizer, to generate channel layouts between specified input and output ports automatically according to the connection relationship of the module pins. In addition, some optimizations are made to speed up the program, such as using a modified A* algorithm to estimate the necessary number of turns in the channel, which can help reduce the number of parameters in the model.

With the help of this program, as long as the placement information of each module on the chip and the connection relationship between modules are given, the layouts of the channels in the chip can be calculated quickly. After testing, this program can complete the channel layout design of a three-layer chip containing six components within one second. Besides that, if the pressure and the steady flow velocity on the inlet side are specified, and a request on the flow pressure at the outlet is made, the program can also generate a path that meets the pressure constraints by calculating with the Navier-Stokes equation.

In conclusion, this program can automatically generate the connection path data of each module pin in the chip quickly and easily, which can be used for 3D printing, saving time and labor costs.

2 Constraint Programming

Before we discuss the implementation details of the program, first introduce the basic knowledge about constraint programming because this is the basic idea of solving the problem in our program.

2.1 Definition of constraint programming

The working principle of constraint programming is that given a set of variables and possible values for each variable, the constraint programming can reduce the set of values every variable can take according to given constraints and then find a viable solution [13].

In summary, a standard constraint programming problem consists of three parts:

- A set of variables X = {x_1, x_2, ..., x_n}
- A set of domains D = {d_1, d_2, ..., d_n}
- A set of constraints C = {c_1, c_2, ..., c_m}

2.2 Types of constraint programming

According to the exponent of the variable in the constraint, people divide constraint programming into two categories: linear constraint programming and non-linear constraint programming [14].

Linear constraint programming refers to the situation in which the exponents of variables in constraints all equal one. See Equation 2.1.

And non-linear constraint programming refers to a situation in which some variables' exponents exist that do not equal one. See Equation 2.2.

$$Ax + b = 0 \tag{2.1}$$

$$Ax^2 + bx + c = 0 \tag{2.2}$$

2.3 Methods for solving constraint programming problems

Currently, there are three main methods to solve the constraint satisfaction problem: backtracking search, local search, and dynamic programming.

2.3.1 Backtracking search

Backtracking search is a primary method to solve constraint satisfaction problems, and its essence is a depth-first search algorithm. The backtracking search consists of three steps:

- Start with an empty assignment.
- Assign a possible value to a variable at each step.
- Undo the last assignment operation and continue exploring when it is impossible to achieve a solution in the current situation.

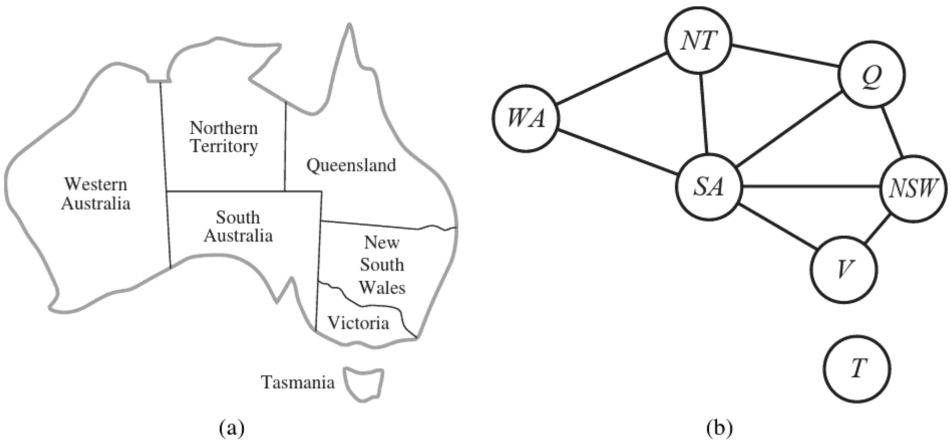


Figure 2.1: An example solved by backtracking search. (a) The map of Australia. (b) The connection relationship between Australia's different regions. The goal is to color every region and ensure that neighboring regions have different colors. [15]

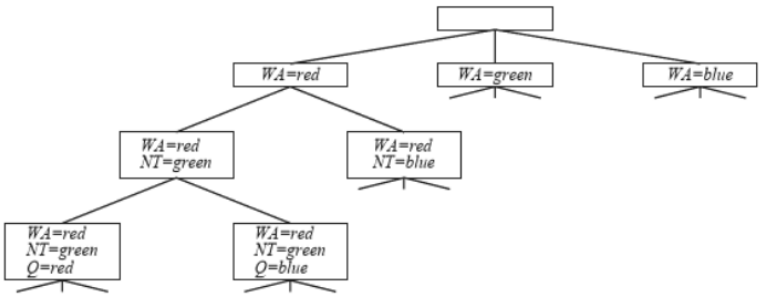


Figure 2.2: Part of the search tree for the coloring problem. [15]

2.3.2 Local search

Local search algorithm solves constraint satisfaction problems effectively through a complete state formulation. First, the algorithm will assign a possible value within the value domain to each variable, during which the assignment is likely to conflict with certain constraints. Second, this algorithm will change the value of one variable at one time until there are no conflicts between assignments and constraints [15] [16].

The min-conflicts algorithm is a popular algorithm that uses local search to solve constraint satisfaction problems. The underlying principle of this algorithm is assigning values that cause minimal conflicts with constraints to the variable each time [15].

The following Figure 2.3 introduces a classic problem - 8-queens problem solved by the min-conflicts constraints. The 8-queens problem is to place the 8-queens in appropriate positions on the chess board so that they cannot attack each other.

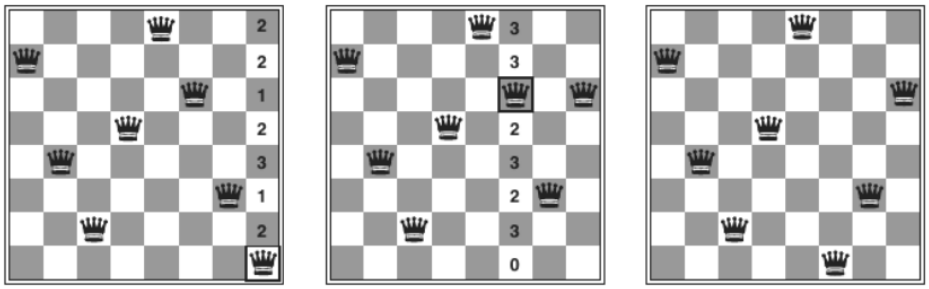


Figure 2.3: 8-queens problem. The numbers in the chessboard grid refer to the number of constraints conflicts that occur when the queen of the current column moves to each position. And the algorithm tends to move the queen to the grid with the least number of constraints conflicts. [15]

After many experiments, it has been proven that the min-conflicts algorithm can effectively solve many CSP problems. For example, the min-conflicts algorithm has been used to schedule observations for the Hubble Space Telescope, reducing the scheduling time from 3 weeks to 10 minutes [15].

2.3.3 Dynamic programming

Dynamic programming is a powerful technique used to solve constraint satisfaction problems efficiently. In computer science and optimization, constraint satisfaction problems involve finding a solution that meets a set of constraints or conditions. Dynamic programming offers a practical approach to tackling these problems by breaking them down into smaller subproblems and solving them systematically and organized [17].

The core idea of dynamic programming involves solving complex problems by dividing them into more straightforward, manageable subproblems. Each subproblem is decrypted only once, and its solution is stored in a table, ensuring that overlapping subproblems are addressed efficiently. By using this approach, dynamic programming significantly reduces redundant calculations and computational effort, resulting in a more optimized and quicker resolution of the overall problem.

In constraint satisfaction problems, dynamic programming helps find an optimal solution that satisfies given constraints. It achieves this by iteratively solving subproblems and utilizing the keys to build the answer for the entire problem. This approach ensures that regulations are met and that the resulting solution is the most efficient and optimal within the given conditions.

By employing dynamic programming, the computational complexity of constraint satisfaction problems is substantially reduced, making it a fundamental technique for solving a wide range of real-world optimization challenges. Its applications extend across various domains, including artificial intelligence, operations research, economics, and many more, providing an invaluable tool for addressing complex problems and enhancing decision-making processes.

3 Mathematical modeling of microfluidic chips

As can be seen chapter 2, constraint programming is expressed by mathematical expressions. Therefore, before introducing how the program establishes constraints on modules and channels, we first need to introduce how to mathematically model chips, modules, and channels, that is, express their boundaries in mathematical form.

In this chapter, we will first introduce the mathematical modeling of chips, modules, and channels. It should be noted that in the mathematical modeling of microfluidic chips, certain idealizations are employed for simplicity. For simplicity in mathematical modeling, channels are typically represented as circular pipes, despite often having square cross-sections in actual microfluidic chips. This approximation simplifies the computational process and typically yields results that do not significantly deviate from actual conditions. And chips and modules are modeled as rectangles. In addition, we also suppose that the flow in channels is laminar because microfluidic flow usually has a low Reynolds number.

Then, we will introduce the fundamental constraints on modules and channels.

3.1 Chip modeling

Chips consist of modules and channels, and modules contain pins. Channels connect different modules via pins to work together to complete tasks. The modeling of modules, pins, and channels will be introduced below.

First, the coordinate's origin and positive direction should be determined before modeling. As shown in Figure 3.1, the coordinate system's origin coincides with the chip's upper left corner endpoint. The side representing the width of the chip is parallel to the X-axis, and the side representing the height is parallel to the Y-axis. In addition, the coordinate system used in this thesis has the positive direction of the X-axis on the right and the positive direction of the Y-axis on the bottom.

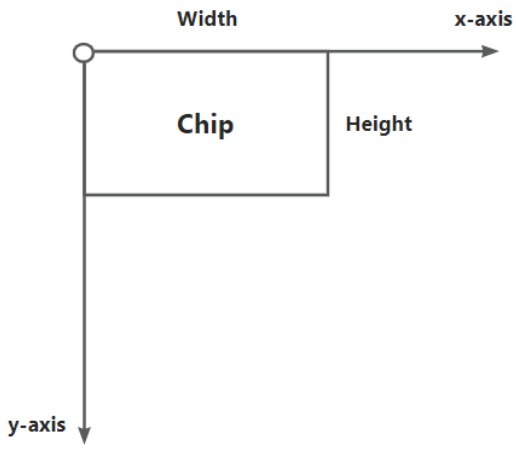


Figure 3.1: The coordinate used in this thesis.

3.1.1 Module modeling

Modules include chambers, mixers, filters, and any other functional components inside a chip. The module of the chip has four main parameters: the coordinates of the upper left corner vertex, the module's length and width, and the module's direction. And after determining the coordinates of the module's upper left corner endpoint and the module's length and width, the module's upper, lower, left, and right boundary values can be determined based on the module's direction.

According to the module's direction, the discussion about modules' boundaries can be divided into two situations: 0 degree and 180 degrees are one situation, 90 degrees and 270 degrees are the other situation. Figure 3.2 and Equation 3.1 introduce the modules upper, lower, left, and right boundaries when the module's orientation is 0 degree or 180 degrees. Figure 3.3 and Equation 3.2 introduce the values of the module's boundaries when module's orientation is 90 degrees or 270 degrees. Among them, x and y refer to the horizontal and vertical coordinates of the upper left corner endpoint of the module.

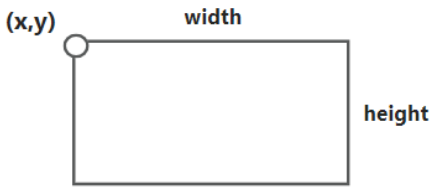


Figure 3.2: The state of module when the orientation is 0 degree or 180 degrees.

$$\begin{cases} left = x \\ right = x + width \\ up = y \\ down = y + height \end{cases} \tag{3.1}$$

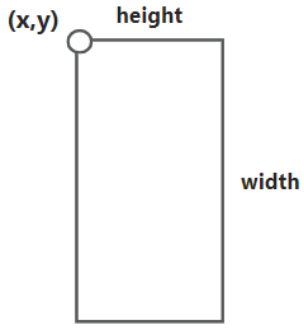


Figure 3.3: The state of module when the orientation is 90 degrees or 270 degrees.

$$\begin{cases} left = x \\ right = x + height \\ up = y \\ down = y + width \end{cases} \tag{3.2}$$

3.1.2 Pin modeling

Pins are ports that allow modules to interact with the outside world. Each module in the chip has many pins that can interact with other modules. Pins in this thesis have two main parameters: horizontal and vertical coordinate position offsets relative to the upper left corner endpoint of the module to which they belong. And through these two offset values, the absolute coordinates of each pin can be calculated.

The following is an example: Suppose there is a module with width w, height h, and orientation 0 degree. The module has a pin with a horizontal relative offset of w/2 and a vertical relative offset of 0. The location of the pin is shown in Figure 3.4.

