RABER: Reliability-Aware Bayesian-Optimization-based Control Layer Escape Routing for Flow-based Microfluidics

Siyuan Liang[1], Rongliang Fu[1], Mengchu Li[2], Tsun-Ming Tseng[2], Ulf Schlichtamnn[2], Tsung-Yi Ho[1]
[1]The Chinese University of Hong Kong, [2]Technical University of Munich
[1]{syliang22, rlfu, tyho}@cse.cuhk.edu.hk, [2]{mengchu.li, tsun-ming.tseng, ulf.schlichtmann}@tum.de

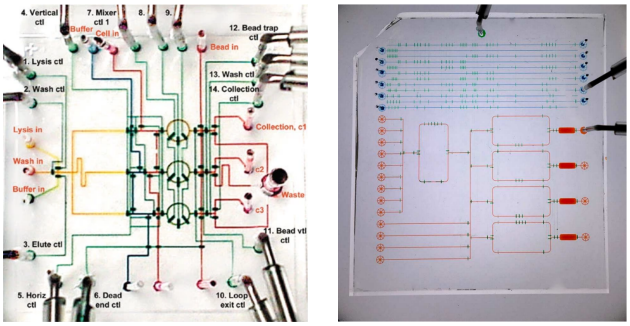
ABSTRACT

After decades of development, flow-based microfluidic biochips have become one of the most promising platforms for biochemical experiments. Control ports, which are remarkably area-consuming punch holes, are interfaces to external pneumatic controllers. To prevent the inserted outer catheters from hindering microscopic observation during experiments, control ports are placed on chip boundaries in practice. In this paper, we propose a practical and novel control layer escape routing methodology, which efficiently connects microvalves to user-specified boundaries. Particularly, the proposed methodology groups certain microvalves, and constructs a tree to connect them with the same control port, which is regarded as the root of the tree. Clustering more microvalves into the same group can reduce the usage of control ports, but will lead to more intensive connections among microvalves, which becomes larger obstacles for the routing of other microvalves outside the group, thereby reducing the routability. To derive an optimized tradeoff between the control port usage and the routability, we adapt a hierarchical clustering algorithm with a dynamically changing threshold that ascertains the closeness of the microvalves. We also adopt the Bayesian optimization (BO) to determine the optimized routing order for better routing results. Additionally, we propose a fault-tolerant structure as an option for users, which only occupies little area around control channels, and significantly improves the reliability against blockage defects. Experimental results demonstrate that the proposed methodology can efficiently connect all microvalves to user-specified boundaries, significantly reduce control port usage, shorten control channels, and improve reliability compared to baseline methods.

1 INTRODUCTION

During the past few decades, flow-based microfluidic biochips have made huge progress and become one of the most promising platforms for biochemical experiments [1, 2]. These coin-sized chips are able to incorporate many miniaturized devices and carry out complex operations that are traditionally conducted in cumbersome laboratory instruments [3]. The miniaturization brings many advantages, including low consumption of reagents, decreased manufacturing costs, and the great potential to incorporate microcontrollers for automated control of experiments [4].

Fluid manipulations on flow-based biochips rely on the switching of microvalves, which are connected via control channels to control ports to receive pressure from external pneumatic controllers. As shown in Figure 1(a), to prevent catheters from hindering microscopic observation, control ports are commonly placed on boundaries. In this case, we need to connect all microvalves in the center of the chip to the boundaries, i.e. conduct escape routing for the microvalves. In particular, some applications require valves to be connected to certain boundaries, such as the design shown in Figure 1 (b), which supplies pressure to all valves via a control multiplexer on the top boundary of the chip. This further requires the escape routing method to support routing to user-specified boundaries. Moreover, the control port is remarkably area-consuming. As the drilling process to fabricate such a port will significantly increase the stress around, a keep-out area with a radius of 2 mm [5] is needed around each control port to prevent the chip from collapsing during fabrication. To reduce chip area, it is highly desired if some microvalves can be connected together to share the same control port during the escape routing. This shared control port configuration should be implemented in a manner that prevents the induction of any control errors. Additionally, as the size of chips increases, the total length of control channels will increase, which leads to higher risks of blockage [6, 7]. Thus, fault-tolerant arrangements are also expected in the escape routing method.



(a) (b)

Figure 1: Photos of microfluidic biochips. (a) A nucleic acid processor [8]. (b) A ChIP 4-IP application [9].

Early routing methods for flow-based biochips [10, 11] cannot guarantee to place the control ports on chip boundaries. In [9] and [12], Tseng et al. and Weng et al. proposed two different escape routing methods that can efficiently route control channels from microvalves to chip boundaries. However, both methods directly follow the settings in mature PCB escape routing [13, 14], assuming that every microvalve needs to be connected to an individual control port, and ignore the fact that certain microvalves can actually share the pressure from the same control port. In this case, these methods have excessive control port usage. Additionally, control channels, due to their smaller features compared to flow channels, are more prone to blockage defects [6, 15]. Given that the likelihood

ICCAD '24, October 27–31, 2024, New York, NY, USA

ACM ISBN 979-8-4007-1077-3/24/10
https://doi.org/10.1145/3676536.3676744

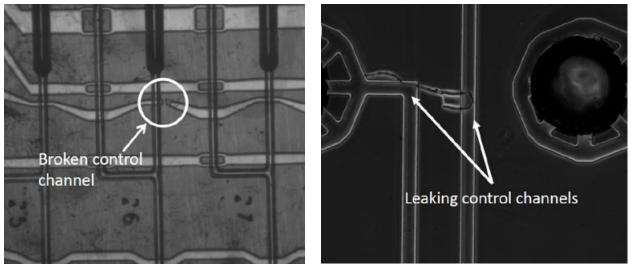


Figure 2: Common manufacturing defects of channels [16]. (a) Blockage. (b) Leakage.

of blockage is exponentially linked to the total length of channels, the risk of blockage escalates significantly as chip size expands[7]. Previous routing methods for flow-based microfluidic biochips have overlooked the reliability of routing results, and lack preventive measures against the escalating risk of blockage.

