Pump-Aware Flow Routing Algorithm for Programmable Microfluidic Devices

Guan-Ru Lai, Chun-Yu Lin, and Tsung-Yi Ho
Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan

***Abstract*—As the biochemical experiment becomes more complicated and more diverse, the process of developing a specific-purpose microfluidic biochip for a new task can be very expensive and time consuming. Therefore, the programmable microfluidic devices (PMDs) are proposed as general purpose devices which can perform multiple functions without any hardware modification. Because the PMDs are controlled by pure software program, the assays can be done in parallel and the total completion time can be reduced. However, the high parallelism may cause congestion problem as different reagents are not allowed to cross each other to avoid unexpected mixing. Moreover, since reagents are pushed by the off-chip pump, the free channel from an off-chip pump to the actuated reagent is also prohibited to pass through. This could further complicate the congestion problem and increase the assay completion time significantly. However, some vulnerable reagents may spoil over time during the experiment. For timing critical application, it is indispensable to ensure the total assay completion time is within an upper limit. Therefore, we propose a pump-aware flow routing algorithm which deals with the complex routing congestion while minimizing the assay completion time within an upper limit.**

I. Introduction

Over the last decades, flow-based microfluidic biochips have become popular in the biochemical application among various types of biochips. By using picoliter volume reagents, complex assays still can be executed normally in high accuracy [1]. With the flow-based microfluidic biochips, continuous liquid-flow reagents can flow in the etched channels to accomplish specific operations, such as mixing, transportation, and dilution. Pumps and valves are interplayed to control the pressure-driven flow. In Fig. 1(a), there are two elastomer layers in a basic flow-based microfluidic biochip, flow layer and control layer [2]. Each layer has its own channel network and functionality. In flow layer, biochemical reagents are manipulated in the channels. On the other hand, valves in control layer are designed for blocking fluidic flows in flow layer. By providing the pressure, the membrane valves would be deflected into the flow channels. Between the two layers, an off-chip pump is connected to channels in both layers to generate the pressure for controlling valves and driving flows [3], [4].

The developments in microfluidic integration have seen significant progress recently [5], [6], [7], [8], [9], and a fully software-programmable microfluidic device (PMD) is one of the most exciting technologies [10]. PMDs are designed for general purpose, and users can control PMDs with only software programs to implement assays. The generic architecture of PMDs is composed of a network of intersecting

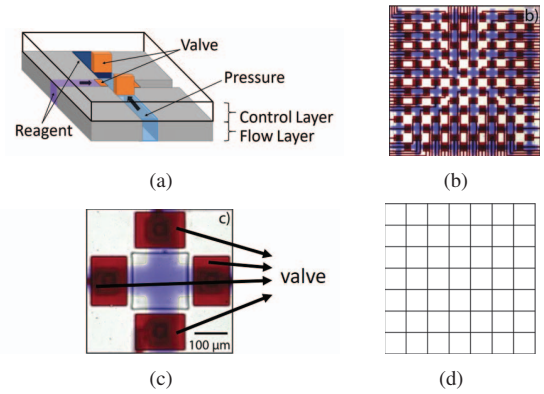


Fig. 1: (a) Schematic of a two-layer flow-based biochip. (b) Central channel network [11]. (c) Every chamber is surrounded by four valves [11]. (d) Mesh-based channel network.

channels controlled by independent valves (Fig. 1(b)), and each chamber in PMDs is surrounded by four valves [11] (Fig. 1(c)). Moreover, the network model of channels can be seen as mesh (Fig. 1(d)). However, when two flows pass through the same chamber, unexpected mixing may happen and lead to incorrect results. Moreover, since the reagent is pushed by the off-chip pump, the free channel from the off-chip pump to the actuated reagent is occupied by pressure and is prohibited from passing through by other reagents, called the pressure-route constraint (Fig. 2(a)). Furthermore, since pressure-route constraint could further complicate the congestion problem, the assay completion time may increase significantly and cause incorrect results due to the spoiled reagents.

