A Contamination Aware Droplet Routing Algorithm for Digital Microfluidic Biochips

Tsung-Wei Huang, Chun-Hsien Lin, and Tsung-Yi Ho*
Department of Computer Science and Information Engineering
National Cheng Kung University, Tainan, Taiwan

ABSTRACT

In this paper, we propose a contamination aware droplet routing algorithm for digital microfluidic biochips (DMFBs). To reduce the routing complexities and the used cells, we first construct preferred routing tracks by analyzing the global moving vector of droplets to guide the droplet routing. To cope with contaminations within one subproblem, we first apply a k-shortest path routing technique to minimize the contaminated spots. Then, to take advantage of multiple wash droplets, we adopt a minimum cost circulation algorithm (MCC) for optimal wash-droplet routing to simultaneously minimize used cells and the cleaning time. Furthermore, a look-ahead prediction technique is used to determine the contaminations between successive subproblems. After that, we can simultaneously clean both contaminations within one subproblem and those between successive subproblems by using the MCC-based algorithm to reduce the execution time and the used cells. Based on four widely used bioassays, our algorithm reduces the used cells and the execution time significantly compared with the state-of-the-art algorithm.

1. INTRODUCTION

Digital microfluidic biochip (DMFB) is an emerging technology that aims to miniaturize and integrate droplet-handling on a chip. Recently, many on-chip laboratory procedures such as immunoassay, real-time DNA sequencing, and protein crystallization have all been successfully demonstrated on DMFBs. The dynamic reconfigurability inherent in DMFBs allows different droplet routes to share cells (electrodes) on the microfluidic array during different time intervals. However, contaminations caused by bead retention and liquid residue between successive droplet routes of different biomolecules may cause inevitable erroneous reaction. Moreover, these errors will possibly breakdown the electrodes and cause electrode short problems, which result in physical defects and produce incorrect behaviors in the electrical domain. Intuitively, contaminations can be avoided by routing in disjoint manner. This method avoids the overlap between different droplet routes thereby minimizing the likelihood of the contamination problem. However, as the increased design complexity enabled more and more biological operations to a DMFB, finding disjoint routes has become more and more difficult. Furthermore, disjoint routes also restrict the spare cells for replacing faulty primary cells to ensure the correctness of bioassay execution. Hence, the fault tolerance of bioassay is significantly reduced.

Although silicone oil with its low surface tension and spreading property has been advocated as a filler medium to prevent contaminations, it has been proved that it is not sufficient enough for many types of proteins and heterogeneous immunoassays. To cope with this problem, a wash droplet is introduced to clean the contaminated spots on the surface of the microfluidic array. Given an initial bioassay with two droplets and peripheral devices as shown in Figure 1 (a). If we adopt the disjoint routes to avoid the contamination problem as shown in Figure 1 (b), the execution time and the number of used cells for nets are 18 and 26, respectively. In Figure 1 (c), a contaminated spot (cross-section) occurs between two different routes by simply adopting shortest path routing. To clean this contaminated spot, a wash droplet is dispensed from the wash reservoir and transported via this contaminated spot. As shown in Figure 1 (d), to ensure the correctness of wash operation, the wash droplet must clean the contaminated spot in the time interval (t^1_{cs}, t^2_{cs}), where t^1_{cs} and t^2_{cs} denote the arrival time at the contaminated spot of d_1 and d_2, respectively. If the wash droplet cannot arrive the contaminated spot before t^2_{cs}, t^2_{cs} must be postponed until this contaminated spot has been cleaned. By this wash operation, the execution time and the used cells for nets are reduced to 12 and 19, thereby achieving a better solution quality. Thus, if the wash operation cannot be simultaneously considered with droplet routing, the droplet transportation time will increase significantly, thereby causing the time-to-result effects and reducing the reliability of bioassay.

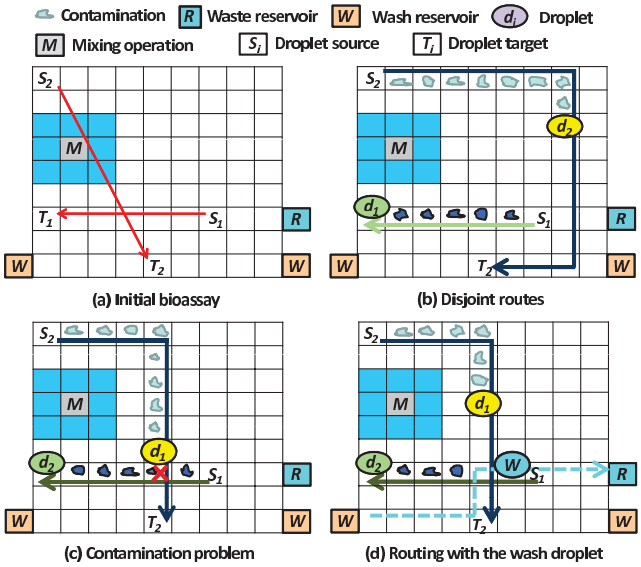

Figure 1: Illustration of the contamination aware droplet routing. (a) Initial bioassay. (b) Disjoint routing for contamination avoidance. (c) Shortest path routing with a contaminated spot. (d) Wash-droplet routing.