In this paper, we propose a practical and novel control layer escape routing methodology, which efficiently and reliably connect microvalves to a reduced number of control ports on user-specified boundaries. Particularly, we adapt a hierarchical clustering algorithm to group microvalves that have compatible operation schedules and that are physically close enough, and let them share the same control port. Microvalves within the same group, together with the corresponding control port, will be considered as terminals of a net. For each net, a tree will be constructed to connect its terminals, with the control port serving as the root of the tree. Considering the relibiality, we introduce keep-out areas around routed nets, and regard them as obstacles in the later routing of other nets.

In the hierarchical clustering algorithm, a threshold is used to determine if the microvalves are close enough. By setting the threshold to larger values, more operation-compatible microvalves will be clustered into the same group, requiring a larger tree with longer edges to connect them. This tree will potentially become a larger obstacle for the routing of other nets, therefore leading to reduced routability. To derive an optimized tradeoff between the usage of control ports and routability, we initially set the threshold to a large enough value, and iteratively adjust it. If the routing fails for the current grouping result, the threshold will be decreased to assign operation-compatible microvalves that are even closer to each other into the same group, which reduces the size of the tree, thereby leading to higher routability. This iterative adjustment continues until the routing succeeds. To ensure routing efficiency, we greedily route one net at a time, and adopt Bayesian optimization (BO) to determine the routing order.

For the routing of each net, given the terminals of this net and the obstacles, we construct an obstacle-avoiding rectilinear Steiner minimum tree (OARSMT) to build the shortest connections among microvalves. A variant of the Lee algorithm is proposed to connect the OARSMT to user-specified boundaries, where a control port will be placed as the root of the tree. In addition, we also propose a novel fault-tolerant structure as an option to enhance the reliability of the routing results against blockage. It is worth mentioning that

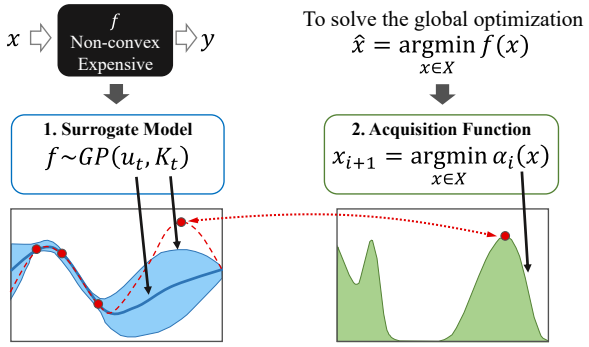


Figure 3: Schematic of Bayesian optimization.

the size of the keep-out areas and the number of the fault-tolerant structures are all adjustable.

Experimental results demonstrate that the proposed methodology can efficiently perform escape routing, reduce control port usage, shorten control channels, and improve the reliability of the chip against manufacturing defects compared to baseline methods.

2 BACKGROUND

2.1 Common Control Channel Defects

Typically, control channels have a smaller feature size than flow channels, which makes them more prone to defects. Specifically, there are two major types of channel defects [7, 16]:

2.1.1 Blockage. The accidentally broken or clogged channel segments. Figure 2(a) shows a blockage in a control channel. A blockage is usually caused by environmental particles and an imperfect silicon wafer mold [16]. For control channels, blockage prevents pressure from reaching the microvalves, leading to losing control over the microvalves along the channel.

2.1.2 Leakage. The unexpected connections between channels. Figure 2(b) shows a leakage between two control channels. For leaky control channels, pressure may leak from one channel to the other, resulting in the unanticipated closure of microvalves.

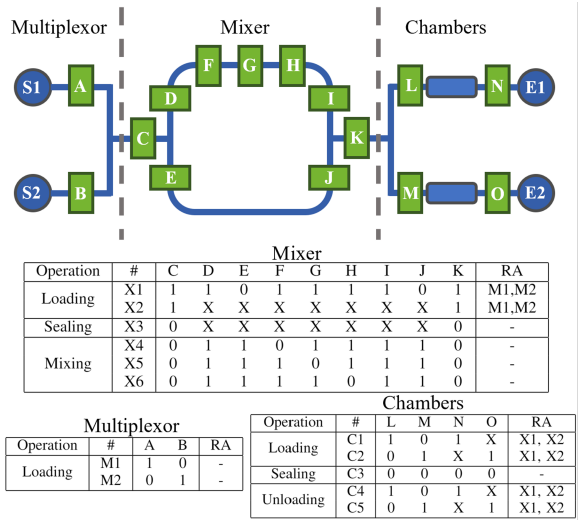
According to [16], the probability of blockage escalates exponentially as the length of the channel increases, while the probability of leakage increases as the length of parallel channels increases or the distance between two neighboring channels decreases.

2.2 Bayesian Optimization

As a query-efficient black-box optimization technique, Bayesian optimization (BO) aims to rapidly find an input $\hat{x}$ that minimizes a function $f(x)$ over some domain $\boldsymbol{X}$ consisting of finite lower and upper bounds on each variable:

$$\hat{x} = \underset{x \in \boldsymbol{X}}{\operatorname{argmin}} f(x) \tag{1}$$

The objective function $f(x)$ often has no analytical expression or derivative, the evaluation of $f(x)$ is restricted to sampling at a point x and obtaining a possibly noisy response. The feasible domain $\boldsymbol{X}$ is typically a hyper-rectangle $\left\{x_i \in \mathbb{R}^d : \ a_i \leq x_i \leq b_i\right\}$ or a d-dimensional simplex $\left\{x_i \in \mathbb{R}^d : \ \sum_i x_i = 1\right\}$.



Mixer

Operation	#	C	D	E	F	G	H	I	J	K	RA
Loading	X1	1	1	0	1	1	1	1	0	1	M1,M2
	X2	1	X	X	X	X	X	X	X	1	M1,M2
Sealing	X3	0	X	X	X	X	X	X	X	0	-
Mixing	X4	0	1	1	0	1	1	1	1	0	-
	X5	0	1	1	1	0	1	1	1	0	-
	X6	0	1	1	1	1	0	1	1	0	-

Multiplexor

Operation	#	A	B	RA
Loading	M1	1	0	-
	M2	0	1	-

Chambers

Operation	#	L	M	N	O	RA
Loading	C1	1	0	1	X	X1, X2
	C2	0	1	X	1	X1, X2
Sealing	C3	0	0	0	0	-
Unloading	C4	1	0	1	X	X1, X2
	C5	0	1	X	1	X1, X2

Figure 4: An example design with three parts and their corresponding unit operations [10].