The work in [12] solved the congestion problem between reagents while not considering pressure-route constraint. However, reagents are pushed by the off-chip pump and the pressure-route constraint is necessary to ensure the flow. Therefore, if we simply apply the proposed approach [12] to resolve the complex routing congestion, a deadlock may occur during the experiment. This is because reagents may wait each other and form the unsolvable waiting cycle. Besides, the completion time may increase significantly due to the complex congestion and make the results incorrect. Therefore, to resolve the congestion caused by reagents and pressure, we propose a pump-aware flow routing algorithm to guarantee all the experiments can be completed and the completion time of all test cases is within the upper limit of completion time.

The rest of this paper is organized as follows. Section II formulates the routing problem on PMDs. Section III presents

978-3-9819263-0-9/DATE18/©2018 EDAA

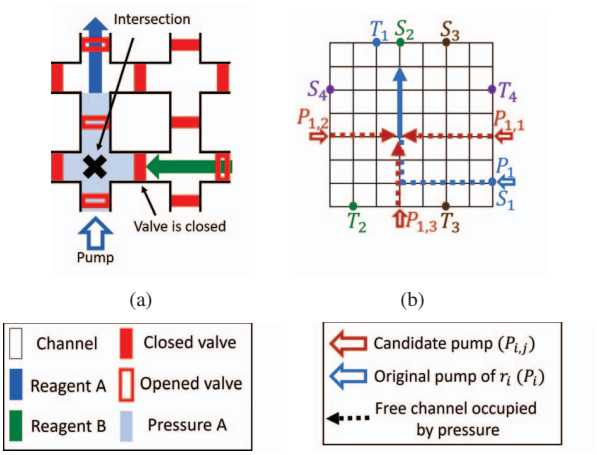


Fig. 2: (a) Pressure-route constraint. (b) $P_{1,1}$, $P_{1,2}$, and $P_{1,3}$ are candidate pumps to replace the original pump P_1.

our proposed algorithm. Section IV shows the experimental results and Section V gives our conclusions.

II. Problem Formulation

The central channel network of the PMDs can be formulated as a mesh graph, shown in Fig. 1(d). Edges and intersections in the mesh graph represent channels and chambers on PMDs, respectively. Sources and targets of reagents, and the off-chip pumps are located on the border of the mesh graph.

The routing problem on PMDs: Given an $m \times n$ mesh-sized graph of the PMD. Each reagent has the respective volume as length l_i, and the coordinates of the source and the target of reagent i are (S_{x_i}, S_{y_i}), (T_{x_i}, T_{y_i}). Moreover, the source and the target of reagent are located in the peripheral of a PMD. The objective of our problem is to minimize the assay completion time without any constraint violation. In our model, we define the time that reagents route one unit of mesh is a timing unit.

The constraints are described in more detail as follows:

1) Source and target restriction: When $t = 0$, each reagent i is located at its source (S_{x_i}, S_{y_i}). Additionally, the reagent must arrive its target (T_{x_i}, T_{y_i}) before the experiment is finished.

2) Valve constraint: The valve constraint states that valves can only be either opened or closed at any timing unit.

3) Fluidic constraint: The fluidic constraint states that an intersection or a channel cannot be passed by two or more reagents at any timing unit.

4) Pressure-route constraint: The pressure-route constraint states that the free channel from pump to the actuated reagent is occupied by pressure and is prohibited to pass through. In Fig. 2(a), reagent A can continue to flow but reagent B cannot. The reason is that the pressure A occupies the intersection and prohibits reagent B from passing through.

5) Pump replacement constraint: Since the pressure-route constraint could complicate the congestion problem, choosing the pump at different locations to push the reagent may solve the congestion. In this paper, only three candidate pumps are eligible to push the reagent. An example is shown in Fig. 2(b), $P_{1,1}$, $P_{1,2}$, and $P_{1,3}$ are at the three candidate locations of the pump to push reagent r_1 at this timing unit.