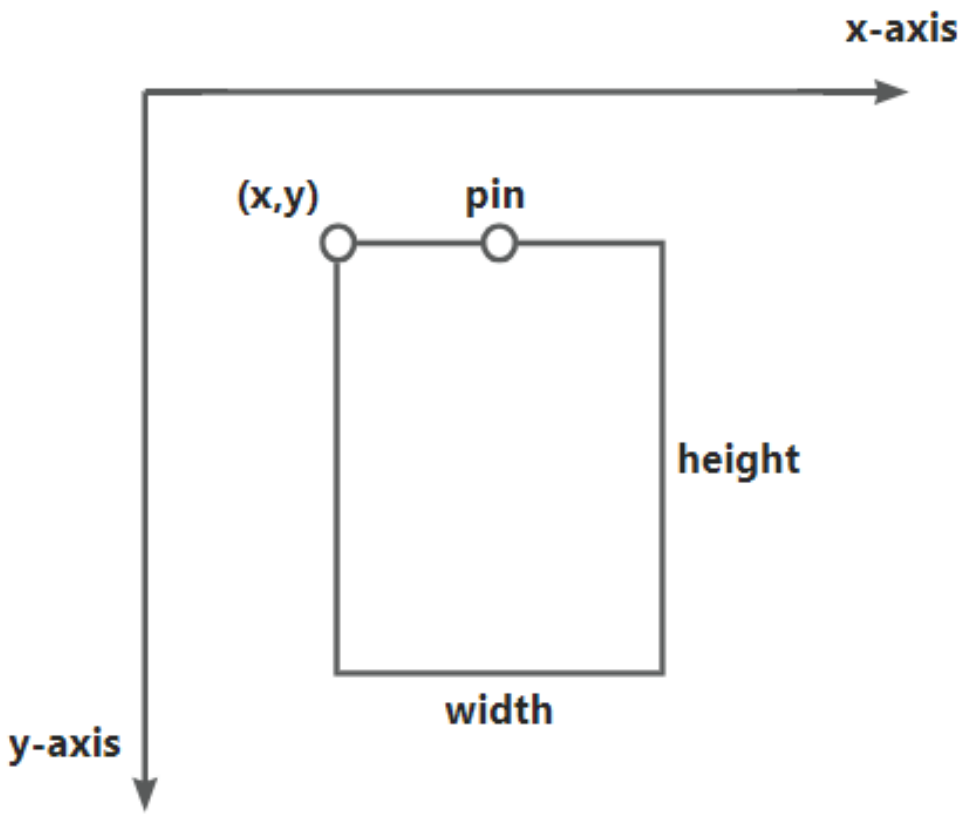


Figure 3.4: Example of a pin's location.

3.1.3 Channel modeling

Channels are mainly responsible for connecting pins or pins to external pins to ensure interactions between modules. There are two kinds of channels: one is a channel that connects on the same horizontal layer, and the other is a channel that connects two pins on different layers.

Although in reality, the channels of microfluidic chips are usually square, in this thesis, channels are modeled as round shape tubes because the primary interest is in understanding the average behavior of the fluid rather than edge effects or corner vortices that might develop in a square channel so that a circular approximation could be sufficiently accurate. Experimental evidence suggests that when operating within sufficiently low Reynolds number regimes, the discrepancy in pressure loss between square and circular tubes is confined to a marginal variation of less than 5% [18]. Besides that, circular channels have well-established and straightforward equations for fluid dynamics, making it easier to calculate fluid flow pressure losses later.

And in this thesis, channels have seven main parameters: the horizontal and vertical coordinates of the two end points of the circular tube, the radius of the circular tube, the

connection which the channel belongs to and the index number in the connection. The connection mentioned contains the connection information of a pair of pins, including the detailed information of all channels used to achieve the relationship between these two pins. And if a channel connects two different layers, it will have two additional parameters: information about two connected layers.

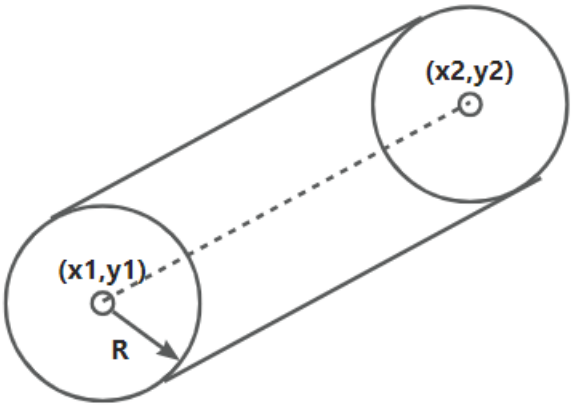


Figure 3.5: Schematic diagram of channel modeling on the same layer.

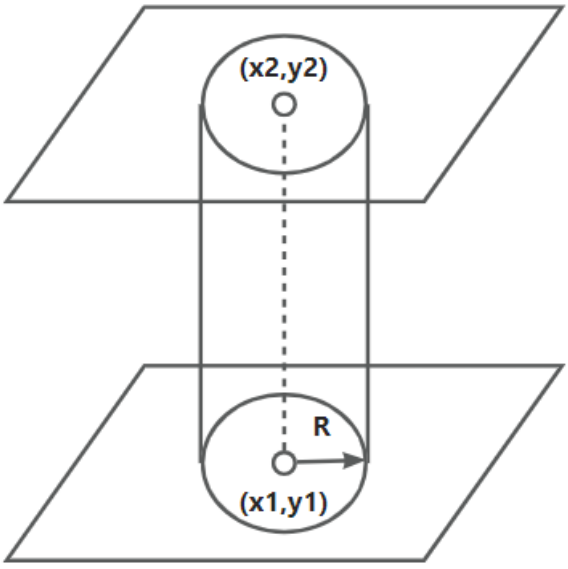


Figure 3.6: Schematic diagram of channel modeling on different layers.

3.2 Constraint modeling

This thesis mainly studies constraints related to channels, which include the primary constraints of a channel itself, the constraints of non-collision between channels and channels, channels and modules, the constraints of channels belonging to the same connection, and the pressure constraints of the liquid flowing in a channel.

3.2.1 Basic constraints of a channel

The basic constraints of a channel can be divided into two situations according to the channel type: channels connecting pins on the same layer and channels connecting pins on different layers, which are discussed separately below.

Basic constraints of a channel connecting pins on the same layer

A channel connecting pins on the same layer can only be parallel to the x-axis or y-axis of the coordinate system and can not exceed the chip area. These constraints can be expressed by Equation 3.4, Equation 3.5, Equation 3.6, Equation 3.7 and Equation 3.8:

$$(y_1 = y_2) \vee (x_1 = x_2) \tag{3.3}$$

$$\begin{cases} y_1 \leq y_2 + b_1 M \\ y_2 \leq y_1 + b_1 M \end{cases} \tag{3.4}$$

$$\begin{cases} x_1 \leq x_2 + b_2 M \\ x_2 \leq x_1 + b_2 M \end{cases} \tag{3.5}$$

$$\begin{cases} r \leq y_1 + b_1 M \\ y_1 \leq H - r + b_1 M \\ 0 \leq x_1 + b_1 M \\ x_1 \leq W + b_1 M \\ 0 \leq x_2 + b_1 M \\ x_2 \leq W + b_1 M \end{cases} \tag{3.6}$$

$$\begin{cases} r \leq x_1 + b_2 M \\ x_1 \leq W - r + b_2 M \\ 0 \leq y_1 + b_2 M \\ y_1 \leq H + b_2 M \\ 0 \leq y_2 + b_2 M \\ y_2 \leq H + b_2 M \end{cases} \tag{3.7}$$

$$b_1 + b_2 = 1 \tag{3.8}$$

where (x_1, y_1) and (x_2, y_2) represent the coordinates of the two endpoints of the channel, b_1 and b_2 are two auxiliary binary variables that determine whether the channel is parallel to the x-axis or the y-axis: if b_1 is set to 0, it means that y_1 must equal y_2, and the channel is parallel to the x-axis, and conversely if b_2 is set to 0, it means that x_1 must equal x_2 and the channel is parallel to the y-axis, and M is a very large constant. In addition, r represents the channel's radius, and W and H represent the width and height of the chip, respectively.

Basic constraints of a channel connecting pins on different layers

A channel connecting pins on different layers must be perpendicular to the layers of the chip and can not exceed the area of the chip. These constraints can be expressed by Equation 3.9 and Equation 3.10:

$$\begin{cases} x_1 = x_2 \\ y_1 = y_2 \end{cases} \tag{3.9}$$

$$\begin{cases} r \leq x_1 \\ x_1 \leq W - r \\ r \leq y_1 \\ y_1 \leq H - r \end{cases} \tag{3.10}$$

where (x_1, y_1) and (x_2, y_2) represent the coordinates of the two endpoints of the channel, and W, H, and r represent the chip's width, height and the channel's radius, respectively.

3.2.2 Constraints of non-collision

Constraints of non-collision require that channels and modules on the same layer do not overlap. The following is a detailed introduction to the two situations where channels and channels on the same layer do not overlap, and channels and modules on the same layer do not overlap.

Constraints of non-collision between channels on the same layer

Since the current discussion is about two channels, according to the previous formulas Equation 3.4 to Equation 3.8, it can be seen that the first and second channels each have two auxiliary binary variables, of which the first channel owns $b_{1,1}$ and $b_{1,2}$ and the second channel holds $b_{2,1}$ and $b_{2,2}$. When $b_{1,1}$ and $b_{2,1}$ are set to 0, it means two channels are parallel to the x-axis, and when $b_{1,2}$ and $b_{2,2}$ are set to 0, it means two channels are parallel to the y-axis.

And Non-collision between channels on the same layer can be further divided into three situations:

- Two channels are parallel to the x-axis.
- Two channels are parallel to the y-axis.
- One channel is parallel to the x-axis, and the other is parallel to the y-axis.

In the following paragraphs, the constraint construction in these three cases will be detailed.

First, the constraints about two channels parallel to the x-axis will be introduced.

This thesis uses two sets of constraints to cover all the possibilities of relative positions between two channels parallel to the x-axis on the same layer. Figure 3.7, Equation 3.11 and Equation 3.12 describe the first set of constraints, and Figure 3.8, Equation 3.13 and Equation 3.14 represent the second:

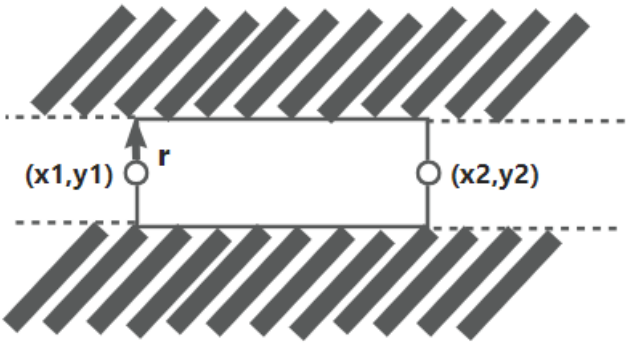


Figure 3.7: First set of constraints(Top view of the chip layer). The shadow in the figure is the position the second channel can occupy using the first set of constraints.

$$r + r^{'} \leq |y_1 - y_1^{'}| \tag{3.11}$$

$$\begin{cases} r + r^{'} \leq y_1 - y_1^{'} + b_1 M + b_{1,1} M + b_{2,1} M \\ y_1 - y_1^{'} \leq -r - r^{'} + b_2 M + b_{1,1} M + b_{2,1} M \end{cases} \tag{3.12}$$

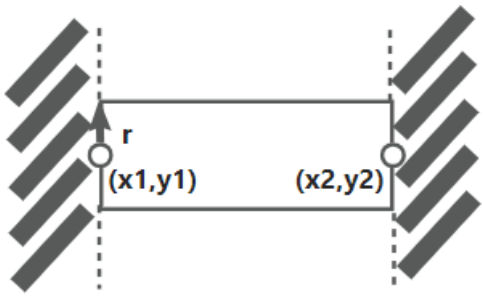


Figure 3.8: Second set of constraints(Top view of the chip layer). The shadow in the figure is the position the second channel can occupy using the second set of constraints.

$$((x_1^{'} \leq x_1) \wedge (x_2^{'} \leq x_1) \wedge (x_1^{'} \leq x_2) \wedge (x_2^{'} \leq x_2)) \vee ((x_1 \leq x_1^{'}) \wedge (x_2 \leq x_1^{'}) \wedge (x_1 \leq x_2^{'}) \wedge (x_2 \leq x_2^{'})) \tag{3.13}$$

$$\begin{cases} x_1^{'} \leq x_1 + b_3 M + b_{1,1} M + b_{2,1} M \\ x_2^{'} \leq x_1 + b_3 M + b_{1,1} M + b_{2,1} M \\ x_1^{'} \leq x_2 + b_3 M + b_{1,1} M + b_{2,1} M \\ x_2^{'} \leq x_2 + b_3 M + b_{1,1} M + b_{2,1} M \\ x_1 \leq x_1^{'} + b_4 M + b_{1,1} M + b_{2,1} M \\ x_2 \leq x_1^{'} + b_4 M + b_{1,1} M + b_{2,1} M \\ x_1 \leq x_2^{'} + b_4 M + b_{1,1} M + b_{2,1} M \\ x_2 \leq x_2^{'} + b_4 M + b_{1,1} M + b_{2,1} M \end{cases} \tag{3.14}$$

$$b_1 + b_2 + b_3 + b_4 = 3 \tag{3.15}$$

where r and r' represent the first and second channels' radius, respectively. (x_1,y_1) and (x_2,y_2) represent the first channel's two endpoints' coordinates, and ($x_1^{'}$,$y_1^{'}$) and ($x_2^{'}$,$y_2^{'}$) represent the second one's. In addition, b_1, b_2, b_3, and b_4 are four auxiliary binary variables to determine which kind of constraints work, and M is a huge constant. And the reason for adding $b_{1,1}M + b_{2,1}M$ after each equation is that if two channels are not all parallel to the x-axis, which means $b_{1,1}$ and $b_{2,1}$ are not all equal to 0, the constraints will always be satisfied.