As shown in Figure 3, there are two critical components in BO: (i) the surrogate model used to approximate the objective function, typically a Gaussian process (GP) [17], and (ii) the acquisition function that directs sampling to areas where an improvement over the current best observation f' is promising. A popular acquisition function is expected improvement (EI) [18], which is given by the expected value of the utility function $u(x) = \max(0, f' - f(x))$ under the GP predictive distribution for f. BO usually iterates between fitting a model and gathering additional data. Specifically, the surrogate model $p(y|x, \mathcal{D}_t)$ is initialized by a small set $\mathcal{D}_t$ of t samples from the domain $\boldsymbol{X}$. After that, new observed points $\{x_{t+1}, x_{t+2}, ..., x_T\}$ within the domain $\boldsymbol{X}$ are sequentially selected by optimizing an acquisition function S which uses the current model $p(y|x, \mathcal{D}_i)$ as a cheap surrogate for the expensive objective f. The point selection, guided by the acquisition function, empowers BO to consistently identify superior points, rather than randomly selecting points without any differentiation. This significantly boosts the efficiency. Each observed point x_{i+1} will be evaluated by the function f to produce an observed result $y_{i+1} = f(x_{i+1})$. The result (x_{i+1}, y_{i+1}) is appended into the historical set $\mathcal{D}_i$ to obtain the new set $\mathcal{D}_{i+1}$, which is used to update the surrogate model $p(y|x, \mathcal{D}_{i+1})$ for generating the next observation point. Finally, the point $\hat{x}$ can be selected from $\mathcal{D}_T$, so that the objective function f approaches or reaches the global optimum at the point $\hat{x}$.

2.3 Problem Formulation

In this paper, we formulate the microfluidic escape routing problem as follows:

Input : *User-Specified Boundaries, Location of Microvalves, Unit Operations of Devices.*

Output : *Routing Result of the Control Layer.*

Objective : 1. *Minimize Control Port Usage*;
2. *Minimize Total Length of Control Channels*;
3. *Maxmize the Reliability.*

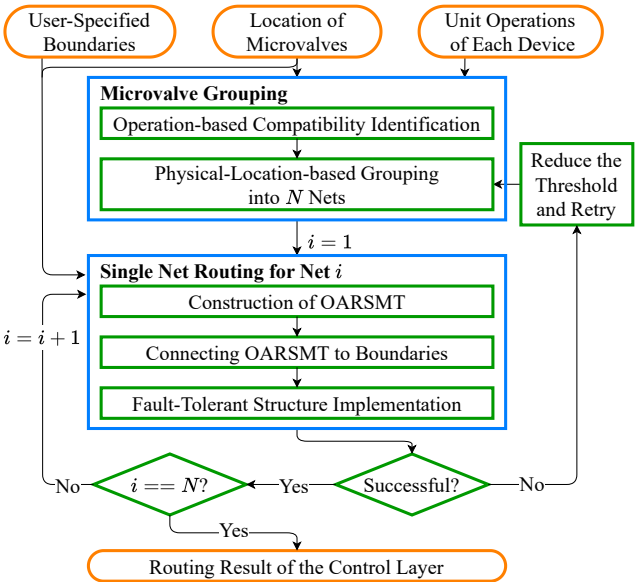


Figure 5: Schematic of the proposed flow using the default routing order.

We take the user-specified boundaries, the location of microvalves and the unit operations of devices as the input, and produce the routing outcome of the control layer as the output. Here, the unit operation refers to the most basic operation of a device. For example, as shown in Figure. 4, "Loading" is a unit operation of the Mixer. Our escape routing result is guided by three prioritized objectives. The primary objective is to minimize the usage of control ports due to their significant area consumption. When the number of control ports is the same, we then wish to complete the routing with shorter control channels, making the reduction of the total length of control channels our secondary objective. Finally, with minimized control port usage and control channel length, we aim to enhance the design's reliability against manufacturing defects.

3 RELIABILITY-AWARE CONTROL LAYER ESCAPE ROUTING METHODOLOGY

In this section, we introduce the proposed reliability-aware control layer escape routing methodology.

3.1 Initial Routing Flow Using the Default Routing Order

The flow of the proposed reliability-aware control layer escape routing methodology is shown in Figure 5. The flow consists of two parts: microvalve grouping, and single net routing.

With the given unit operations of devices, we can conduct the operation-based compatibility identification to determine what microvalves can share the same control port without any errors. Considering that some compatible microvalves might be distant from each other, directly connecting them together could lead to substantial obstacles, potentially impeding the successful routing of other nets. To avoid this, we introduce a distance threshold between valves to ensure valves that are assigned to the same group are

close enough to one another. The threshold is initially configured to a sufficiently large value that permits all compatible microvalves to be included in the same net.

After the nets are determined, we repeatedly carry out single net routing, utilizing the default routing order that is organized based on the serial number of the nets. The process of each single net routing has three stages: the construction of OARSMT to connect all microvalves in the net, the connection of the OARSMT to user-specified boundaries of the design, and an optional fault-tolerant structure implementation to enhance the reliability. When all nets are successfully routed, the result will be produced. If the routing of any net fails, we will reduce the threshold, revert to the microvalve grouping stage, and start again. By reducing the threshold, compatible microvalves that are not physically close enough will no longer be grouped into the same net. This helps avoid long connections, which are huge obstacles potentially hindering the routing of other nets. In this case, the routing of all nets will become more likely to succeed.

By utilizing the proposed flow, we can minimize the use of control ports to complete the escape routing of all microvalves.

3.2 Microvalve Grouping

To enable as many microvalves as possible to share the same control port without causing control errors, we propose a 2-staged approach to properly group microvalves.

3.2.1 Operation-based Compatibility Identification. Since it is hard to get the complete control logic table of a design [10], we choose to identify compatibility based on unit operations of each device in the design to avoid causing errors: As shown in Figure. 4, the example design has three parts. We sort out all the unit operations of each device and the corresponding control logic. For the control logic of a specific operation, "1" and "0" stand for the corresponding microvalve being depressurized and pressurized, "X" represents "don't care", which means the status of the microvalve does not affect the operation. The content in column "RA" shows unit operations of other parts that the specific unit operation needs to be coordinated with. For example, unit operation "X1" is "Loading to the upper half of the mixer", and according to the content in its "RA" column, it needs to coordinate with "M1, M2" to form "Load sample S1/S2 to the upper half of the mixer" as chip-level operations. After all chip-level operations are found, we are able to know the status of every microvalve in different chip-level operations, and the compatibility among microvalves can then be identified without loss of functionality.