III. Pump-aware Flow Routing Algorithm

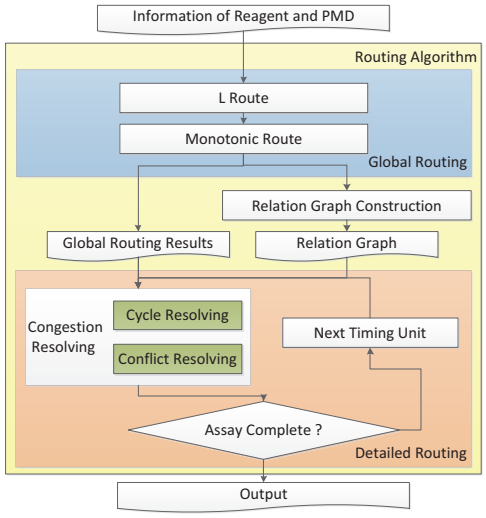


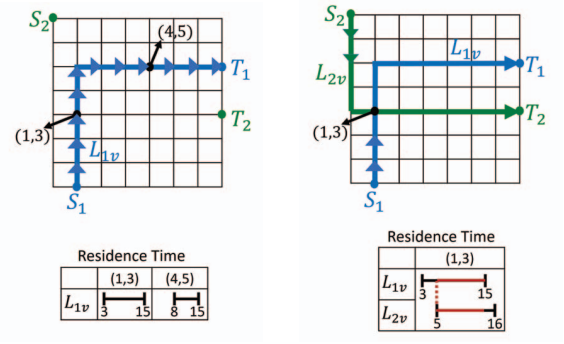
Fig. 3: Overview of our pump-aware flow routing algorithm.

Fig. 3 shows our flow chart of pump-aware flow routing algorithm. First, we apply L route to find the initial route for each reagent. Second, the monotonic route is applied to seek more probable routes. After applying the two routing methods, the global routing results for every reagent are determined. Then, the relation graph is constructed to record the relation between reagents based on the global routing results. Once we have the global routing results and the relation graph, we can simulate the experiment in detailed routing. Since we need to ensure the experiment is completed without any constraint violation, congestion problems should be resolved in detailed routing phase. The details of the algorithms are presented in the following subsections.

A. Global Routing

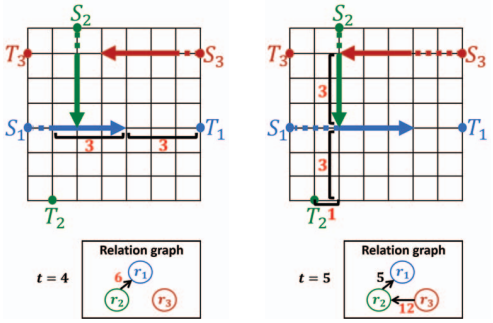
1) L Route: There are two L-shape routes for each reagent, one is L_h, and the other is L_v. For example, L_h routes horizontally first, and then routes vertically to arrive the target. We record the time that every reagent passes through the intersections to determine whether the L-shape routes of any two different reagents are conflicted or not. If the two L-shape routes are conflicted to each other, they would have the conflict weight between them. For each reagent, we choose the L-shape route with less conflict weight as the initial solution.

The arrival time a and the exit time e are recorded to the nodes on two L-shape routes. We record the time by the notations $a_{(i,x,y,Lroute)}$ and $e_{(i,x,y,Lroute)}$. Take Fig. 4(a) for example, r_1 will arrive node $(1, 3)$ at $t = 3$ and exit it at $t = 15$. Thus, $a_{(1,1,3,L_{1v})}$ is 3 and $e_{(1,1,3,L_{1v})}$ is 15. Note that the exit time of all nodes on L_{1v} are $t = 15$ since the pressure will occupy the free channel until all the units of the reagent



(a) (b)

Fig. 4: (a) Record arrival and exit time of L_{1v}. (b) Conflict weight between L_{1v} and L_{2v} at node $(1,3)$.



(a) (b)

Fig. 5: (a) The relation graph at $t = 4$. (b) The relation graph at $t = 5$.

arrive the target. Based on the residence time, we can calculate the conflict weight between two reagents. In Fig 4(b), L_{1v} and L_{2v} will have conflict at node $(1,3)$ from $t = 5$ to $t = 15$. Therefore, the conflict weight between L_{1v} and L_{2v} at $(1,3)$ will be $15 - 5 = 10$. The cost function of conflict weight $W_{i,x,y,Lroute}$ is presented as follows:

$$W_{i,x,y,Lroute} = \sum_{j=1}^{N} \sum_{(k=1)}^{2} max(e_{min} - a_{max}, 0) \quad (1)$$