Second, the thesis will introduce the constraints of two channels all parallel to the y-axis. As with the case where both channels are parallel to the x-axis, the thesis also uses two alternative sets of constraints to cover all cases. Figure 3.9, Equation 3.16 and Equation 3.17 describe the first set of constraints, and Figure 3.10, Equation 3.18 and Equation 3.19 represent the second:

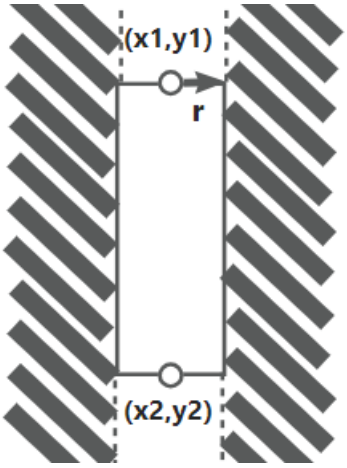


Figure 3.9: First set of constraints(Top view of the chip layer). The shadow in the figure is the position the second channel can occupy using the first set of constraints.

$$r + r^{'} \leq |x_1 - x_1^{'}| \tag{3.16}$$

$$\begin{cases} r + r^{'} \leq x_1 - x_1^{'} + b_5 M + b_{1,2} M + b_{2,2} M \\ x_1 - x_1^{'} \leq -r - r^{'} + b_6 M + b_{1,2} M + b_{2,2} M \end{cases} \tag{3.17}$$

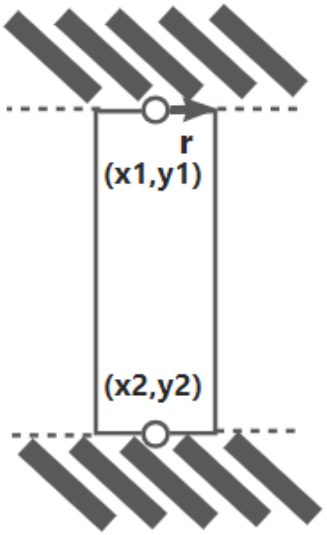


Figure 3.10: Second set of constraints(Top view of the chip layer). The shadow in the figure is the position the second channel can occupy using the second set of constraints.

$$((y_1^{'} \leq y_1) \wedge (y_2^{'} \leq y_1) \wedge (y_1^{'} \leq y_2) \wedge (y_2^{'} \leq y_2)) \vee ((y_1 \leq y_1^{'}) \wedge (y_2 \leq y_1^{'}) \wedge (y_1 \leq y_2^{'}) \wedge (y_2 \leq y_2^{'})) \tag{3.18}$$

$$\begin{cases} y_1^{'} \leq y_1 + b_7 M + b_{1,2} M + b_{2,2} M \\ y_2^{'} \leq y_1 + b_7 M + b_{1,2} M + b_{2,2} M \\ y_1^{'} \leq y_2 + b_7 M + b_{1,2} M + b_{2,2} M \\ y_2^{'} \leq y_2 + b_7 M + b_{1,2} M + b_{2,2} M \\ y_1 \leq y_1^{'} + b_8 M + b_{1,2} M + b_{2,2} M \\ y_2 \leq y_1^{'} + b_8 M + b_{1,2} M + b_{2,2} M \\ y_1 \leq y_2^{'} + b_8 M + b_{1,2} M + b_{2,2} M \\ y_2 \leq y_2^{'} + b_8 M + b_{1,2} M + b_{2,2} M \end{cases} \tag{3.19}$$

$$b_5 + b_6 + b_7 + b_8 = 3 \tag{3.20}$$

where r and r' represent the first and second channels' radius, respectively. (x_1,y_1) and (x_2,y_2) represent the first channel's two endpoints' coordinates, and ($x_1^{'}$,$y_1^{'}$) and ($x_2^{'}$,$y_2^{'}$) represent the second one's. In addition, b_5, b_6, b_7, and b_8 are four auxiliary binary variables to determine which kind of constraints work, and M is a huge constant. And the reason for adding $b_{1,2}$M + $b_{2,2}$M after each equation is that if two channels are not all parallel to the y-axis, which means $b_{1,2}$ and $b_{2,2}$ are not all equal to 0, the constraints will always be satisfied.

Last, the thesis will introduce the constraints when one channel is parallel to the x-axis and the other is parallel to the y-axis. This situation differs from the previous two situations, where the two channels are parallel. One prejudgment and two sets of constraints are used to cover all possibilities.

Before adding two sets of constraints on a pair of channels, check whether the two channels belong to the same connection and have adjacent index numbers, which means these two channels are connected. If the pair of channels meets this prerequisite, the two sets of constraints will not be added. The reason for adding this prejudgment condition is to consider the situation shown in Figure 3.11. If this prerequisite is not added, two mutually perpendicular channels will never be allowed to connect.

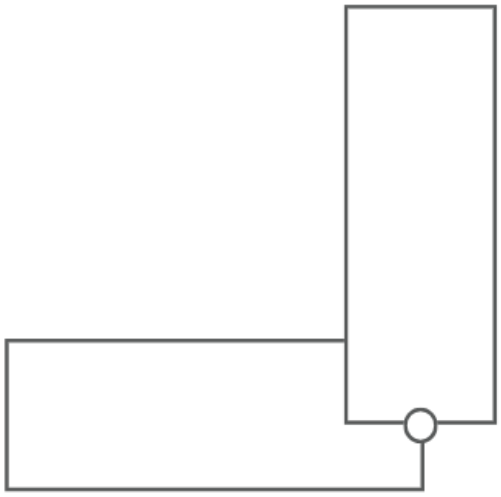


Figure 3.11: A situation where two channels are perpendicular and connected.

On the contrary, if this prerequisite is not satisfied, two sets of constraints will be added to these two channels. Figure 3.12, Equation 3.21 and Equation 3.22 describe the first set of constraint, and Figure 3.13, Equation 3.23 and Equation 3.24 describe the second set of constraints:

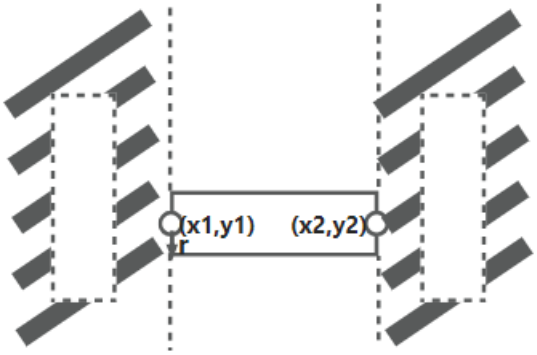


Figure 3.12: First set of constraints(Top view of the chip layer). The shadow in the figure is the position the second channel can occupy using the first set of constraints and dashed rectangles are examples of possible channel placements.

$$((x_1^{'} + r^{'} \leq x_1) \wedge (x_1^{'} + r^{'} \leq x_2)) \vee ((x_1^{'} - r^{'} \geq x_1) \wedge (x_1^{'} - r^{'} \geq x_2)) \tag{3.21}$$

$$\begin{cases} x_1^{'} + r^{'} \leq x_1 + b_9 M + b_{1,1} M + b_{2,2} M \\ x_1^{'} + r^{'} \leq x_2 + b_9 M + b_{1,1} M + b_{2,2} M \\ x_1 \leq x_1^{'} - r^{'} + b_{10} M + b_{1,1} M + b_{2,2} M \\ x_2 \leq x_1^{'} - r^{'} + b_{10} M + b_{1,1} M + b_{2,2} M \end{cases} \tag{3.22}$$

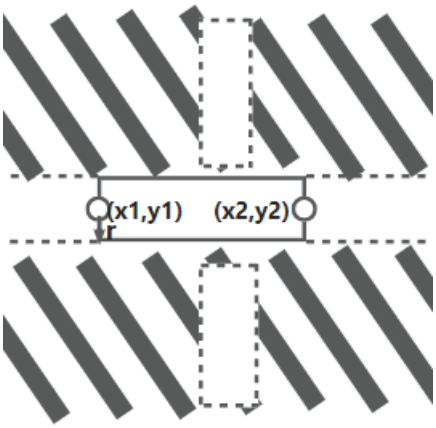


Figure 3.13: Second set of constraints(Top view of the chip layer). The shadow in the figure is the position the second channel can occupy using the second set of constraints and dashed rectangles are examples of possible channel placements.

$$((y_1^{'} + r \leq y_1) \wedge (y_2^{'} + r \leq y_1)) \vee ((y_1^{'} - r \geq y_1) \wedge (y_2^{'} - r \geq y_1)) \tag{3.23}$$

$$\begin{cases} y_1^{'} + r \leq y_1 + b_{11} M + b_{1,1} M + b_{2,2} M \\ y_2^{'} + r \leq y_1 + b_{11} M + b_{1,1} M + b_{2,2} M \\ y_1 \leq y_1^{'} - r + b_{12} M + b_{1,1} M + b_{2,2} M \\ y_1 \leq y_2^{'} - r + b_{12} M + b_{1,1} M + b_{2,2} M \end{cases} \tag{3.24}$$

$$b_9 + b_{10} + b_{11} + b_{12} = 3 \tag{3.25}$$

where r and r′ represent the first and second channels' radius, respectively. (x_1,y_1) and (x_2,y_2) represent the first channel's two endpoints' coordinates, and ($x_1{}^{'}$,$y_1{}^{'}$) and ($x_2{}^{'}$,$y_2{}^{'}$) represent the second one's. In addition, b_9, b_{10}, b_{11}, and b_{12} are four auxiliary binary variables to determine which kind of constraints work, and M is a huge constant. And the reason for adding $b_{1,1}M + b_{2,2}M$ after each equation is that if two channels are not perpendicular, which means $b_{1,1}$ and $b_{2,2}$ are not all equal to 0, the constraints will always be satisfied.

Constraints of non-collision between channels and modules on the same layer

Since the current discussion is about one channel, according to the previous formulas Equation 3.4 to Equation 3.8, it can be seen that the channel has two auxiliary binary variables, represented by $b_{1,1}$ and $b_{1,2}$. When $b_{1,1}$ is set to 0, the channel is parallel to the x-axis, and when $b_{1,2}$ is set to 0, the channel is parallel to the y-axis. This thesis will discuss the situation according to whether the channel is parallel to the x-axis or the y-axis.

First, when the channel is parallel to the x-axis, two sets of constraints will be used to cover all the possible placements of the channel, which will not cause a collision with the module. These two sets of constraints define the space that can be occupied by the channel in the vertical (top and bottom) and horizontal (left and right) directions.

Figure 3.14, Equation 3.26 and Equation 3.27 describe the constraints on the space that can be occupied above and below the module, and Figure 3.15, Equation 3.28 and Equation 3.29 describe the constraints on the available space to the left and to the right of the module:

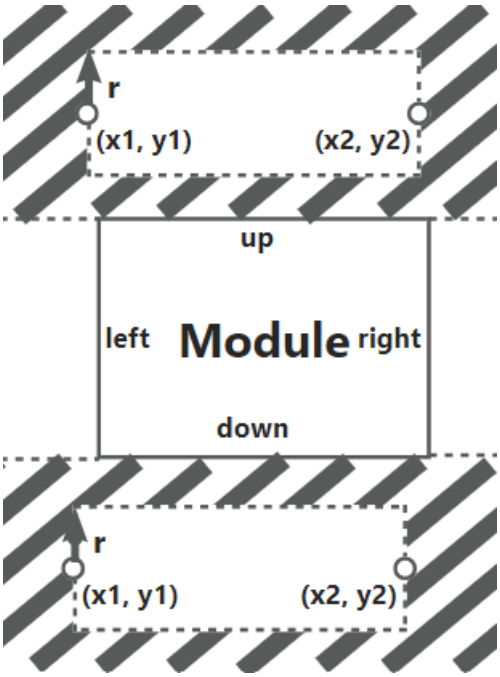


Figure 3.14: First set of constraints(Top view of the chip layer). The shadow in the figure is the position the channel can occupy using the first set of constraints and dashed rectangles are examples of possible channel placements.

$$(y_1 + r \leq up) \vee (y_1 - r \geq down) \tag{3.26}$$

$$\begin{cases} y_1 + r \leq up + b_{1,1}M + b_1M \\ down \leq y_1 - r + b_{1,1}M + b_2M \end{cases} \tag{3.27}$$

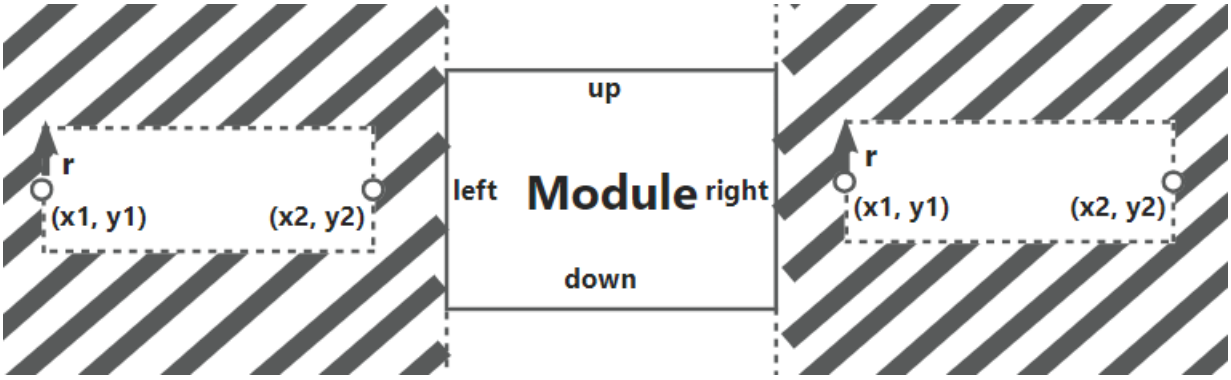


Figure 3.15: Second set of constraints(Top view of the chip layer). The shadow in the figure is the position the channel can occupy using the second set of constraints and dashed rectangles are examples of possible channel placements.

$$((x_1 \leq left) \wedge (x_2 \leq left)) \vee ((x_1 \geq right) \wedge (x_2 \geq right)) \tag{3.28}$$

$$\begin{cases} x_1 \leq left + b_{1,1}M + b_3M \\ x_2 \leq left + b_{1,1}M + b_3M \\ right \leq x_1 + b_{1,1}M + b_4M \\ right \leq x_2 + + b_{1,1}M + b_4M \end{cases} \tag{3.29}$$

$$b_1 + b_2 + b_3 + b_4 = 3 \tag{3.30}$$

Where r represents the channel's radius, and (x_1,y_1) and (x_2,y_2) represent the two endpoints' coordinates of the channel. Besides that, left, right, up, and down refer to the values of the four boundaries of the module, respectively. And b_1, b_2, b_3, and b_4 are four auxiliary binary variables that determine the channel will occupy which location relative to the module. For example, if b_1 is set to 1, the channel is above the module. Last but not least, the reason for adding $b_{1,1}M$ to each equation is to ensure the constraints only work when the channel is parallel to the x-axis, which means $b_{1,1}$ equals 0.