3.2.2 Physical-Location-based Grouping. Since the identified operational compatible microvalves may be physically far from each other, it might cause channel length overhead if we directly include all compatible microvalves in a net for the later routing process. Starting with microvalves that are closest to each other, we apply the hierarchical clustering algorithm to gradually cluster microvalves based on their average distances. The binary tree in Figure 6 shows the example results: each leaf represents a microvalve, every node represents a cluster formed by its child nodes or leaves, with the value at each node indicating the average distance among microvalves within the cluster. Denoted by the red line in Figure 6(a), we then refer to a user-specified threshold denoting the maximum distance, at which child nodes or leaves can be considered to

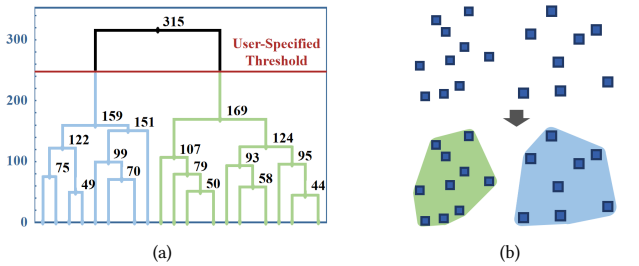


Figure 6: Schematics of hierarchical clustering. (a) An example binary tree of hierarchical clustering result. (b) The corresponding clustering result based on user-specified threshold.

be close enough to be included within the same cluster. Clusters represented by the highest nodes below the red line will be regarded as microvalve groups, as shown in Figure 6(b). Microvalves within the same group will be connected by a net and will share the same control port.

3.3 Single Net Routing

For the routing of a single net, in order to maintain safe distances between other nets, we set a user-adjustable keep-out zone with a default width of 200μm around the components of other nets, and regard the keep-out zone together with other routed nets as obstacles. An example is shown in Figure 7(a), the blue polygons represent the obstacles, the red dots stand for the microvalves that need to be connected by the net, and the dashed lines are the boundaries of the design. We then construct an OARSMT generation algorithm to connect all microvalves within the same group, and adapt the Lee algorithm to connect the OARSMT to user-specified boundaries. At the end, we propose a fault-tolerant structure as an option for users.

3.3.1 Construction of the OARSMT. As shown in Figure 7(b)-(d), The construction of OARSMT consists of three steps: construction of the obstacle avoiding spanning graph (OASG), construction of the minimum terminal spanning tree (MTST), and construction of the OARSMT.

To start with, we generate an OASG according to the given obstacles and microvalves. As shown by red dots and blue dots respectively in Figure 7(a), microvalves and corners of obstacles are regarded as vertices. At each vertex, we conduct octant division to divide the entire design into 8 sectors with an angle of 45 degrees. To avoid taking into account too many edges that have little benefit on the results, in each sector, we only connect the vertex to the nearest vertex that is not blocked by obstacles, and take the Manhattan distance between them as the weight of the corresponding edge. Since some vertices of the same obstacles have the same x-coordinate or y-coordinate, which can not be handled by the octant division, we manually add all edges of obstacles into the edges of OASG.

After deriving the OASG, we need to extract an MTST, which contains only edges with the smallest sum of weights in the OASG that can connect all microvalves. As shown in Figure 7(b), the OASG contains many edges, which form multiple paths between microvalves. We first apply Dijkstra's algorithm [19] to find the

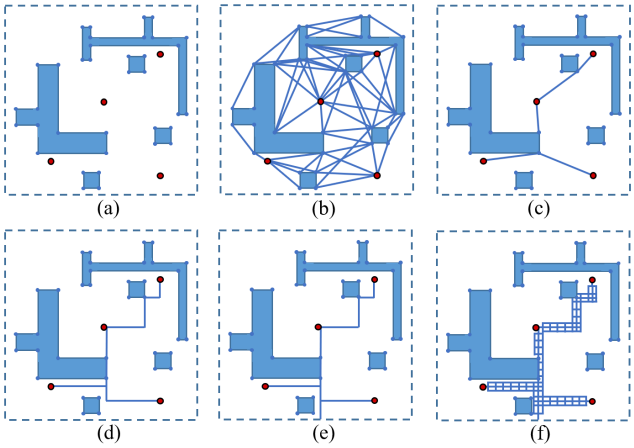


Figure 7: Schematics of the single net escape routing process. (a) Construction of obstacles. (b) OASG. (c) MTST. (d) OARSMT. (e) Connecting the OARSMT to the nearest boundary. (d) Implementation of fault-tolerant structures.

shortest paths between every two microvalves and the corresponding edges that form these paths. Then we can derive a simplified graph that only contains the edges in the shortest paths found by Dijkstra's algorithm. Finally, we apply Prim's algorithm [20] to the simplified graph, and derive the MTST.

Based on the MTST, we only need to conduct rectilinearization to turn each edge into a horizontal wiring and a vertical wiring, and derive the final result. For each edge in the MTST, since it is also an edge of the OASG, there are no obstacles within the rectangle formed by the maximum and minimum x, y coordinates of the two end vertices. In this case, we directly start from the vertex closest to the bottom, conduct horizontal routing first, and then conduct vertical routing. After handling all edges in the MTST, the OARSMT is eventually derived.

3.3.2 Connecting OARSMT to User-Specified Boundaries. After connecting all microvalves within the same group to each other, we propose a variant of the Lee algorithm [21] to connect the OARSMT to user-specified boundaries. In the proposed variant, we regard the whole OARSMT as the source and user-specified boundaries as the target. We conduct Lee-like propagation from the source, and record where we reach at each propagation. Once the target is reached, we stop the propagation, backtrack along the path with the steepest propagation gradient, and construct the connection accordingly. With all four boundaries specified as connectable, the example OARSMT shown in Figure 7(d) is successfully connected to the nearest one of the specified boundaries as shown in Figure 7(e).

3.3.3 Fault-Tolerant Structure Implementation. As the size of the design expands, the total length of control channels correspondingly increases. Given that the likelihood of blockage is exponentially associated with the total length of channels [7], larger designs inherently carry a significantly heightened risk of blockage. To improve channels' reliability against blockage, we propose a novel fault-tolerant structure as an option for users.