$$e_{min} = min(e_{(i,x,y,Lroute)}, e_{(j,x,y,k)}) \quad (2)$$

$$a_{max} = max(a_{(i,x,y,Lroute)}, a_{(j,x,y,k)}) \quad (3)$$

$$W_{i,Lroute} = \sum_{(x=0)}^{m} \sum_{(y=0)}^{n} W_{i,x,y,Lroute} \quad (4)$$

In the Equation (1), N is the number of reagents, and for every reagent, we need to choose one of the two L-shape routes as initial solution. Take r_1 for example, if L_{1v} is chosen, the value of W_{1,L_v} is $10 + 4 = 14$ (conflict weight with L_{2v} and L_{2h}, respectively). On the other hand, the value of W_{1,L_h} is $4 + 13 = 17$. Therefore, L_{1v} will be the initial solution for r_1.

2) Monotonic route: A good global routing strategy may reduce the congestion problem significantly. Therefore, it is necessary to seek more patterns for every reagent to find a route with less conflict. Thus, monotonic routing approach is performed to search multiple shortest routes within the bounding box of source S and target T. The order of applying monotonic algorithm to the reagents is based on the number of overlaps between the bounding boxes. That is, the reagent which has the least overlaps will have the highest priority as the route of this reagent may have less influence on other reagents. We denote the chosen reagent as r_c.

In order to adopt monotonic algorithm, the cost function of each node in bonding box is presented as follows:

$$C_{(x,y)} = min(C_{(x-1,y)}, C_{(x,y-1)}) + W_{i,x,y} \quad (5)$$

$$C_{(S_x,S_y)} = W_{i,S_x,S_y} \quad (6)$$

Similar to the L Route, conflict weight $W_{i,x,y}$ between r_c and other reagents is determined by their arrival and exit time to the node. Note that the equation is a special case that the source of the bounding box must at the lower left of the target. After calculating the conflict weight of every node in the bounding box, denoted by $C_{(x,y)}$, the route with the least cost from S to T will be chosen as the final route.

B. Relation Graph Construction

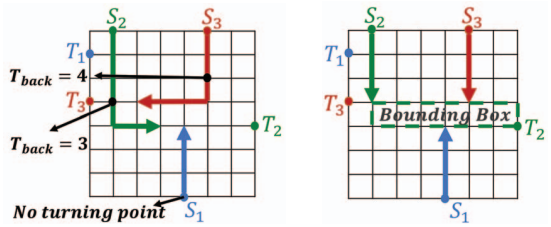
Once global routing is finished, the route of every reagent is determined. Then, the relation graph can be constructed to record the relation between reagents at each timing unit. According to the relation graph, we can predict whether the reagent may block other reagents or pass by a congested area. Both situations may increase the experimental time and should be avoided.

In the relation graph, the vertices represent the reagents, and the directed edges indicate that reagent r_w needs to wait for reagent r_b. In Fig. 5(a), at $t = 4$, there is a conflict between r_1 and r_2. Since r_2 needs to wait for r_1, r_1 and r_2 are defined as r_b and r_w, respectively. r_2 waits 6 timing units for r_1, which is calculated according to the remainder time that r_1 arrives its target.

Moreover, the waiting time of r_b may affect the waiting time of r_w. In Fig, 5(b), at $t = 5$, r_3 waits 7 timing units for r_2. However, at the same time, r_2 needs to wait 5 timing units for r_1. Therefore, the total waiting time of r_3 is $7 + 5 = 12$. During relation graph construction, we only roughly calculate the waiting time at the timing unit of conflict without updating with timing condition. This is because the real simulation is time consuming and the relation graph is only a prediction to assist the strategy of detailed routing.

C. Detailed Routing

In detailed routing phase, we find the exact route for each reagent by simulating the experiment with timing condition. When simulating the experiment based on the global routing results, there may be congestion problems due to the conflicts and the waiting cycles between reagents. Since the congestion problem may increase the assay completion time or even cause a deadlock, the results may become incorrect. Therefore, we propose the approaches to resolve the cycles and the conflicts in the following subsections.



(a) (b)

Fig. 6: (a) Backtracks each reagent to its p_{b_i}. (b) Using monotonic algorithm to reroute the reagent in the bounding box.