Second, if the channel is parallel to the y-axis, which is the same as parallel to the x-axis, two sets of constraints are adequate to cover all the free space above, below, left, and right of the module.

The first set of constraints is represented by Figure 3.16, Equation 3.31 and Equation 3.32, and Figure 3.17, Equation 3.33 and Equation 3.34 represents the situation where the second set of constraints considers:

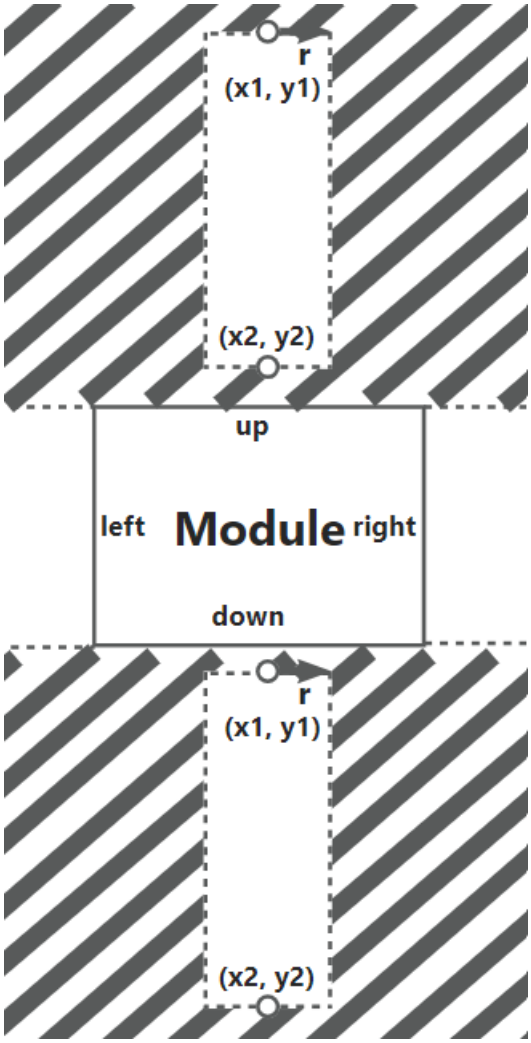


Figure 3.16: First set of constraints(Top view of the chip layer). The shadow in the figure is the position the channel can occupy using the first set of constraints and dashed rectangles are examples of possible channel placements.

$$((y_1 \leq up) \wedge (y_2 \leq up)) \vee ((y_1 \geq down) \wedge (y_2 \geq down)) \tag{3.31}$$

$$\begin{cases} y_1 \leq up + b_{1,2}M + b_5M \\ y_2 \leq up + b_{1,2}M + b_5M \\ down \leq y_1 + b_{1,2}M + b_6M \\ down \leq y_2 + b_{1,2}M + b_6M \end{cases} \tag{3.32}$$

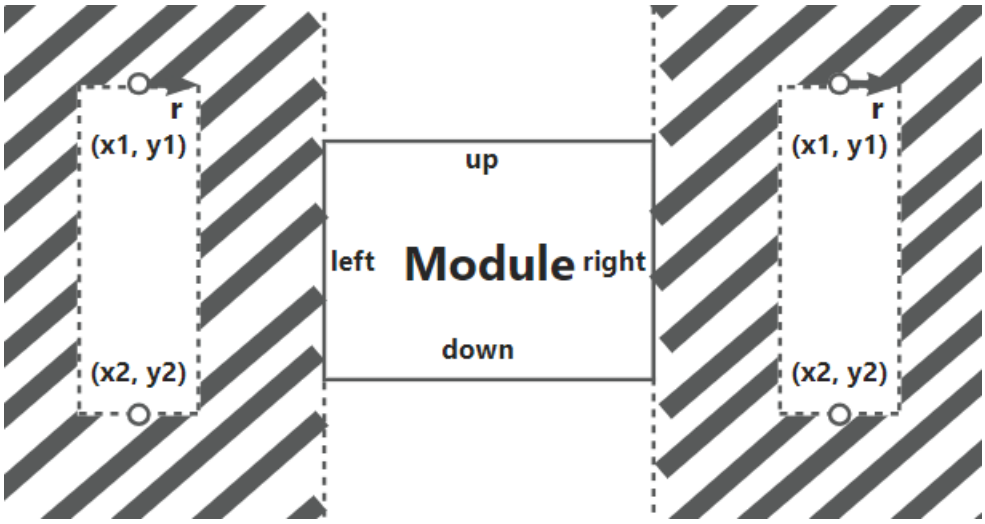


Figure 3.17: Second set of constraints(Top view of the chip layer). The shadow in the figure is the position the channel can occupy using the second set of constraints and dashed rectangles are examples of possible channel placements.

$$(x_1 + r \leq left) \vee (x_1 - r \geq right) \tag{3.33}$$

$$\begin{cases} x_1 + r \leq left + b_{1,2}M + b_7M \\ right \leq x_1 - r + b_{1,2}M + b_8M \end{cases} \tag{3.34}$$

$$b_5 + b_6 + b_7 + b_8 = 3 \tag{3.35}$$

Where r represents the channel's radius, and (x_1,y_1) and (x_2,y_2) represent the two endpoints' coordinates of the channel. Besides that, left, right, up, and down refer to the values of the four boundaries of the module, respectively. And b_5, b_6, b_7, and b_8 are four auxiliary binary variables that determine the channel will occupy which location relative to the module. For example, if b_5 is set to 1, the channel is above the module. Last but not least, the reason for adding $b_{1,2}M$ to each equation is to ensure the constraints only work when the channel is parallel to the y-axis, which means $b_{1,2}$ equals 0.

3.2.3 Constraints of channels belonging to the same connection

The connection mentioned here means a connection between two pins and a set of many channels is needed instead of just one channel to accomplish this connection. This subsection mainly focuses on constraints about channels within the same set.

It is required that the first endpoint of the first channel in a set of channels must coincide with the first pin, and the second endpoint of the last channel in the same set must coincide with the second pin. In addition, these channels in the same set must be guaranteed to be connected end to end. That is, the second endpoint of the previous channel must coincide with the first endpoint of the following channel.

The constraints are shown in Figure 3.18, Equation 3.36 and Equation 3.37:

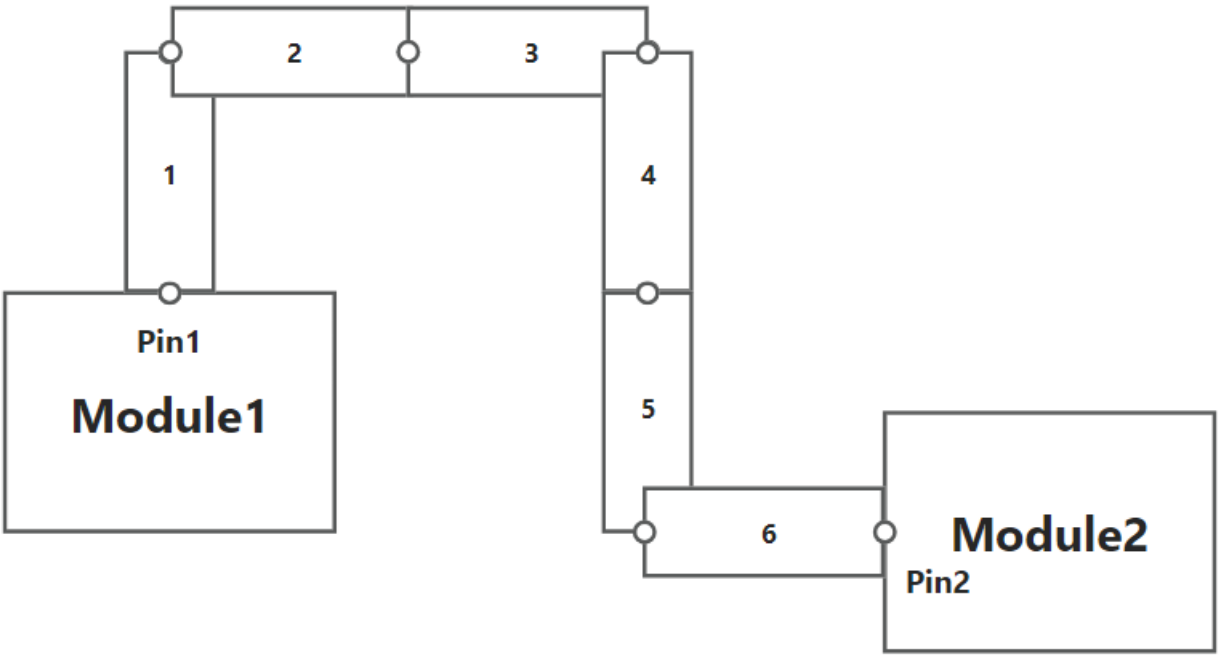


Figure 3.18: A connection example where a set of six channels realizes the connection work of two pins.

$$\begin{cases} x_0 = pin_1x \\ y_0 = pin_1y \\ x_{n-1}^{'} = pin_2x \\ y_{n-1}^{'} = pin_2y \end{cases} \tag{3.36}$$

$$\begin{cases} x_i^{'} = x_{i+1} \\ y_i^{'} = y_{i+1} \end{cases} \quad i = 0, 1, \ldots, n-1 \tag{3.37}$$

where n represents the the number of channels in a set, and (x_i,y_i) and ($x_i^{'}$,$y_i^{'}$) express the coordinates of the first endpoint and the second endpoint in the i-th channel, respectively. In addition, the coordinates of two pins are represented by (pin_1x,pin_1y) and (pin_2x,pin_2y).

3.2.4 Constraints on pressure losses in channels

The program described in this thesis can specifically design the wiring layout of channels based on the user's expectations for the fluid pressure at a particular output pin. The prerequisite is that users provide the input pressure for the channels, the expected steady flow rate, and the density and dynamic viscosity of the liquid flowing through the channels.

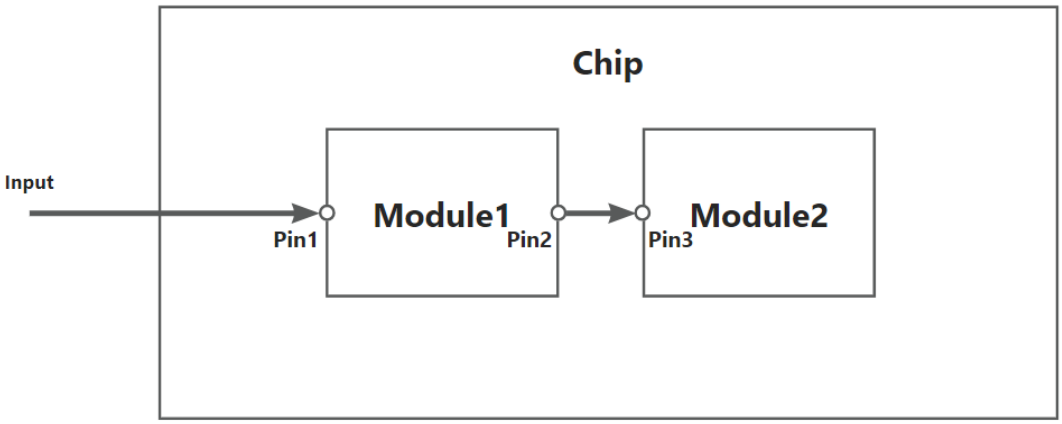


Figure 3.19: An example of constraints on pressure losses. The input fluid pressure is given, and now users make expectations on the liquid pressure at Pin3.

The idea of implementing this function is that the inlet pressure and the expected outlet pressure are already known. Therefore, the pressure that should be lost when the fluid flows in the channels can be calculated. Based on the pressure that should be lost in the channel flow, the total length of the liquid that should flow in the channel and how many right-angle turns it should make in the channel can be calculated. These lengths and the number of turns

can be used as constraints to generate the final channel layout.

First, the thesis will introduce the calculation method of pressure losses when liquid flows in a straight channel and when a channel turns 90 degrees.

Pressure losses when liquid flows in a straight channel

In the thesis, the pressure losses when fluid flows through straight channels are calculated by the Navier-Stokes equation.

And before using the Navier-Stokes equation, three assumptions are made. These three assumptions and their rationality are explained below.

The first assumption is that the fluid flowing in channels is an incompressible Newtonian fluid because microfluidic chips typically operate at small length scales, such as micrometers or nanometers. At these scales, the fluid's volume or density changes as it flows through the device are negligible. Therefore, the fluid can be approximated as incompressible, adhering to the assumption of constant density. Besides that, many microfluidic applications involve fluids with relatively low concentrations of solutes or particles, and the shear rates experienced in microfluidic channels are often within a range where the fluid behaves as a Newtonian fluid. The assumption of constant viscosity simplifies the mathematical modeling of fluid flow in microfluidic channels, making it easier to predict and analyze flow behavior.

The second assumption is that the fluid flowing in channels is laminar because, in microfluidic chips, the flow velocities and channel dimensions are small, resulting in low Reynolds numbers, and at low Reynolds numbers, the flow tends to be laminar and predictable, with viscous forces dominating over inertial forces. In addition, microfluidic channels have small cross-sectional dimensions, often in the micrometer range, and the small channel sizes constrain the fluid flow and inhibit the development of turbulent flow, promoting laminar behavior. Last but not least, flow rates in microfluidic chips are usually low, contributing to the development and maintenance of laminar flow. Low flow rates reduce the likelihood of turbulence and mixing between different fluid layers, resulting in well-defined, orderly flow patterns [19].

The last assumption is that no-slip condition takes effect, which means the velocity of the fluid at the channels' wall is 0. The reason why this assumption can be made is that in microfluidic devices, the channels are microscopic, typically in the range of micrometers or smaller, and the fluid molecules in contact with the solid surface experience a significant amount of interaction and friction due to their proximity to the solid material. At this scale, the effects of molecular interactions dominate and tend to cause the fluid to stick to the surface, adhering to a no-slip condition.