As shown in Figure 8(a), two vertical parallel channels are placed at a distance w to the left and right of the original channel, and horizontal channels are repeatedly placed at a distance of d to connect

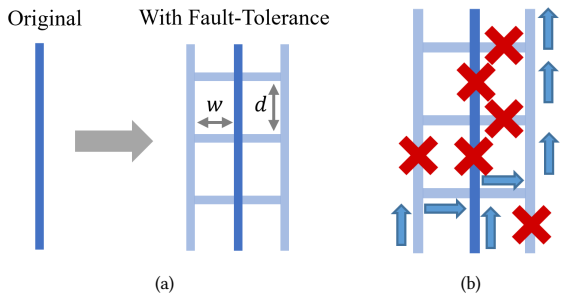


Figure 8: Schematics of the fault-tolerant structure. (a) Implementation to a single channel. (b) Transporting pressure normally even under serious faults.

these parallel channels. w and d are user-adjustable parameters that can adapt the structure to different user needs. Figure 8(b) represents an extreme scenario that demonstrates the remarkable effectiveness of the proposed structure. The red crosses indicate blockages at various points within the structure. Despite these significant faults, the structure still manages to transport pressure along the path depicted by the blue arrows, thereby demonstrating the robustness of the proposed fault-tolerant structure.

Whether to implement the proposed fault-tolerant structure in designs is decided by users. If the user thinks some connections are very important and wishes to ensure their functionality, they can enable this function. As an example, for the design shown in Figure 7(e), the implementation of the proposed structure is shown in Figure 7(f), in which channel segments with only one side being free area can just get half of the proposed structure implemented, and those with both sides being free area can get the complete fault-tolerant structure implemented.

4 BAYESIAN OPTIMIZATION OF ROUTING ORDER

For efficiency reasons, the proposed method greedily routes net by net. However, the nets routed earlier will become obstacles for the nets routed later, causing detours to varying degrees. Hence, the routing order can significantly affect the routing results. Since it is impractical to traverse all possible orders to find the best one, we need a more efficient strategy to derive a good order based on just a limited number of attempts. Inspired by [22], we propose to adopt BO to optimize the routing order, and thus to achieve good routing results.

4.1 Overview of the Improved Routing Flow

The improved routing flow is shown in Figure 9. Compared to the original routing flow shown in Figure 5, the improved flow has an additional part of BO-based net reordering. The BO-based net reordering aims to efficiently find the optimized routing order, which leads to better routing results. Specifically, the optimized routing order can result in a reduced total length of control channels. It can even yield a valid routing result by utilizing a grouping result that was deemed unroutable in the original flow, thereby reducing the usage of control ports.

A set D is utilized to record all the routing orders o and their corresponding costs f explored during the BO process. For a given

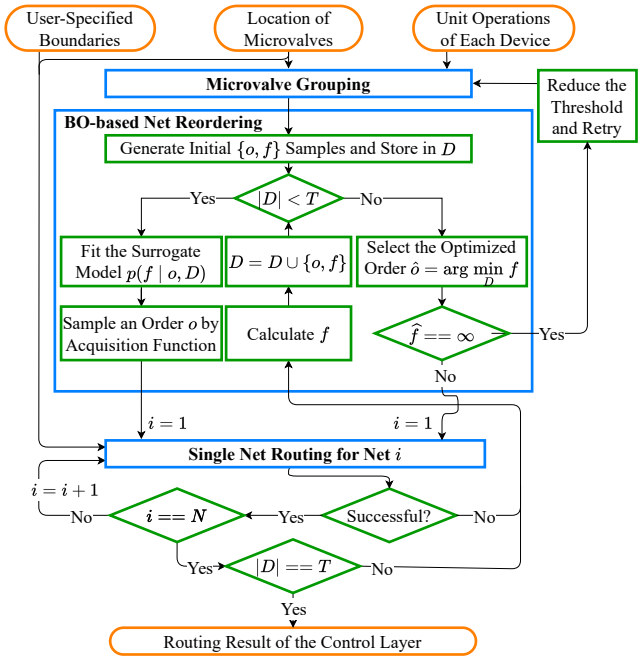


Figure 9: Schematic of the proposed flow using the optimized routing order derived by BO.

microvalve grouping result, we first randomly generate a set of routing orders, conduct routing following these orders to derive their costs, and record them in D as the initial samples. When the size of D is smaller than the manually-set limit T, we will fit the surrogate model, sample a new order, and conduct routing to derive the cost, and update D. Once $|D| = T$, we will find the optimized order with the smallest cost. If the cost of the optimized order indicates this order is unroutable, we will reduce the threshold for grouping and start over. Otherwise, the routing result derived by following the optimized order will be output.

4.2 Problem Model

Given a routing order, we can carry out our proposed routing method net by net in terms of this order and obtain the final routing results. Since each net is routed only once in the routing phase, the routing order of N nets can be viewed as a permutation of these net indexes. That is, each routing order is defined as a bijective mapping $o : \{1, 2, \cdots, N\} \mapsto \{1, 2, \cdots, N\}$. To find an optimized routing order, we can view the routing order problem as a black box optimization problem over all permutations for N nets. Due to the recent significant performance of BO on the exploration over permutation spaces [22, 23], we employ a BO method named BOPS-H [22] to address this problem.

We define o_{based} as the sequence $(1, 2, \cdots, N)$. The set of all routing orders along with the composition binary operation $((o_i \circ o_j)(o_{base})) = o_i(o_j(o_{base}))$ constitutes the feasible domain. This domain is known as the symmetric group S_N, and $|S_N| = N!$. For the simplicity of illustration, we will use o_i to represent $o_i(o_{base})$ in later paragraphs.