1) Cycle Resolving: When two or more reagents wait for each other, they will form the waiting cycle and make the experiment never terminate. Therefore we propose a progressive algorithm with three phases to resolve the cycles: (1) Backtrack method, (2) Dijkstra's algorithm, and (3) Rollback method. When one phase is disable to resolve cycle, we will apply the next phase. In order to resolve cycle without increasing any routing length of the reagent, we first apply Backtrack method. However, since Backtrack method may increase the CPU time, we define a limit of the timing units that the experiment can backtrack. In the second phase, Dijkstra's algorithm, is applied to reroute the reagents which compose the waiting cycle. In this phase, the length of the route may increase since the reagent should bypass the other reagents to resolve cycle. If first two phases cannot resolve cycle, we will apply the last phase to ensure the cycle can be definitely resolved. In Rollback method, we lock one of the reagents at its source until one of other reagents arrives the target. We will detail the approach in the following subsections.

a) Backtrack method: In Backtrack method, we first backtrack the assay to the time before the waiting cycle forms. Then, reroute one of reagents which compose the waiting cycle in order to prevent the same cycle forms again. The approaches that determine the number of timing units to backtrack and the reagent to be rerouted are presented as follows. First, we determine the point p_{b_i} that the reagent should be backtracked to for all the reagents which compose the waiting cycle, and p_{b_i} is the point before arriving the last turn point of each reagent. Second, we calculate the total timing units that each reagent should be backtracked. In the end, we choose the reagent with least number of timing units as the reagent to be rerouted, and its number of timing units is the total timing unit that the assay should be backtracked.

In Fig. 6(a), the p_{b_i} of reagents are marked with black dots. The number of timing units that r_2 is backtracked to p_{b_2} is 3, r_3 is 4, and r_1 does not have any p_{b_1} in this example. Therefore, r_3 is chosen to be rerouted and the total number of timing units to be backtracked is 3. Then, r_2 is rerouted by monotonic algorithm similar to global routing shown in Fig. 6(b). If the cycle still cannot be resolved after rerouting, Backtrack method will be applied once again until the cycle is resolved or the number of timing units exceeds δ. We define δ as half of the width of a mesh to reduce CPU time.

b) Dijkstras algorithm: In this phase, we reroute one of the reagents which compose the waiting cycle with Dijkstras algorithm. The function of distance is presented as follows.

$$d_{x,y} = min(d_{x-1,y}, d_{x,y-1}) + 1 + T_{stay} + T_{punish} \quad (7)$$

$$d_{S_x,S_y} = 0 \quad (8)$$

In Equation (7), $d_{x,y}$ is the distance from source of the reagent to the node (x, y). If no conflict occurs at node (x, y), the distance $d_{x,y} = min(d_{x-1,y}, d_{x,y-1}) + 1$. However, if there is the conflict, two additional costs T_{stay} and T_{punish} may be added to $d_{x,y}$. T_{stay} is the remaining number of timing units that reagent r_o which have occupied node (x, y) will exit. T_{punish} is used to prevent other reagents from passing through the node that is occupied by the reagent which is waiting for the other reagent. We define T_{punish} as half the size of a PMD.

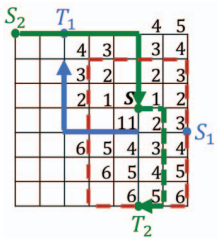


Fig. 7: Reroute r_2 with Dijkstra's algorithm to solve the cycle.

In Fig. 7, r_2 is rerouted with its new source at node $(5, 4)$. Take $d_{6,4}$ and $d_{5,3}$ as examples, since no reagent occupies node $(6, 4)$, $T_{stay} = 0$ and $T_{punish} = 0$. The value of $d_{6,4} = 0 + 1 + 0 + 0 = 1$. On the other hand, since node $(5, 3)$ is occupied by r_1, $T_{stay} = 7$ and $T_{punish} = 7/2 = 3$. The value of $d_{5,3}$ is 11. Based on the distance of nodes from source to target, we can find a new route for r_2 to resolve cycle.

c) Rollback method: To ensure the cycle can be resolved, we restart the total experiment and lock one of the reagents which compose the cycle at its source at the beginning of the experiment. Only when one of the other reagents which compose the cycle arrives the target, the locked reagents can start the route. The reagent which has the shortest path from the source to the target will be chosen to be locked.