Based on these three assumptions, the Navier-Stokes equation is shown in Equation 3.38 and a schematic diagram of fluid flow in a channel is shown in Figure 3.20:

$$\rho(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u}) = -\nabla p + \eta \nabla^2 \boldsymbol{u} + \boldsymbol{f} \tag{3.38}$$

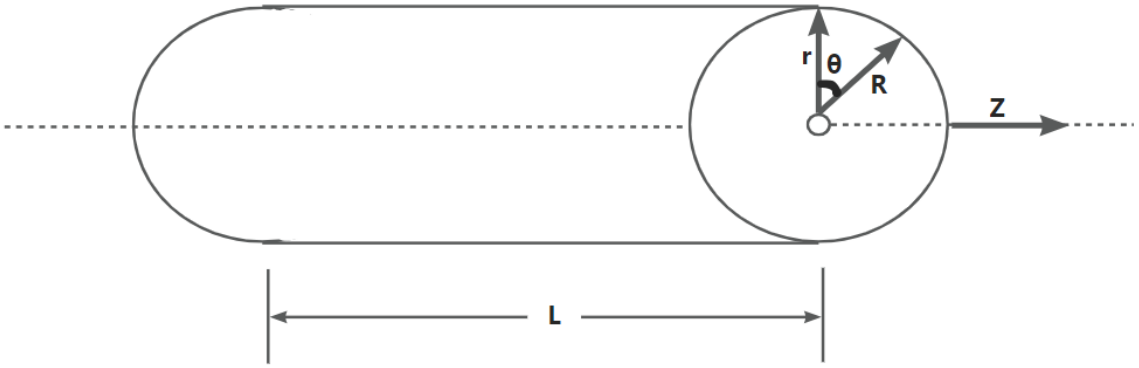


Figure 3.20: A schematic diagram of fluid flow in a microfluidic channel. R and L represent the channel's radius and length, respectively, and the z-axis direction represents the flow direction of the fluid. In addition, (z,r,θ) forms a cylindrical coordinates.

where ρ, η and p represent the density, the dynamic viscosity, and the pressure of the fluid, respectively. $\boldsymbol{u}$ represents the velocity of the fluid along the z-direction shown in Figure 3.20, and f represents densities of body force, such as gravity and electromagnetic force. In fact, the Navier-Stokes equation is a continuum representation of Newton's second law ($F = ma$) in terms of acceleration (a) per unit volume [19]. $\frac{\partial \boldsymbol{u}}{\partial t}$ represents the unsteady term or the time derivative of the velocity field $\boldsymbol{u}$. It describes how the velocity of a fluid particle changes with time. $\boldsymbol{u} \cdot \nabla \boldsymbol{u}$ represents the convective acceleration or the advection of momentum. It is the change in velocity due to the motion of the fluid itself and is a nonlinear term. $-\nabla p$ represents the gradient of pressure p. The negative sign indicates that fluid moves from regions of high pressure to regions of low pressure. $\nabla^2 \boldsymbol{u}$ is the Laplacian of the velocity field, which represents the diffusion of momentum due to viscosity. This term is responsible for the viscous dissipation of energy within the fluid flow.

And in microfluidic chips, inertial forces are so small compared to viscous forces that the non-linear term in the Navier-Stokes equation can be neglected. So, the Navier-Stokes equation can be simplified as Equation 3.39 when calculating microfluidics [19].

$$\rho\frac{\partial \boldsymbol{u}}{\partial t} = -\nabla p + \eta\nabla^2\boldsymbol{u} + \boldsymbol{f} \tag{3.39}$$

Now, put all the variables into the Navier-Stokes equation to get Equation 3.40:

$$\rho\frac{\partial v_z}{\partial t} = -\frac{\partial P}{\partial z} + \eta\left[\frac{1}{r}\frac{\partial}{\partial r}(r\frac{\partial v_z}{\partial r}) + \frac{1}{r^2}\frac{\partial^2 v_z}{\partial \theta^2} + \frac{\partial^2 v_z}{\partial z^2}\right] + \rho g_z \tag{3.40}$$

Where v_z represents the velocity of the fluid flowing along the z-direction. Since the current consideration is the steady flow of fluid, the value of v only corresponds to the distance between the cross-section position of the fluid and the origin and has nothing to do with time, theta, and place along the z-direction, which means $\frac{\partial v_z}{\partial t}$, $\frac{\partial^2 v_z}{\partial \theta^2}$, and $\frac{\partial^2 v_z}{\partial z^2}$ always equal 0. Besides that, the thesis will first introduce how to calculate the losses of pressure in channels that connect two pins on the same layer, so the effect of gravity can also be ignored because it is perpendicular to the direction of the fluid flow, which means item ρg_z can also be removed from Equation 3.40. So Equation 3.40 can be further simplified to Equation 3.41.

$$0 = -\frac{\partial P}{\partial z} + \eta \cdot \frac{1}{r}\frac{\partial}{\partial r}(r\frac{\partial v_z}{\partial r}) \tag{3.41}$$

Next, the thesis processes Equation 3.41 to make it easy to integrate and get Equation 3.42.

$$\int \frac{\partial}{\partial r}(r\frac{\partial v_z}{\partial r})dr = \frac{1}{\eta}\frac{\partial P}{\partial z}\int r dr \tag{3.42}$$

After the first integration, the following equation can be obtained:

$$r\frac{\partial v_z}{\partial r} = \frac{1}{\eta}\frac{\partial P}{\partial z} \cdot \frac{1}{2}r^2 + C_1 \tag{3.43}$$

where C_1 represents a constant resulting from the integration process. To calculate the value of C_1, divide both sides of Equation 3.43 by r and then get Equation 3.44.

$$\frac{\partial v_z}{\partial r} = \frac{1}{\eta}\frac{\partial P}{\partial z} \cdot \frac{1}{2}r + \frac{C_1}{r} \tag{3.44}$$

When r approaches 0, the first term on the right side of Equation 3.44 ($\frac{1}{\eta}\frac{\partial P}{\partial z} \cdot \frac{1}{2}r$) comes 0, and the second term($\frac{C_1}{r}$) approaches infinity if C_1 does not equal 0, so it can be seen that the left side of the equation($\frac{\partial v_z}{\partial r}$) should also approach infinity. However, it is known that there is

no infinite momentum at the center of channels. So, C_1 must equal 0. Equation 3.44 can then be readjusted to Equation 3.45 to prepare for the second integration.

$$\int \frac{\partial v_z}{\partial r} dr = \frac{1}{2\eta} \frac{\partial P}{\partial z} \int r dr \tag{3.45}$$

After integration, the result is shown in Equation 3.46:

$$v_z(r) = \frac{1}{2\eta} \frac{\partial P}{\partial z} \cdot \frac{1}{2} r^2 + C_2 \tag{3.46}$$

where C_2 represents a constant resulting from the integration process. The assumption about the no-slip condition is used to calculate the value of C_2, which means the velocity of the part of the fluid in contact with the channels' surface is 0. So v_z(R) should be 0, and therefore C_2 should be $-\frac{1}{4\eta} \frac{\partial P}{\partial z} R^2$.

Now, the final expression for solving the pressure difference can be obtained through some sorting.

$$\triangle P = \frac{4\eta L \cdot v_z(r)}{R^2 \left(1 - \left(\frac{r}{R}\right)^2\right)} \tag{3.47}$$

In addition, the fluid flow rate mentioned in this thesis defaults to the velocity of the fluid at the center of the channel so that Equation 3.47 can be further simplified to Equation 3.48.

$$\triangle P = \frac{4\eta L \cdot v_z(R)}{R^2} \tag{3.48}$$

In the same way, this equation can be used for calculating the pressure losses in channels that connect two pins on different layers.

$$\triangle P = \eta L \left(\rho g_z - \frac{4 v_z(R)}{R^2}\right) \tag{3.49}$$

Pressure losses when liquid flows in a channel that turns 90 degrees

In this thesis, the Darcy-Weisbach equation calculates the pressure losses when fluid turns 90 degrees in channels. The Darcy–Weisbach equation in fluid dynamics is an empirical formula connecting the loss of pressure caused by friction over a specific channel length to the mean velocity of incompressible fluid flow. Its basic form is as follows [20]:

$$\frac{\triangle P}{L} = f_D \cdot \frac{\rho}{2} \cdot \frac{< v >^2}{D_H} \tag{3.50}$$

where ρ and v represent the density and the velocity of the fluid, respectively, and L represents the length of turn. Besides that, D_H represents the hydraulic diameter of the channel. For a channel with a circular cross-section, this value is equal to channel's diameter(in meters). For channels with non-circular cross-sections, D_H can be calculated as $D_H = \frac{4A}{P}$, where A represents the cross-sectional area and P is the perimeter (both in meters). Last but not least, f_D represents the Darcy friction factor, which is only related to the Reynolds number for laminar fluid [21].

The process of adding speed constraints

When receiving a requirement for the pressure difference between two pins, first find out how many connections and modules the path of the two pins contains. Then, count the channels' length of each connection and the number of turns and express them with a mathematical expression. Last, calculate the pressure losses according to the two methods of calculating pressure loss mentioned before and require it to be equal to the pressure difference given by the user.

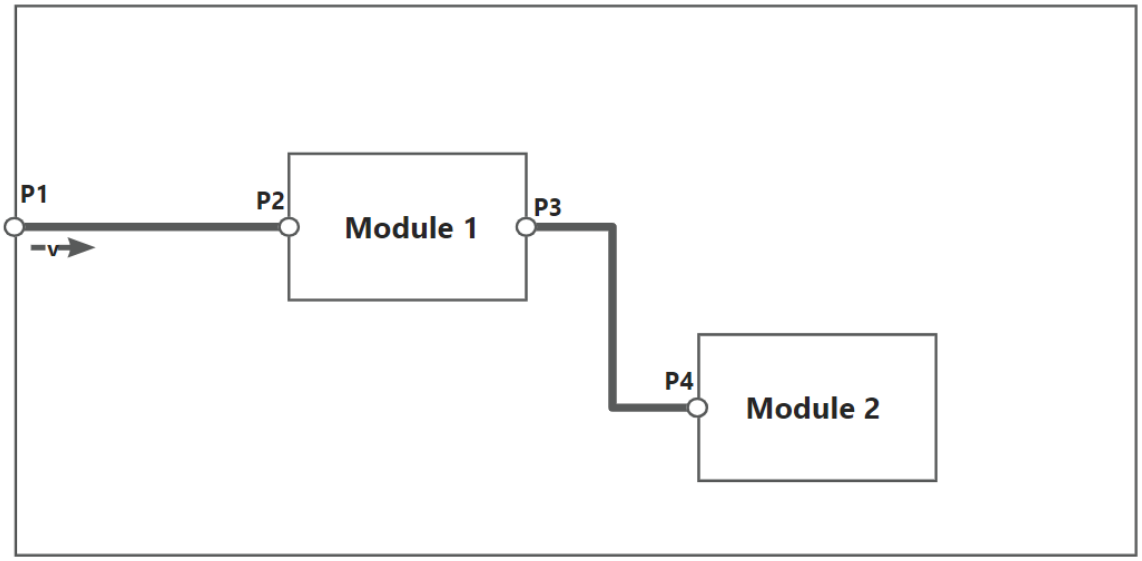


Figure 3.21: An example of constraints about pressure losses of channels on the same layer. The user specifies the velocity of the fluid at pin p1 and makes requirements for the pressure difference between p1 and p4.

Here is an instance shown in Figure 3.21. First, the program will figure out that there are two connections and one module on the path between p1 and p4, and then, the program will

compute the total length of channels between p1 and p2 and between p3 and p4. Besides that, the program will also count the number of turns on the path, which is 2 in this figure. Last, the program will use the Navier-Stokes equation and the Darcy-Weisbach equation to calculate the total pressure losses and require it to equal the pressure difference users expected in Gurobi.

The equation used for counting turns and calculating the total length of connections is shown in Equation 3.51:

$$\begin{cases} t_i = 1 - abs\left(\dfrac{x_i^{'} - x_i}{x_i^{'} - x_i + y_i^{'} - y_i} \cdot \dfrac{x_{i+1}^{'} - x_{i+1}}{x_{i+1}^{'} - x_{i+1} + y_{i+1}^{'} - y_{i+1}} + \dfrac{y_i^{'} - y_i}{x_i^{'} - x_i + y_i^{'} - y_i} \cdot \dfrac{y_{i+1}^{'} - y_{i+1}}{x_{i+1}^{'} - x_{i+1} + y_{i+1}^{'} - y_{i+1}}\right) \\ l_i = abs(x_i^{'} - x_i) + abs(y_i^{'} - y_i) \\ t = \sum_{i=0}^{n-2} t_i \\ l = \sum_{i=0}^{n-1} l_i \end{cases} \tag{3.51}$$

where (x_i,y_i) and ($x_i^{'}$,$y_i^{'}$) represent two endpoints' coordinates of i-th channel in a connection, respectively. n represents the total number of channels in a connection. Besides that, t_i denotes the i-th channel in a connection whether it has a turn. When t_i equals 0, there is no turn; when t_i equals 1, there is a turn. And t represents the total number of turns in a connection. Last but not least, l_i expresses the length of the i-th channel in a connection, and l represents the total length of a connection.

4 Program performance optimization

By passing the mathematical modeling of the chip and chip internal constraints in chapter 3 as input to Gurobi, the program has the essential ability to automatically design channel connections based on chip information.

However, in actual applications, the program will often encounter channel connection design needs of large chips, which contain dozens of modules and dozens of connections and it takes a long time for these large-scale design projects to produce results, which cannot meet users' expectations. Therefore, in addition to realizing the basic functions of channel design, this thesis optimizes the program's performance in three aspects. These optimizations are introduced separately below.

4.1 Chip's downscaling and upscaling

Typically, most microfluidic chips are several thousand microns by several thousand microns in size. In Gurobi, each unit area, the square of a micrometer, is used to explore the possibility of a solution, which means that the number of unit squares that need to be explored will increase exponentially as the chip size increases, which will lead to the significant increase on the running time of the program.