Then we treat the quality of the corresponding routing result of a specific routing order as the objective function $f : S_N \mapsto \mathbb{R}$, which is defined as:

$$f(o_i) = \begin{cases} WL(ROUTE(o_i)) & Successful\ routing; \\ \infty & Failed\ Routing \end{cases} \tag{2}$$

where $ROUTE(_i)$ represents the routing result obtained by executing our proposed routing method on the input routing order π, and WL gives the total channel length in this routing result, respectively. Our goal is to efficiently find the optimized routing order $\hat{o}$ within only a limited number of attempts:

$$\hat{o} = \underset{o_i \in D}{\operatorname{argmin}} f(o_i),\ |D| <= T \tag{3}$$

4.3 Surrogate Model

The Gaussian process is employed as the surrogate model in the BOPS-H algorithm. Assuming the function g is a realization of a GP with mean function μ and covariance kernel K, we denote it as $g \sim \mathcal{GP}(\mu, K)$. The choice of a kernel function K can drastically affect the quality of the surrogate model. Since the Mallows kernel is universal over the space of permutations [24], we employ the Mallows kernel as the kernel function. It evaluates the similarity of two permutations $k(o_i, o_j)$ as the exponentiated negative of Kendall-tau distance $n_d(o_i, o_j)$ [25] between o_i and o_j:

$$k\left(o_i, o_j\right) = exp(-l \cdot n_d(o_i, o_j)), \tag{4}$$

$$\begin{gathered} n_d(o_i, o_j) = \sum_{n<m} [\mathbb{1}(rank(o_i, n) > rank(o_i, m))\mathbb{1}(rank(o_j, m) < rank(o_j, m)) \\ +\mathbb{1}(rank(o_i, m) < rank(o_i, n))\mathbb{1}(rank(o_j, m) > rank(o_j, n))] \\ n, m \in [1, N] \end{gathered} \tag{5}$$

where $l \geq 0$ is a length-scale hyper-parameter of the kernel. Besides, [22] empirically demonstrated the superior modeling capability of the Mallows kernel. $\mathbb{1}(\cdot)$ is the indicator function, and $rank(o_i, m)$ returns the ranking of m in o_i. Thus, when being used to learn GP-based surrogate models, the Mallows kernel is powerful enough to allow us to capture rich structures in routing orders.

4.4 Acquisition Function and Optimizer

Since the feature space of the Mallows kernel is exponentially large, we adopt the expected improvement as the acquisition function to improve the efficiency of sampling functions from the GP posterior. Meanwhile, to overcome the additional complexity of the GP-based surrogate model with the Mallows kernel, we use a heuristic local search strategy with multiple restarts.

5 EXPERIMENTAL RESULTS

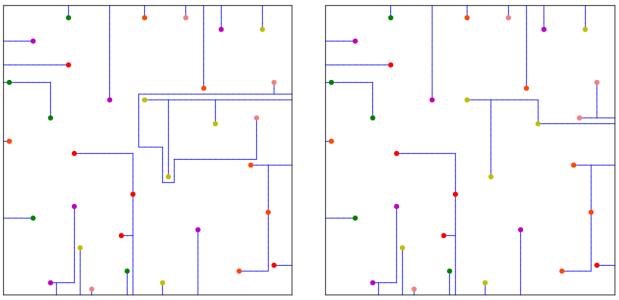
To demonstrate the performance of our method, we compare the proposed method to classic microfluidic control layer escape routing methods with two groups of experiments.

5.1 Experiments on Routing Performance

We simulated six applications and generated corresponding synthetic chip layouts as benchmarks. For baseline methods, we implemented a greedy method [26] that applies operation based compatibility identification to group microvalves, and also an ultra-fast escape routing algorithm proposed in [12]. The proposed method and baseline methods were implemented in Python, and experiments were carried out on an Intel(R) Xeon(R) Gold 6226R 2.90GHz Linux machine with 256 GB RAM.

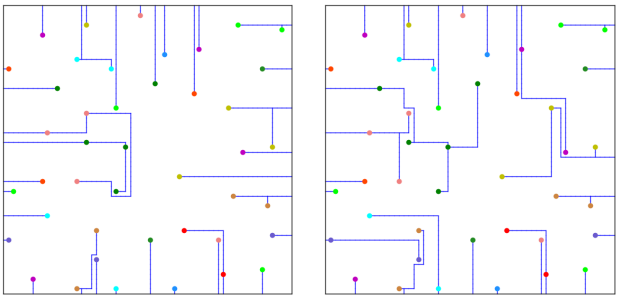
Table 1: Comparison of Routing Performance with Two Baseline Methods Based on Synthesized Benchmarks.

Benchmarks			The Ultra-Fast Escape Routing Method [12]			Greedy Method with OBCI-based Grouping [26]			The Proposed Method		
Index	Chip Size	# microvalves	Channel Length	Runtime (s)	# Ports	Channel Length	Runtime (s)	# Ports	Channel Length	Runtime (s)	# Ports
1	16*16	8	29	0.00098	8	85	0.28	3	86	0.15	3
2	20*20	10	37	0.0014	10	104	0.010	3	63	0.57	5
3	35*35	16	104	0.00044	16	Failed			239	7.03	6
4	42*42	24	183	0.0089	24	Failed			295	32.34	8
5	50*50	32	263	0.014	32	Failed			333	242.53	23
6	60*60	43	445	0.028	43	Failed			436	672.12	33



(a) (b)

Figure 10: Routing results for benchmark#5 by the proposed method. (a) Using the default routing order. (b) Using the optimized routing order derived by BO.

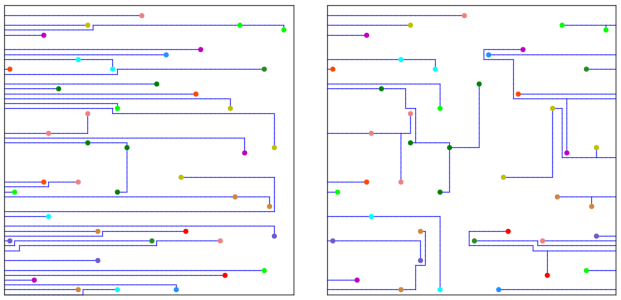


(a) (b)

Figure 11: Routing results for benchmark#6 by the proposed method. (a) Using the default routing order. (b) Using the optimized routing order derived by BO.

To start with, we compare the routing performances of these methods using the default routing orders, which are organized based on the serial number of the nets. 6 benchmarks with different chip sizes and containing different number of microvalves are used in our experiment. By default, we simply specify all four sides of the design as boundaries for the routing process.

Grouping more microvalves together necessitates additional connections among them, which in turn increases the total length of control channels. Consequently, comparing the total length of control channels between two routing results with differing control port usage becomes meaningless. To determine the superiority of one routing result over another, we initially compare their control port usage. Only when the control port usage is identical do we proceed to compare the total length of their control channels.