For eample, in Fig. 8(a), r_1, r_2, and r_3 form a cycle at $t = 6$. The shortest path of r_1, r_2, and r_3 are 13, 10, and 8, respectively. Thus, r_3 is chosen to be locked at node $(5, 7)$, shown in Fig. 8(b), until r_2 arrives its target at $t = 15$ (Fig. 8(c)), r_3 can restart its experiment at $t = 16$ (Fig. 8(d)). The Rollback method can resolve the cycle definitely but may increase CPU time significantly.

2) Conflict Resolving: There are two conflict types in detailed routing:

Type1: Two or more reagents occupy the same intersection at the same time. Therefore, the priority of the reagents to pass through the intersection should be determined. The cost function of the priority is presented as follows.

$$P_i = -\alpha * l_i + \beta * N_{blocked} - \gamma * D_{Area} \quad (9)$$

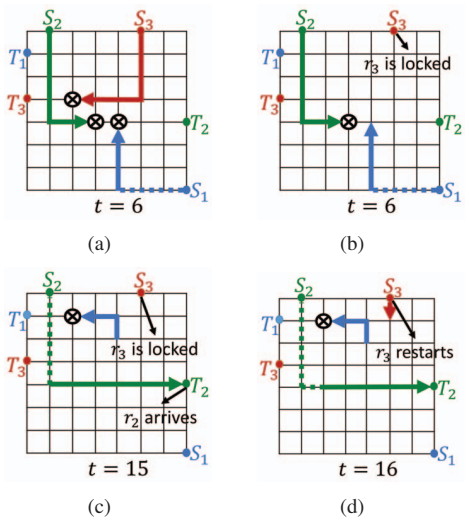


Fig. 8: (a) An example of cycle caused by three reagents. (b) $t = 6$, r_3 is blocked at source. (c) $t = 16$, r_2 arrives target. (d) $t = 16$, r_3 restarts its experiment.

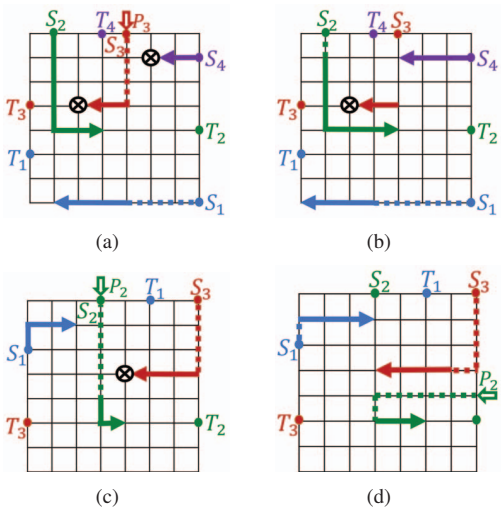


Fig. 9: (a) r_3 waits for r_2 at $t = 6$. (b) P_3 is turned-off to enable r_4 to pass through. (c) r_1 and r_3 wait the pressure of r_2. (d) Changing P_2 to enable r_1 and r_3 to pass through.

The priority P_i is determined by the lengths of r_i, the number of reagents which are predicted to be blocked by r_i with the relation graph ($N_{blocked}$), and the degree of congestion of the routing area in the future (D_{Area}). The reagent which has the highest priority can pass through the intersection first.

Type2: A reagent occupies the intersection that is already occupied by another reagent. We divide this type of conflict into two cases.

Case1: Reagent r_w waits for another reagent r_b.

Step1. Find a new route by monotonic algorithm for r_w.

Step2. If there is no available route for r_w, r_w should wait for r_b until there is an available route for r_w or r_b exits the intersection. While r_w is waiting for r_b, the pump of r_w will be turned-off in order to decrease the the degree of congestion caused by pressure.

In Fig. 9(a), r_3 waits for r_2 at $t = 6$. First, we apply monotonic algorithm to reroute r_3, but obviously, there is no available route in this example. Therefore, r_3 will wait for r_2 until r_2 exits the conflict intersection. Moreover, since pump of r_3 is turned-off and would not occupy the intersection, r_4 can pass through the intersection instead of waiting for r_3, shown in Fig. 9(b).