On the contrary, if the chip area is reduced, the number of unit squares that the Gurobi optimizer needs to explore can be reduced, and the channel design results can be obtained quickly.

Based on this idea, a chip information preprocessing module is added before the chip data is input to the program, which is responsible for dividing the size parameter of the chip and the modules inside by a scaling factor to reduce the area of the chip.

Besides that, a reduction module is added before the program outputs. This reduction module will multiply the size data of the chip, the modules inside, and the channel design by the same scaling factor in the chip information preprocessing module to restore them to normal size.

The flow chart of the entire process of downscaling and upscaling is as follows:

Figure 4.1: Optimization flow chart of downscaling and upscaling.

The key to downscaling and upscaling optimization is selecting the scaling factor. The method of selecting the scaling factor will be introduced in detail below. The core target of selecting the scaling factor is to scale the chip and internal components as small as possible while ensuring no overlap between the edge of the chip and modules inside, modules and modules, and pins and pins.

To achieve this target, first, the minimum distances between the four boundaries of the chip and the module, between modules, and between pins should be calculated. In addition, a distance threshold will be preset in advance, representing the acceptable minimum value of the distance between modules and the surrounding boundaries of the chip, modules and modules, and pins after chip scaling. Subsequently, one only needs to use the threshold set in advance to remove the statistically calculated minimum distance in the chip to obtain the optimal scaling factor, as shown in Equation 4.1:

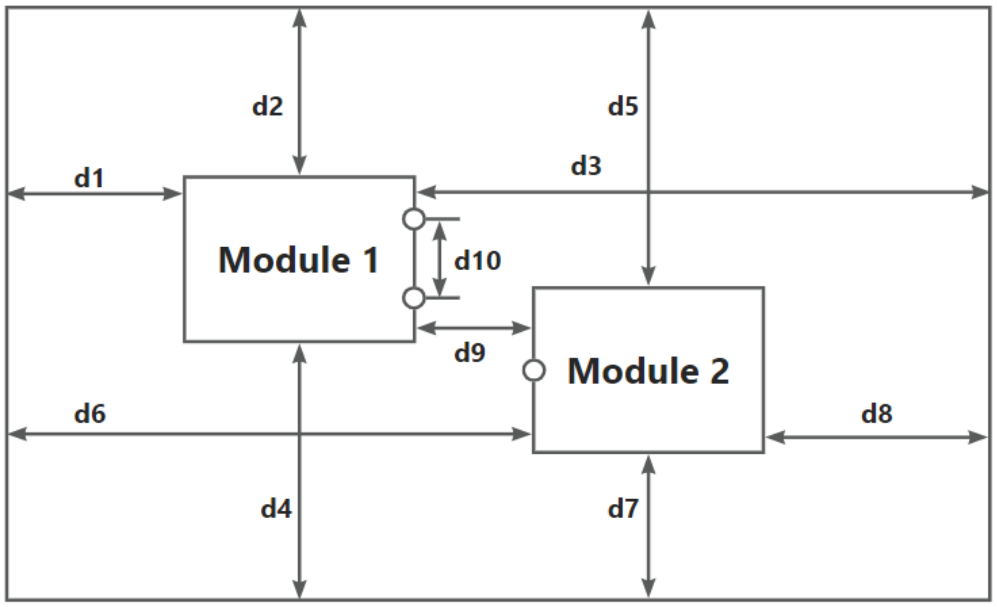


Figure 4.2: An example of counting the minimum distance between modules and the four boundaries of the chip, modules and modules, pins and pins.

$$\alpha = \frac{d_{min}}{threshold} \tag{4.1}$$

where α represents the scaling factor, d_{min} denotes the minimum distance, and *threshold* refers to the minimum distance threshold predetermined by the user.

4.2 An advance estimate of the number of channels required in a connection

Before passing parameters to the Gurobi optimizer for a solution, it is necessary to determine how many channels each connection consists of. If the number of channels required for all connections is uniformly set to the default value, it will likely result in redundant parameters. Therefore, this thesis introduces a functional module that can estimate the number of channels required for each connection based on the relative positions of each module and pin in the chip in adavance. In other words, customize each connection's required number of channels according to its situation. The following describes how this functional module works.

Since channels can only be horizontal or vertical, the number of channels required for a connection is related to the number of turns in the connection path because each turn requires converting channels to change the direction of travel. More precisely, the number of channels needed for a connection equals the minimum number of necessary turns plus 1. An example is shown in Figure 4.3. According to Figure 4.3, there are 2 turns; therefore, 3 channels are needed.

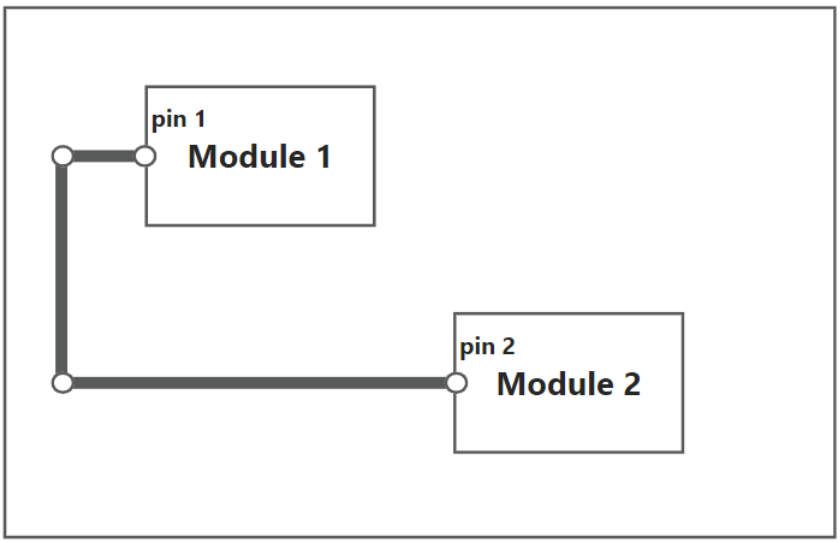


Figure 4.3: An example of calculating the number of channels needed in a connection.

And now, the core problem is estimating the minimum number of turns required for a connection. In this thesis, A^* algorithm is used to estimate the minimum turns of a connection.

A* is an informed search algorithm, categorized as a best-first search method, designed for solving problems in weighted graphs. It commences from a designated initial node within the graph and strives to identify the path to a specified goal node with the least cost, which can be defined as the minimal distance traveled or shortest time taken, among other factors. This approach involves the creation of a tree structure that branches out from the starting node, continuously expanding these paths by incrementally adding one edge at a time until certain termination conditions are met. During each cycle of its primary loop, A* must decide which path to prolong [15].

This choice hinges on two key factors: the current cost of the path and an estimation of the cost needed to extend that path to reach the goal. In particular, A* opts for the path that minimizes [15]:

$$f(n) = g(n) + h(n) \tag{4.2}$$

Here, n represents the next node in the path, g(n) signifies the cumulative cost from the start node to node n, and h(n) is a heuristic function providing an estimate of the most economical path cost from n to the goal. A* concludes its operation when it either identifies a path extending from the starting point to the goal or encounters a situation where no eligible paths for extension remain. The appropriateness of the heuristic function is reliant on the specific problem at hand. If the heuristic function is deemed admissible, indicating it never overestimates the actual cost to reach the goal, A* is assured to yield the least-cost path from the start to the goal [15].

Below is an example of applying the A* algorithm to solve a problem that aims to find the shortest distance from Arad to Bucharest. In this problem, h(n) equals the distance from the current place to Bucharest.

Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

Figure 4.4: The distance from each city to Bucharest [15].

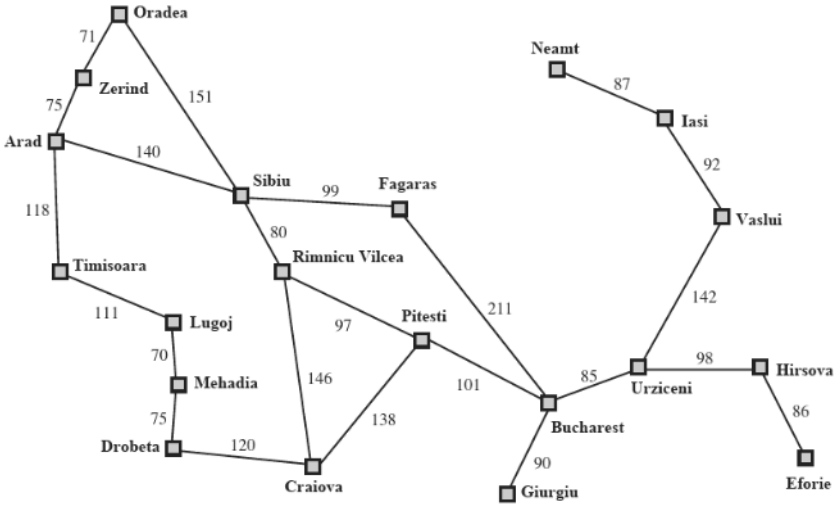


Figure 4.5: A road map of part of Romania [15].

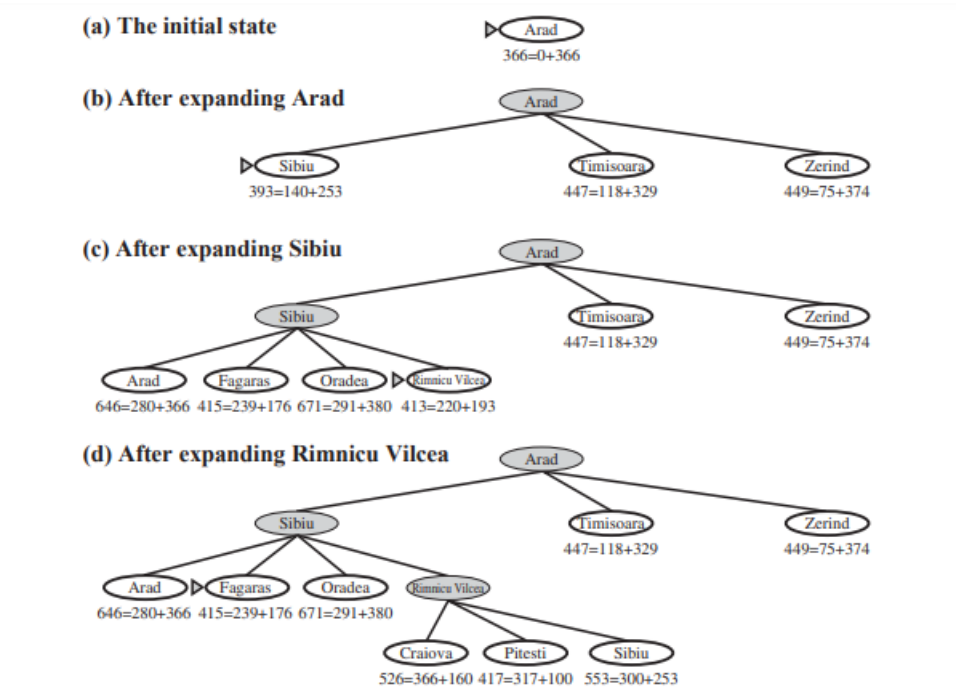


Figure 4.6: The stages in an A* search for Bucharest. Nodes are labeled with f = g + h, where h values represent the straight-line distances to Bucharest as obtained from Figure 4.4 [15].

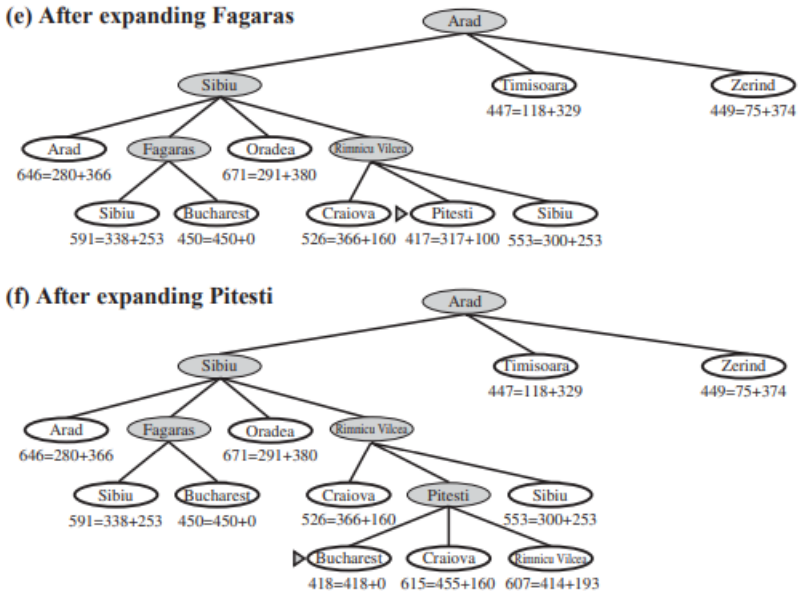


Figure 4.7: The stages in an A* search for Bucharest. Nodes are labeled with f = g + h, where h values represent the straight-line distances to Bucharest as obtained from Figure 4.4 [15].

As mentioned above, the selection of h(n) varies depending on the type of problem. And the h(n) used in this thesis is shown in Equation 4.3 and Equation 4.4:

$$p = (x_n - x_{n-1}) \cdot (x_{n-1} - x_{n-2}) + (y_n - y_{n-1}) \cdot (y_{n-1} - y_{n-2}) \tag{4.3}$$

$$h(n) = \begin{cases} abs(x_n - x_{target}) + abs(y_n - y_{target}) & p = 0 \\ abs(x_n - x_{target}) + abs(y_n - y_{target}) * 2 & p \neq 0 \end{cases} \tag{4.4}$$

where (x_{n-2}, y_{n-2}) and (x_{n-1}, y_{n-1}) represent the two endpoints' coordinates of the previous channel, and (x_{n-1}, y_{n-1}) and (x_n, y_n) represent the coordinates of the two endpoints of the current channel. p represents the dot product of the current channel and the previous channel. From Equation 4.4, it is seen that the essential cost is the Manhattan Distance between the current node and the target node. Besides that, when the current channel is perpendicular to the previous channel, which means there is a turn, an additional cost equal to the Manhattan distance between the two points mentioned before will be added.