(a) (b)

Figure 12: Routing results for benchmark#6 with different user-specified boundaries. (a) Only specifying the left side as the boundary. (b) Specifying both the left and right sides of the design as boundaries.

Table 2: The Improvement Achieved by Using BO-Optimized Routing Orders.

Benchmark Index	1	2	3	4	5	6
# Ports	3	5	3	8	23	27
Improvement	0%	0%	50.00%	0%	0%	18.18%
Channel Length	59	63	289	295	271	467
Improvement	31.49%	0%	-20.92%	0%	18.62%	-7.11%

As shown in Table 1, the ultra-fast escape routing method [12] completes routing for all benchmarks with very short runtime. However, the ultra-fast escape routing method assigns an individual control port to each mircovalve, therefore will consume much area. The greedy method proposed in [26] did not indicate how to group microvalves. Here, we apply the previously introduced operation-based compatibility identification (OBCI) to conduct microvalve grouping for it. According to the results, the greedy method greatly reduces the usage of control ports. However, as simply applying OBCI to group microvalves without considering the physical location of them will likely lead to very long connections in a net, which hinders the routing for other nets. As a result, the greedy method only succeeds in the routing for benchmark #1 and #2, which are relatively small and simple, and fails in the routing for all the other benchmarks.

In contrast, the proposed method succeeds in the routing for all benchmarks, and significantly reduces the usage of control ports. Figure 10(a) and Figure 11(a) show the routing results of the proposed method for benchmark #5 and #6, respectively. In these figures, blue lines are the routes, and dots in the same color represent

Table 3: Reliability Evaluation of the Output Generated by Different Methods.

Benchmarks	The Ultra-Fast Escape Routing Method [12]		Greedy Method with OBCI-based Grouping [26]		The Proposed Method without Fault-Tolerant Structures		The Proposed Method with Fault-Tolerant Structures	
Index	P_{anti-b}	P_{anti-l}	P_{anti-b}	P_{anti-l}	P_{anti-b}	P_{anti-l}	P_{anti-b}	P_{anti-l}
1	97.15%	99.98%	91.56%	98.79%	91.76%	96.43%	99.99%	95.62%
2	96.37%	99.76%	90.13%	97.03%	93.89%	99.77%	99.99%	99.75%
3	90.13%	99.86%	N/A		78.75%	90.44%	99.99%	88.33%
4	83.28%	95.41%	N/A		74.46%	93.16%	99.99%	91.66%
5	76.88%	96.65%	N/A		71.68%	91.63%	99.99%	89.01%
6	64.09%	87.84%	N/A		64.67%	90.32%	99.99%	88.39%

the operation-compatible microvalves. Additionally, as introduced before, the proposed method supports user-specified boundaries. As shown in Figure 12, when only specifying the left side as the boundary, or both the left and right sides as boundaries, the proposed method can still complete the routing. It is worth mentioning that the proposed methods can iteratively adjust the threshold for microvalve grouping to find a feasible solution. When the benchmark is extremely complex, the proposed method will eventually choose a very small threshold for microvalve grouping, thereby simply assigning an individual control port to each microvalves, which becomes much simpler to route. In this case, the proposed method can achieve 100% routability for all benchmarks.

We then applied the BO to determine the optimized routing orders that can lead to a shorter channel length. Figure 10(b) and Figure 11(b) show the routing results with optimized routing orders for benchmark #5 and #6, respectively. Detailed results for all benchmarks are shown in Table 2. Benchmark #3 and #6 have significantly improved control port usage. Benchmark #1 and #5 have significantly improved total length of control channels even though they do not have improved control port usage.

5.2 Experiments on Reliability against Defects

In [27], a reliability quantification tool was proposed to objectively evaluate the probabilities of a design having manufacturing defects. In their tool, probabilities of a design having blockage and leakage are quantified as P_b and P_l, respectively. Here, we use $P_{anti-b} = 1-P_b$ and $P_{anti-l} = 1-P_l$ to represent a design's reliability against blockage and leakage, which are probabilities of the design working without blockage and leakage, respectively.

For blockage, they referred to the conclusion based on mass manufacturing in [16], which states that the probability of a channel having blockage is positively correlated to the channel length. Their tool calculates the probability of a channel of length l having blockage by $P_b = 1-(1-p_{blockage})^l$, so $P_{anti-b} = (1-p_{blockage})^l$, where $p_{blockage}$ is the probability of a unit-length channel having blockage. For leakage, they referred to the conclusion that the probability of leakage increases as the length of paralleled channels increases and the spacing between them decreases [16]. Their tool assumes a channel will only have leakage with its neighboring channel, and calculates the probability of leakage happening between a channel pair of length l' and spacing d by $P_l = p_{leakage} \times l'/d^2$, so $P_{anti-l} = 1-(p_{leakage} \times l'/d^2)$, where $p_{leakage}$ is the probability of a unit-length channel pair with unit spacing having leakage. The value of $p_{blockage}$ and $p_{leakage}$ vary with different manufacturing processes [27]. In our experiment, we set $p_{blockage} = 0.01\%$ and $p_{leakage} = 1.00\%$ without loss of generality.

We adopt the reliability quantification tool to evaluate the original output of different methods in the previous experiment for different benchmarks. It is worth mentioning that the original output of the proposed method in the previous experiment does not get our proposed fault-tolerant structures implemented. To demonstrate the effectiveness of the proposed fault-tolerant structures, we implement the fault-tolerant structures to all nets in the original output, and add it to the reliability comparison. As depicted in Table 3, the reliability against leakage for all methods' output remains relatively high and does not significantly decrease as the design size increases. However, the reliability against blockage substantially decreases with increasing design size. The proposed method's output, without fault-tolerant structures, exhibits less blockage reliability than the output of the ultra-fast method but more than the output of the greedy method with OBCI-based microvalve grouping. Upon implementing fault-tolerant structures, the enhanced output slightly reduces the distances between different nets' components, resulting in a minor sacrifice of less than 2% in leakage reliability. On one hand, regarding the reliability against leakage, the proposed method may not be as good as the ultra-fast escape routing method, yet still maintains a satisfactory performance at around 90% On the other hand, the enhanced output's robust connections within each net significantly boost the reliability against blockage to nearly 100% across all cases, demonstrating the substantial effectiveness of the proposed fault-tolerant structure.