Case2: Reagent r_w waits for the pressure of r_p.

Step1. Find a new route by monotonic algorithm for r_w.

Step2. If there is no any available route for r_w, check whether there is any candidate pump can push r_p.

Step3. If there is no candidate pump can push r_p, one of r_w and r_p that has lower priority (Equation (9)) should wait and turn-off its pump.

Take Fig. 9(c) as an example, r_1 and r_3 wait for the pressure of r_2. Since there are no available routes for r_1 and r_3, we then check the candidate pump of r_2. There are three candidate pumps for r_2, one is located at node $(0, 3)$, another is located at node $(7, 3)$, and the other is its original pump P_2. The cost function to determine activated pump is presented as follows.

$$C_p = \alpha * l_p + \beta * D_{Area} \tag{10}$$

In Equation (10), the cost of pump C_p is determined by the length of free channel occupied by pressure (l_p), and the degree of congestion of routing area in the future (D_{Area}). The candidate pump which has lower C_p will be chosen as the pump of r_2. Fig. 9(d) shows the result after changing the pump of r_2.

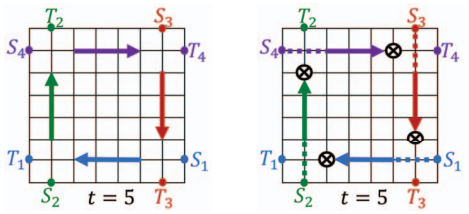
IV. Experimental Results

Our algorithm is implemented in the C++ programming language on Xeon 3.50GHz CPU with 128GB memory. We compare our algorithm with two baselines. One is Sequential Route which routes the reagents one after another. That is, only one reagent can move from the source to the target, and the remaining reagents are locked at their sources. If one reagent arrives its target, one of the remaining reagents begins to move. The second baseline is Naïve Approach which does not apply any strategy of turning-off the pump or pump replacement to resolve congestion problem. Therefore, the pressure will always occupy the free channel until the reagent arrives its target. Moreover, the work [12] cannot route all cases since with backtrack method, reagents need to change the directions in the bounding boxes and restart the routes at the same time. Therefore, the same cycle may always occur and cause the deadlock, shown in Fig.10.

We use the benchmarks provided by [12], which has 10 cases for each benchmark. Table I compares the resulting assay completion and CPU time of Sequential Route, Naïve Approach, and our pump-aware flow routing algorithm. The value of assay completion time and CPU time in the table are the average value of each benchmark. The experimental results show that both completion time and CPU time of Naïve Approach are much larger than ours, which certifies the congestion problems caused by the pressure-route constraint

TABLE I: Experimental results for Sequential Route, Naïve Approach, and our pump-aware flow routing algorithm

Benchmark	Sequential		Naïve		Ours	
	Assay completion time	CPU Time	Assay completion time	CPU Time	Assay completion time	CPU Time
test1 (10 × 10)	82.1	< 0.001	43.2	0.037	34.5	0.037
test2 (10 × 10)	162.9	< 0.001	84.8	0.33	56.2	0.09
test3 (20 × 20)	537.9	< 0.001	207.6	14.8	104.2	0.95
test4 (30 × 30)	1150.5	< 0.001	396.6	77.15	162.1	7.7
test5 (40 × 40)	1973.3	< 0.001	720.9	721.99	238.9	60.52
test6 (50 × 50)	3006.5	< 0.001	958.8	2880.8	315.7	230.09
N. Average	7.5766	approx. 0	2.646	12.34	1	1



(a) (b)

Fig. 10: (a) The situation obtained from [12] which does not consider pressure-route constraint. (b) Backtrack method cannot solve the cycle when considering pressure-route constraint.

will affect the results significantly. This is because Rollback method may be applied frequently and decreases the degree of parallelism without the strategy of turning-off or changing pumps to deal with the pressure-route constraint. As shown in the Table I, the assay completion time of our pump-aware flow routing algorithm is 7.57x and 2.64x faster than the Sequential Route and Naïve Approach, respectively.