The reason for choosing the Manhattan Distance between the current node and the target node as the cost of turn is that when far away from the target, the extra cost is very high, and turning is discouraged. And when approaching the target, there may be some direction adjustments to make. If the cost of turning is still high, it is hard to find a feasible solution.

So when choosing the Manhattan Distance as the turn cost, the turn cost will decrease as it approaches the target, which makes it possible to find feasible solutions.

Through the above A^* algorithm, the program can set the required number of channels according to the situation of each connection, avoiding parameter redundancy and improving program operation efficiency.

4.3 Merging part of channels of different connections that have the same pins

Suppose there are several connections with the same start pin or end pin. In that case, sharing the channel directly connected to the same pin is advisable because it will reduce the number of total channels, reducing the number of parameters that Gurobi should calculate. It will speed up the program processing time. Besides that, the chip will also be easier to fabricate because fewer channels need to be printed.

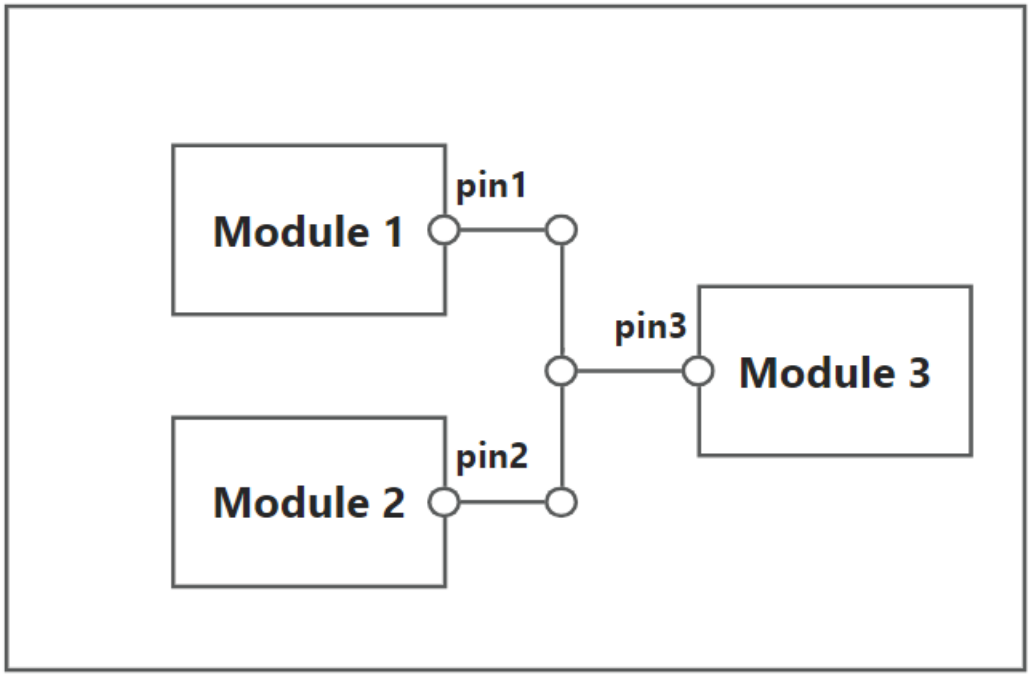


Figure 4.8: An example of a situation where fusion channels are required. pin_1 and pin_2 are both connected to pin_3, so they share the last channel directly connected to pin_3.

5 Calculation examples and analysis of experimental results

5.1 Calculation case design and experimental equipment setup

5.1.1 Test case without a constraint of pressure losses

This thesis uses a three-layer chip containing six components with five connection relationships as a test case to verify whether the primary channel planning function of the program is practical. The following tables introduce the chip size and module size in the chip of the test case and the connection relationship between each pin port.

Table 5.1: Specifications of the chip in the test case.

	chip
Device length(μm)	20000
Device width(μm)	15000
Device thickness(μm)	4000
Channel width(μm)	400
Module Height(μm)	100
Number of layers	3

Table 5.2: Parameters of modules of the DelayChannel type in the chip.

Module name	m1	m2	m3
Type	DelayChannel	DelayChannel	DelayChannel
Coordinate of the upper left endpoint(μm)	(5000,3000)	(5000,9500)	(1500,5500)
Width(μm)	6000	6000	6000
Height(μm)	4000	4000	4000
Rotation(degrees)	0	0	0
LayerId	0	0	1
Turning radius(μm)	400	400	400
Number of turnings	4	4	6

Table 5.3: Parameters of modules of the Filter type in the chip.

Module name	m4	m5
Type	Filter	Filter
Coordinate of the upper left endpoint(µm)	(10000,3500)	(10000,10000)
Width(µm)	4000	4000
Height(µm)	3000	3000
Rotation(degrees)	0	0
LayerId	1	1
Number of pillar rows	3	3
Number of pillar columns	4	4
Pillar radius(µm)	200	200

Table 5.4: Parameters of modules of the Chamber type in the chip.

Module name	m6
Type	Chamber
Coordinate of the upper left endpoint(µm)	(8000,6000)
Width(µm)	4000
Height(µm)	3000
Rotation(degrees)	0
LayerId	2

Table 5.5: Parameters of the pins in the chip.

Pin name	x(µm)	y(µm)	Module belongs to
pin1	6000	2000	m1
pin2	6000	2000	m2
pin3	4000	1500	m4
pin4	0	1500	m4
pin5	4000	1500	m5
pin6	0	1500	m5
pin7	0	2000	m3
pin8	6000	2000	m3
pin9	0	1500	m6
pin10	4000	1500	m6

Table 5.6: Connection relationship in the chip.

Connection	PinId	PinId
1	pin1	pin3
2	pin2	pin5
3	pin4	pin8
4	pin6	pin8
5	pin7	pin9

5.1.2 Test case with a constraint of pressure losses

This thesis uses a one-layer chip containing three components with two connection relationships as a test case to verify whether the channel planning function of the program is practical when considering the constraints of pressure losses. The following tables introduce the chip size and module size in the chip of the test case and the connection relationship between each pin port. Besides that, the following tables also introduce information about the constraint of pressure losses in this test case.

Table 5.7: Specifications of the chip in the test case.

	chip
Device length(µm)	20000
Device width(µm)	15000
Device thickness(µm)	4000
Channel width(µm)	400
Module Height(µm)	100
Number of layers	1

Table 5.8: Parameters of modules of the DelayChannel type in the chip.

Module name	m1
Type	DelayChannel
Coordinate of the upper left endpoint(µm)	(1500,5500)
Width(µm)	4000
Height(µm)	4000
Rotation(degrees)	0
LayerId	0
Turning radius(µm)	400
Number of turnings	4

Table 5.9: Parameters of modules of the Filter type in the chip.

Module name	m2
Type	Filter
Coordinate of the upper left endpoint(μm)	(8000,6000)
Width(μm)	4000
Height(μm)	3000
Rotation(degrees)	0
LayerId	0
Number of pillar rows	3
Number of pillar columns	4
Pillar radius(μm)	200

Table 5.10: Parameters of modules of the Chamber type in the chip.

Module name	m3
Type	Chamber
Coordinate of the upper left endpoint(μm)	(8000,11000)
Width(μm)	4000
Height(μm)	3000
Rotation(degrees)	0
LayerId	0

Table 5.11: Parameters of the pins in the chip.

Pin name	x(μm)	y(μm)	Module belongs to
pin1	0	2000	m1
pin2	4000	2000	m1
pin3	0	1500	m2
pin4	2000	3000	m2
pin5	2000	0	m3
pin6	4000	1500	m3

Table 5.12: Connection relationship in the chip.

Connection	PinId	PinId
1	pin2	pin3
2	pin4	pin5

Table 5.13: Information about the constraint of the pressure losses in the chip

Requirement of the pressure losses(Pa)	82737.12
μ(kg/m·s)	0.001
Density(kg/m^3)	997
Velocity(μm/s)	200
Tolerance for Deviation	10%
Input pin	p1
Output pin	p6

5.1.3 Experimental equipment setup

Table 5.14: Parameters of the various hardware of the experimental equipment

Processor	11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz 2.42 GHz
RAM	16.0 GB
System Type	64-bit operating system, x64-based processor

5.2 Experimental results and feasibility analysis

5.2.1 Experimental results of the test case without a constraint of pressure losses

The following pictures introduce in order the test case running environment, program processing time, the final output SVG format data, and the chip design diagram generated based on the output data.

```
Gurobi Optimizer version 9.1.2 build v9.1.2rc0 (win64)
Thread count: 4 physical cores, 8 logical processors, using up to 8 threads
Optimize a model with 14942 rows, 3660 columns and 49652 nonzeros
Model fingerprint: 0x1fd2fa6f
Model has 416 general constraints
Variable types: 832 continuous, 2828 integer (2674 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+05]
  Objective range  [0e+00, 0e+00]
  Bounds range     [1e+00, 2e+09]
  RHS range        [1e+00, 5e+01]
Warning: Model contains large bounds
         Consider reformulating model or setting NumericFocus parameter
         to avoid numerical issues.
Presolve removed 7805 rows and 969 columns
Presolve time: 0.37s
Presolved: 7137 rows, 2691 columns, 20257 nonzeros
Variable types: 870 continuous, 1821 integer (1785 binary)
```

Figure 5.1: Test case running environment. It can be seen that the program uses eight threads to calculate the test task.

```
Root relaxation: objective 0.000000e+00, 1704 iterations, 0.04 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0    0.00000    0   12          -    0.00000      -     -    0s
     0     0    0.00000    0   22          -    0.00000      -     -    0s
     0     0    0.00000    0    1          -    0.00000      -     -    0s
H    0     0                       0.0000000    0.00000  0.00%     -    1s
```

Figure 5.2: The processing time of the program. It can be seen that the program completes the solution within one second.

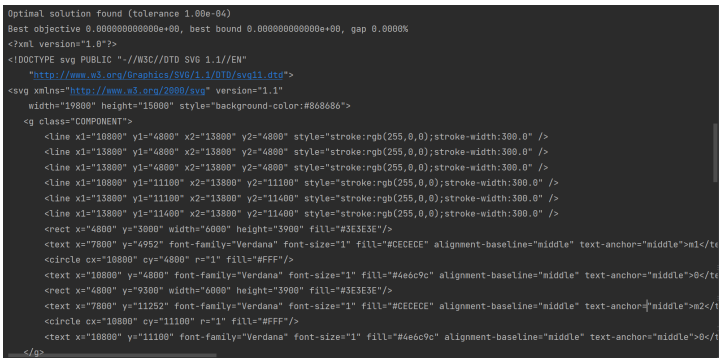


Figure 5.3: The final output SVG format data.

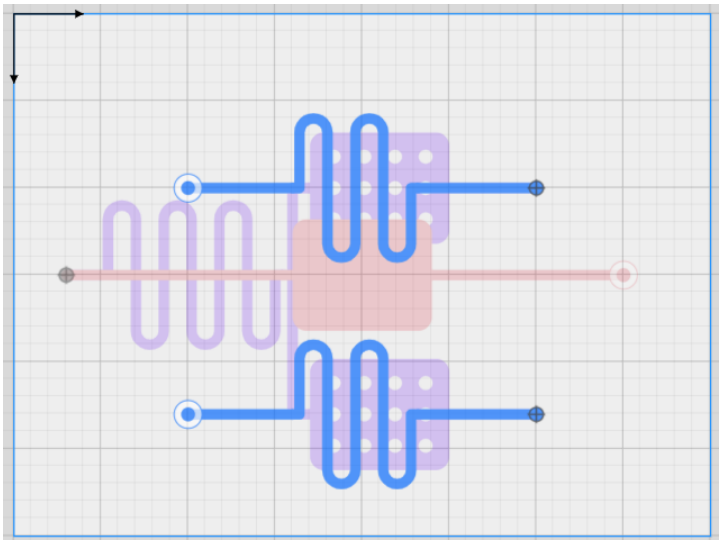


Figure 5.4: The chip design diagram of layer 0.

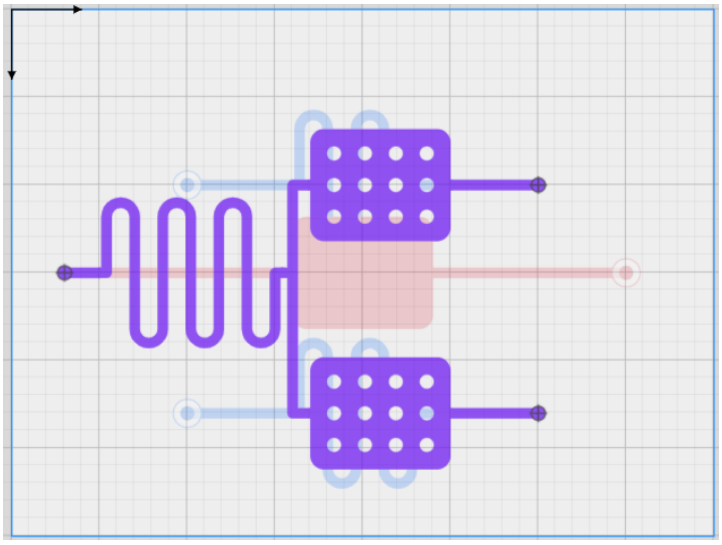


Figure 5.5: The chip design diagram of layer 1.