6 CONCLUSION

In this paper, we proposed a new microfluidic control layer escape routing methodology. We adopted a hierarchical clustering algorithm to group compatible microvalves that are physically close, and created nets accordingly. To make the routing process efficient, we greedily routed one net at a time and used BO to determine the optimized routing order. During the routing, we created keep-out zones around other nets to ensure safe distances and minimize leakage risk. We used an OARSMT to connect microvalves within a net and adapted the Lee algorithm to connect the OARSMT to user-specified boundaries. We also implemented a fault-tolerant structure as an option to improve reliability against blockage. Experimental results showed that the proposed methodology can efficiently route the channels, significantly reduce control port usage, shorten control channels, and improve the reliability against manufacturing defects compared to baseline methods.

7 ACKNOWLEDGMENT

The research work described in this paper was conducted in the JC STEM Lab of Intelligent Design Automation funded by The Hong Kong Jockey Club Charities Trust. This work is jointly supported by the Research Grants Council of Hong Kong SAR (No. CUHK14217022), and the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation, No. 515003344).

REFERENCES

[1] K. Hu, K. Chakrabarty, and T.-Y. Ho, *Computer-aided design of microfluidic very large scale integration (mVLSI) biochips.* Springer, 2017.

[2] S. Liang, Y. Zhang, R. Altay, H. Gasvoda, M. Li, I. E. Araci, T.-M. Tseng, U. Schlichtmann, and T.-Y. Ho, "LaMUX: Optimized logic-gate-enabled high-performance microfluidic multiplexer design," in *The 61st Design Automation Conference (DAC)*, 2024.

[3] J. W. Hong, V. Studer, G. Hang, W. F. Anderson, and S. R. Quake, "A nanoliter-scale nucleic acid processor with parallel architecture," *Nature biotechnology*, vol. 22, no. 4, pp. 435–439, 2004.

[4] Y. Shi, Y. Cai, Y. Cao, Z. Hong, and Y. Chai, "Recent advances in microfluidic technology and applications for anti-cancer drug screening," *TrAC Trends in Analytical Chemistry*, vol. 134, p. 116118, 2021.

[5] "Microfluidics design rules." https://www.stanfordmicrofluidics.com/design-basics.

[6] K. Hu, F. Yu, T.-Y. Ho, and K. Chakrabarty, "Testing of flow-based microfluidic biochips: Fault modeling, test generation, and experimental demonstration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 10, pp. 1463–1475, 2014.

[7] S. Liang, M. Lian, M. Li, T.-M. Tseng, U. Schlichtmann, and T.-Y. Ho, "ARMM: Adaptive reliability quantification model of microfluidic designs and its graph-transformer-based implementation," in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 1–9, IEEE, 2023.

[8] J. W. Hong, V. Studer, G. Hang, W. F. Anderson, and S. R. Quake, "A nanoliter-scale nucleic acid processor with parallel architecture," *Nature biotechnology*, vol. 22, no. 4, pp. 435–439, 2004.

[9] 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*, pp. 1–6, 2018.

[10] K. Hu, T. A. Dinh, T.-Y. Ho, and K. Chakrabarty, "Control-layer routing and control-pin minimization for flow-based microfluidic biochips," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2017.

[11] 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," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 8, pp. 1588–1601, 2017.

[12] J. Weng, T.-Y. Ho, W. Ji, P. Liu, M. Bao, and H. Yao, "Urber: Ultrafast rule-based escape routing method for large-scale sample delivery biochips," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 1, pp. 157–170, 2018.

[13] Z. Chen, T.-Y. Ho, U. Schlichtmann, D. Chen, M. Liu, H. Yao, and X. Yin, "NeuroEscape: Ordered escape routing via monte-carlo tree search and neural network," in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 01–09, IEEE, 2023.

[14] Q. Liu, D. Lin, C. Chen, H. He, J. Chen, and Y.-W. Chang, "A matching based escape routing algorithm with variable design rules and constraints," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, 2023.

[15] M. Li, H. Gu, Y. Zhang, S. Liang, H. Gasvoda, R. Altay, I. E. Araci, T.-M. Tsneg, T.-Y. Ho, and U. Schlichtmann, "Late breaking results: Efficient built-in self-test for microfluidic large-scale integration (mlsi)," in *The 61st Design Automation Conference (DAC)*, 2024.

[16] K. Hu, T.-Y. Ho, and K. Chakrabarty, "Test generation and design-for-testability for flow-based mVLSI microfluidic biochips," in *2014 IEEE 32nd VLSI Test Symposium (VTS)*, pp. 1–6, IEEE, 2014.

[17] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 11 2005.

[18] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[19] E. W. Dijkstra, "A note on two problems in connexion with graphs," in *Edsger Wybe Dijkstra: His Life, Work, and Legacy*, pp. 287–290, 2022.

[20] R. C. Prim, "Shortest connection networks and some generalizations," *The Bell System Technical Journal*, vol. 36, no. 6, pp. 1389–1401, 1957.

[21] C. Y. Lee, "An algorithm for path connections and its applications," *IRE transactions on electronic computers*, no. 3, pp. 346–365, 1961.

[22] A. Deshwal, S. Belakaria, J. R. Doppa, and D. H. Kim, "Bayesian optimization over permutation spaces," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, pp. 6515–6523, 2022.

[23] C. Oh, R. Bondesan, E. Gavves, and M. Welling, "Batch bayesian optimization on permutations using the acquisition weighted kernel," in *Advances in Neural Information Processing Systems* (S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, eds.), vol. 35, pp. 6843–6858, Curran Associates, Inc., 2022.

[24] Y. Jiao and J.-P. Vert, "The kendall and mallows kernels for permutations," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, p. 1935–1944, JMLR.org, 2015.

[25] M. G. Kendall, "A new measure of rank correlation," *Biometrika*, vol. 30, no. 1/2, pp. 81–93, 1938.

[26] J. McDaniel, D. Grissom, and P. Brisk, "Multi-terminal PCB escape routing for digital microfluidic biochips using negotiated congestion," in *2014 22nd International Conference on Very Large Scale Integration (VLSI-SoC)*, pp. 1–6, IEEE, 2014.

[27] S. Liang, M. Li, T.-M. Tseng, U. Schlichtmann, and T.-Y. Ho, "CoMUX: Combinatorial-coding-based high-performance microfluidic control multiplexer design," in *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, pp. 1–9, 2022.