Besides, Fig. 11 shows the comprehensive evaluation of finished test cases for each benchmark. Since each test case has its upper limit of assay completion time to avoid the incorrect results, we apply maximum (CT_{max}) and minimum (CT_{min}) of assay completion time to evaluate the results of our algorithm. CT_{max} is the completion time with Sequential Route and CT_{min} is the completion time that every reagent routes with its shortest path in parallel without any constraints. For example, in test6 (Fig. 11(f)), if the upper limit is assigned to 10% more than CT_{min}, all the test cases with our algorithm are finished within the upper limit and promised to be correct. However, not all the test cases with Naïve Approach can be finished within the same upper limit.

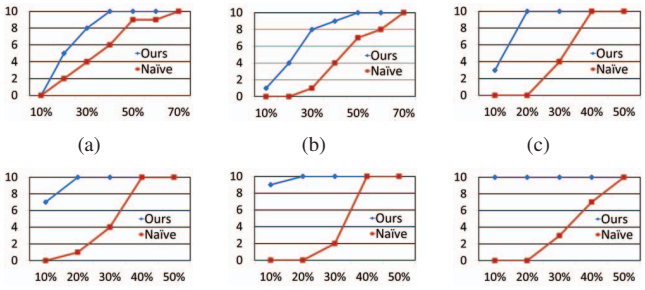


Fig. 11: Number of finished test cases for each benchmark. (a) test1. (b) test2. (c) test3. (d) test4. (e) test5. (f) test6.

V. Conclusions

This paper has presented the first pump-aware flow routing algorithm on PMDs which considers pressure-route constraint and minimizes assay completion time. In the pump-aware flow routing algorithm, we used L route and monotonic route to find the initial route for each reagent in global routing phase. Then, we simulate the experiment and deal with the congestion caused by reagents and pressure in detailed routing phase. Experimental results show that our algorithm can achieve the best assay completion time among all baseline methods in reasonable CPU time and ensure all the assay completion time to be within the upper limit of assay completion time.

References

[1] T. Squires and S. Quake, "Microfluidics: Fluid physics at the nanoliter scale," Reviews of Modern Physics, vol. 77, no. 3, pp. 977-1026, 2005.

[2] M. A. Unger, "Monolithic microfabricated valves and pumps by multilayer soft lithography," Science, vol. 288, pp. 113-116, 2000.

[3] J. Melin and S. R. Quake, "Microfluidic large-scale integration: The evolution of design rules for biological automation," Annual Review of Biophysics and Biomolecular structure, vol. 36, pp. 213-231, 2007.

[4] "Stanford microfluidic foundry: Basic design rules," http://thebigone.stanford.edu/foundry/services/basicdesignrules.html

[5] K. Hu, F. Yu, T.-Y. Ho, and K. Chakrabarty, "Testing of Flow-Based Microfluidic Biochips: Fault Modeling, Test Generation, and Experimental Demonstration," IEEE TCAD, vol. 33, no. 10, pp. 1463-1475, 2014.

[6] W. H. Minhass, P. Pop, J. Madsen, and T.-Y. Ho, "Control Synthesis of the Flow-Based Microfluidic Large-Scale Integration Biochips," Proc. ASPDAC, pp. 205-212, 2013.

[7] K.-H. Tseng, S.-C. You, J. Y. Liu, and T.-Y. Ho, "A Top-Down Synthesis Methodology for Flow-Based Microfluidic Biochips Considering Valve-Switching Minimization," Proc. ISPD, pp. 123-129, 2013.

[8] T.-M. Tseng, M. Li, B. Li, T.-Y. Ho, and U. Schlichtmann, "Columba: Co-Layout Synthesis for Continuous-Flow Microfluidic Biochips," Proc. DAC, pp.147:1-6, 2016.

[9] H. Yao, Q. Wang, Y. Ru, T.-Y. Ho, and Y. Cai, "Integrated Flow-Control Co-Design Methodology for Flow-Based Microfluidic Biochips," IEEE D&T, vol. 32, no. 6, pp. 60-68, 2015.

[10] E. C. Jensen, B. P. Bhat and R. A. Mathies, "A digital microfluidic platform for the automation of quantitative biomolecular assays," Lab on Chip, 10(6): pp. 685-691, 2010.

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

[12] Y.-S. Su, T.-Y. Ho, and D.-T. Lee. "A routability-driven flow routing algorithm for programmable microfluidic devices," Proc. ASPDAC, pp. 605-610, 2016.