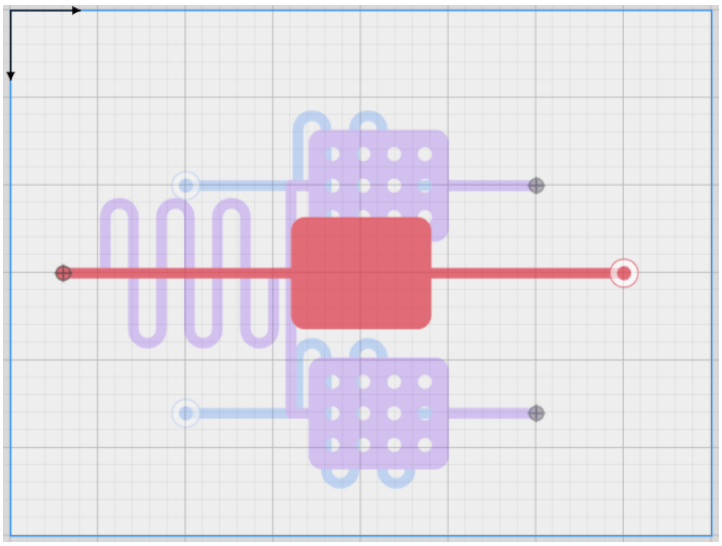


Figure 5.6: The chip design diagram of layer 2.

The data presented in the preceding figures illustrate that the program completes the channel layout design in under one second, a task that would require up to two minutes if performed manually. Furthermore, a comparison of the module and pin coordinates generated by the program against the pre-established coordinates reveals a marginal error of 0.2%, attributable to scaling effects discerned upon analysis.

5.2.2 Experimental results of the test case with a constraint of pressure losses

The following pictures introduce in order the test case running environment, program processing time, the final output SVG format data, and the chip design diagram generated based on the output data.

```
Academic license - for non-commercial use only - expires 2023-11-17
Gurobi Optimizer version 9.1.2 build v9.1.2rc0 (win64)
Thread count: 4 physical cores, 8 logical processors, using up to 8 threads
Optimize a model with 5153 rows, 1341 columns and 16958 nonzeros
Model fingerprint: 0x129b4e6b
Model has 2 quadratic constraints
Model has 156 general constraints
Variable types: 314 continuous, 1027 integer (948 binary)
```

Figure 5.7: Test case running environment. It can be seen that the program uses eight threads to calculate the test task.

```
Root relaxation: objective 0.000000e+00, 528 iterations, 0.02 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0    0.00000    0   16          -    0.00000      -     -    0s
     0     0    0.00000    0   71          -    0.00000      -     -    0s
H    0     0                       0.0000000    0.00000  0.00%     -    0s
     0     0    0.00000    0   10    0.00000    0.00000  0.00%     -    0s
```

Figure 5.8: The processing time of the program. It can be seen that the program completes the solution within one second.

```
Best objective 0.000000000000e+00, best bound 0.000000000000e+00, gap 0.0000%
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
   "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg xmlns="http://www.w3.org/2000/svg" version="1.1"
    width="20000" height="15000" style="background-color:#868686">
   <g class="COMPONENT">
      <line x1="5400" y1="7400" x2="5400" y2="7400" style="stroke:rgb(255,0,0);stroke-width:200.0" />
      <line x1="5400" y1="7400" x2="5400" y2="7400" style="stroke:rgb(255,0,0);stroke-width:200.0" />
      <line x1="5400" y1="7400" x2="5400" y2="7400" style="stroke:rgb(255,0,0);stroke-width:200.0" />
      <line x1="5400" y1="7400" x2="5400" y2="7400" style="stroke:rgb(255,0,0);stroke-width:200.0" />
      <line x1="5400" y1="7400" x2="5400" y2="7400" style="stroke:rgb(255,0,0);stroke-width:200.0" />
      <line x1="5400" y1="7400" x2="8000" y2="7400" style="stroke:rgb(255,0,0);stroke-width:200.0" />
      <line x1="10000" y1="9000" x2="10000" y2="9000" style="stroke:rgb(255,0,0);stroke-width:200.0" />
      <line x1="10000" y1="9000" x2="10000" y2="9000" style="stroke:rgb(255,0,0);stroke-width:200.0" />
      <line x1="10000" y1="9000" x2="10000" y2="9000" style="stroke:rgb(255,0,0);stroke-width:200.0" />
      <line x1="10000" y1="9000" x2="10000" y2="11000" style="stroke:rgb(255,0,0);stroke-width:200.0" />
      <line x1="10000" y1="11000" x2="10000" y2="11000" style="stroke:rgb(255,0,0);stroke-width:200.0" />
      <line x1="10000" y1="11000" x2="10000" y2="11000" style="stroke:rgb(255,0,0);stroke-width:200.0" />
      <rect x="1400" y="5400" width="4000" height="4000" fill="#3E3E3E"/>
      <text x="3400" y="7402" font-family="Verdana" font-size="1" fill="#CECECE" alignment-baseline="middle" text-anchor="middle">m1</t
      <circle cx="1400" cy="7400" r="1" fill="#FFF"/>
      <text x="1400" y="7400" font-family="Verdana" font-size="1" fill="#4e6c9c" alignment-baseline="middle" text-anchor="middle">0</te
```

Figure 5.9: The final output SVG format data.

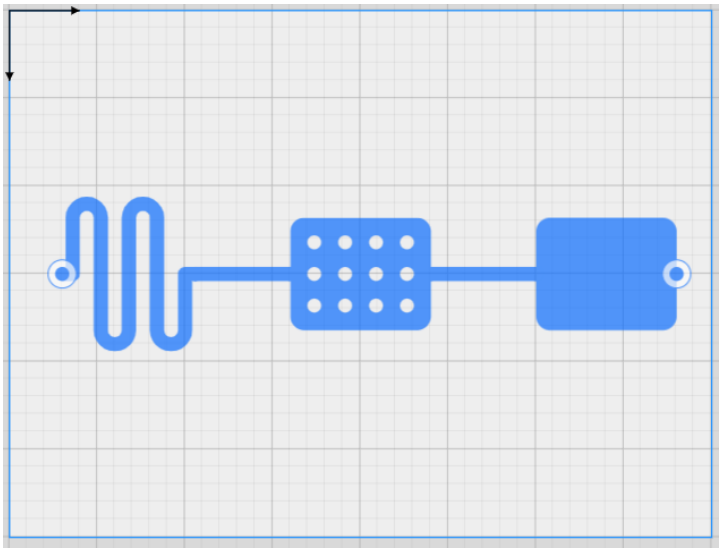


Figure 5.10: The chip design diagram of layer 0.

From the above figures, we can see that the program finishes the design of the channels' layout within one second, while the same task will take five minutes to complete manually because manually calculating pressure loss is a time-consuming process. And it has been verified that the error in the pressure loss of the channels generated by the program is only 4% compared with the pressure loss expected by the user.

5.2.3 Feasibility analysis

Developing an automated channel design program for microfluidic chips represents a promising technological innovation. This innovation provides a crucial tool for addressing microfluidic chip channel design's intricacies and labor intensity. Here is an analysis of the feasibility of this program:

First and foremost, this program significantly enhances design efficiency. Traditional microfluidic chip channel design often consumes substantial time and human resources. In contrast, the automated design process of this program expedites the generation of channel layouts, thereby substantially reducing the design cycle.

Furthermore, the automated design program mitigates the risk of design errors. Human-designed channels are susceptible to errors, whereas the computerized program helps eliminate these errors, thereby improving the reliability and performance of microfluidic chips.

In addition, this program offers versatility. It can generate various channel layouts tailored to specific needs, making it applicable to multiple microfluidic experiments and applications, from biomedical to chemical analysis.

Most importantly, this program provides visualization and simulation capabilities, enabling users to preview and assess the performance of microfluidic chips before actual manufacturing. This feature saves on experimental costs and time.

In summary, the automated microfluidic chip channel design program demonstrates significant feasibility, with the potential to enhance design efficiency, reduce error risk, introduce versatility, and provide visualization and simulation capabilities.

5.3 Shortcomings and future improvement directions

After tests, it is verified that this program can generate channel layouts of microfluidic chips according to the placement information of modules, the connection relationship between modules, and the requirement of pressure loss between two modules. Although with this program, users can design microfluidic chips more efficiently, some defects still exist. The following will introduce the two main problems currently existing in the program and the corresponding solutions in the future.

First, there may be position errors when scaling and restoring the chip. For example, when the original upper left endpoint's coordinate of one module in the chip is (1803, 1803), and the scaling factor is 10, the coordinate will become(180, 180) after downscaling. After restoring, the coordinate will become (1800, 1800) instead of (1803, 1803), which means there is an error of 3 microns in the x-axis and y-axis coordinates. To overcome this shortcoming in the future, the program can store the exact coordinate bits omitted during downscaling and add these ignored bits during the recovery phase.

Second, currently, this program does not support the calculation of pressure loss constraints for channel mergers because when fluids are mixed, the pressure changes depend on many factors, such as the fluid mixing method, the velocity of the liquid during mixing, the shape of the mixing container, etc. In the future, empirical formulas for calculating pressure changes in specific mixing methods of specific fluids can be introduced to solve the problem of pressure loss when specific liquids merge in channels. For example, we can use the continuity equation and the Bernoulli equation to calculate the pressure loss when two fluid streams merge into one channel. For incompressible fluids, the continuity equation assures that the flow rate is conserved across a junction. And then, we can apply the Bernoulli equation to points just before the merging and in the merged stream, which can give a relationship between velocities and pressure.

In the future, as these two problems are solved, the application scenarios of this program will become more extensive.

List of Figures

List of Tables

Bibliography

[1] S. Battat, D. A. Weitz, and G. M. Whitesides. "An outlook on microfluidics: the promise and the challenge". In: *Lab Chip* (2022).

[2] A. K. Au, W. Lee, and A. Folch. "Mail-order microfluidics: evaluation of stereolithography for the production of microfluidic devices". In: *Lab on a Chip* 14.7 (2014), pp. 1294–1301.

[3] B. Carnero, C. Bao-Varela, A. I. Gómez-Varela, E. Álvarez, and M. T. Flores-Arias. "Microfluidic devices manufacturing with a stereolithographic printer for biological applications". In: *Materials Science and Engineering: C* 129 (2021), p. 112388.

[4] D. J. Beebe, G. A. Mensing, and G. M. Walker. "Physics and Applications of Microfluidics in Biology". In: *Annual Review of Biomedical Engineering* (2002).

[5] K. G. Lee, K. J. Park, S. Seok, S. Shin, J. Y. Park, Y. S. Heo, S. J. Lee, T. J. Lee, et al. "3D printed modules for integrated microfluidic devices". In: *Rsc Advances* (2014).

[6] C. Chen, B. T. Mehl, A. S. Munshi, A. D. Townsend, D. M. Spence, and R. S. Martin. "3D-printed microfluidic devices: fabrication, advantages and limitations—a mini review". In: *Anal. Methods* (2016).

[7] L. J. Y. Ong, A. Islam, R. DasGupta, N. G. Iyer, H. L. Leo, and Y.-C. Toh. "A 3D printed microfluidic perfusion device for multicellular spheroid cultures". In: *Biofabrication* (2017).

[8] P. Pattanayak, S. K. Singh, M. Gulati, S. Vishwas, B. Kapoor, D. K. Chellappan, K. Anand, G. Gupta, N. K. Jha, P. K. Gupta, et al. "Microfluidic chips: recent advances, critical strategies in design, applications and future perspectives". In: *Microfluidics and Nanofluidics* (2021).

[9] T.-M. Tseng, M. Li, D. N. Freitas, T. McAuley, B. Li, T.-Y. Ho, I. E. Araci, and U. Schlichtmann. "Columba 2.0: A co-layout synthesis tool for continuous-flow microfluidic biochips". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2017).

[10] T.-M. Tseng, M. Li, D. N. Freitas, A. Mongersun, I. E. Araci, T.-Y. Ho, and U. Schlichtmann. "Columba S: A scalable co-layout design automation tool for microfluidic large-scale integration". In: *Proceedings of the 55th Annual Design Automation Conference*. 2018.

[11] FLUI3D. Available online: `https://www.flui3d.org/`.

[12] S. Wu. "Microfluidic Design Automation based on Deep Reinforcement Learning in Parameterized Action Space". en. MA thesis. Technische Universität München, 2023.

[13] S. C. Brailsford, C. N. Potts, and B. M. Smith. "Constraint satisfaction problems: Algorithms and applications". In: *European Journal of Operational Research* (1999).

[14] F. Rossi, P. Van Beek, and T. Walsh. *Handbook of constraint programming*. Elsevier, 2006.

[15] S. Russell and P. Norvig. *Artificial Intelligence: a Modern Approach, EBook, Global Edition : A Modern Approach*. Pearson Education, Limited, 2016.

[16] M. Pirlot. "General local search methods". In: *European Journal of Operational Research* 92.3 (1996), pp. 493–511. ISSN: 0377-2217. DOI: https://doi.org/10.1016/0377-2217(96)00007-0. URL: https://www.sciencedirect.com/science/article/pii/0377221796000070.

[17] R. E. Bellman and S. E. Dreyfus. *Applied dynamic programming*. Vol. 2050. Princeton university press, 2015.

[18] A. Abdelrazek, S. Kazi, O. A. Alawi, N. Yusoff, S. Oon, and H. Ali. "Heat transfer and pressure drop investigation through pipe with different shapes using different types of nanofluids". In: *Journal of Thermal Analysis and Calorimetry* 139 (July 2019). DOI: 10.1007/s10973-019-08562-5.

[19] T. M. Squires and S. R. Quake. "Microfluidics: Fluid physics at the nanoliter scale". In: *Reviews of Modern Physics* 77.3 (Oct. 2005), pp. 977–1026. DOI: 10.1103/RevModPhys.77.977.

[20] G. M. Jones, B. E. Bosserman, G. Tchobanoglous, et al. *Pumping station design*. Gulf Professional Publishing, 2006.

[21] G. W. Howell, T. M. Weathers, and T. S. G. R. B. CA. *Aerospace fluid component designers' handbook, volume II, Revision D*. Tech. rep. Report No. RPL-TDR-64-25, TRW Systems Group, One Space Park, Redondo Beach ..., 1970.