Furthermore, contaminated spots occur not only within one subproblem (intra-contaminations) but also between successive subproblems (inter-contaminations). Contaminations in the previous subproblem are treated as blockages for the next subproblem. Additional wash droplets are needed to clean the inter-contaminations and may cause timing overhead for bioassays.

In this paper, we propose a contamination aware droplet routing algorithm on DMFBs. To fully utilize wash droplets, we simultaneously clean both intra- and inter-contaminations within one subproblem that can reduce the execution time significantly. Furthermore, we can effectively minimize the used cells to achieve better reliability and fault tolerance for bioassays.

1.1 Background and Related Prior Work

Droplet routing is a critical step in DMFB physical design automation. Unlike traditional VLSI routing, in addition to routing path selection, the droplet routing problem needs to address the issue of scheduling droplets under the practical constraints imposed by the fluidic property and the timing restriction of the synthesis result.

*This work was partially supported by the National Science Council of Taiwan ROC under Grant No. NSC 96-2220-E-006-013.

ICCAD'09, November 2–5, 2009, San Jose, California, USA.

Droplet routing problem has attracted much attention in the literature recently [3, 4, 7, 9, 11]. Current droplet-routing algorithms have a detrimental effect on the unrestricted sharing of used cells by various droplet routes. In [12], a novel droplet routing algorithm is proposed for cross-contamination avoidance. It attempts to determine disjoint droplet routes by applying modified Lee algorithm. If it is infeasible to find disjoint routes, the wash droplets are then scheduled between successive droplet visits to clean up the contaminations. However, disjoint routes restrict the spare cells for replacing faulty primary cells to ensure the correctness of bioassay execution. Hence, the fault tolerance of bioassay is significantly reduced. Since contaminated spots may also occur between successive subproblems (inter-contaminations), a wash operation is also scheduled between successive subproblems to avoid the contamination problem. However, the extra execution time of wash operation interrupts the continuous biological operations between successive subproblems, which suffers from the time-to-result effects and sample degradation problem. Figure 2(a) illustrates the method proposed in [12].

1.2 Our Contribution

In this paper, we propose a contamination-aware droplet routing algorithm for DMFBs. Unlike the aforementioned routing algorithm of wash-droplet, which cleans intra- and inter-contaminations separately, our algorithm simultaneously clean them within one subproblem to reduce the execution time significantly (see Figure 2 for an illustration). To fully utilize the wash droplets, our algorithm consists of three major stages: preprocessing, intra-contamination aware routing, and inter-contamination aware routing between successive subproblems. Different from the aforementioned work, our algorithm has the following distinguished features:

- A global moving vector analysis for constructing preferred routing tracks to minimize the number of used unit cells.
- A k-shortest path routing technique to minimize the contaminated spots within one subproblem.
- A routing compaction technique by dynamic programming to determine the contaminated spots within one subproblem and minimize the execution time of bioassays.
- A look-ahead prediction technique to determine the contaminated spots between successive subproblems and minimize the total routing time of wash droplets.
- A minimum cost circulation technique is adopted to simultaneously clean intra- and inter-contaminations to minimize the used cells and the execution time.

Based on four widely used bioassays, our algorithm reduces the used cells and the execution time significantly compared with the state-of-the-art algorithm.

The rest of this paper is organized as follows. Section 2 formulates the contamination aware droplet routing problem, while Section 3 describes the overview of our algorithm. Our algorithm consists of three major stages: preprocessing, intra-contamination aware routing, and inter-contamination aware routing which are given in Section 4, 5, and 6, respectively. Finally, the experimental results and the concluding remarks are provided in Section 7 and 8.

2. PROBLEM FORMULATION

There are three routing constraints in contamination aware droplet routing: the fluidic constraint, the timing constraint, and the contamination constraint. The fluidic constraint is used to avoid the unexpected mixtures between two droplets of different nets during their transportation and it can further be divided into the static and dynamic fluidic constraints [9]. Let d_i at (x_i^t, y_i^t) and d_j at (x_j^t, y_j^t) denote two independent droplets at time t. Then, the following constraints should be satisfied for any t during routing:

- Static constraint: $|x_i^t - x_j^t| > 1$ or $|y_i^t - y_j^t| > 1$.
- Dynamic constraint: $|x_i^{t+1} - x_j^t| > 1$ or $|y_i^{t+1} - y_j^t| > 1$ or $|x_i^t - x_j^{t+1}| > 1$ or $|y_i^t - y_j^{t+1}| > 1$.

The static fluidic constraint states that the minimum spacing between two droplets is one cell for any t during routing. The dynamic fluidic constraint states that the activated cell for d_i cannot be adjacent to d_j. The reason is there can be more than one activated neighboring cell for d_j. Therefore, we may have an unexpected mixing between d_i and d_j.

Besides the fluidic constraint, there exists the timing constraint. The timing constraint specifies the maximum arrival time of a droplet from its source to target. For fast bioassay execution or better reliability, it is desirable to minimize the execution time among all droplets. Furthermore, it is desirable to minimize the number of unit cells that are used during routing. Since a unit cell of a DMFB can be defective due to manufacturing or environmental issues, using a smaller number of unit cells for routing can be beneficial for robustness.

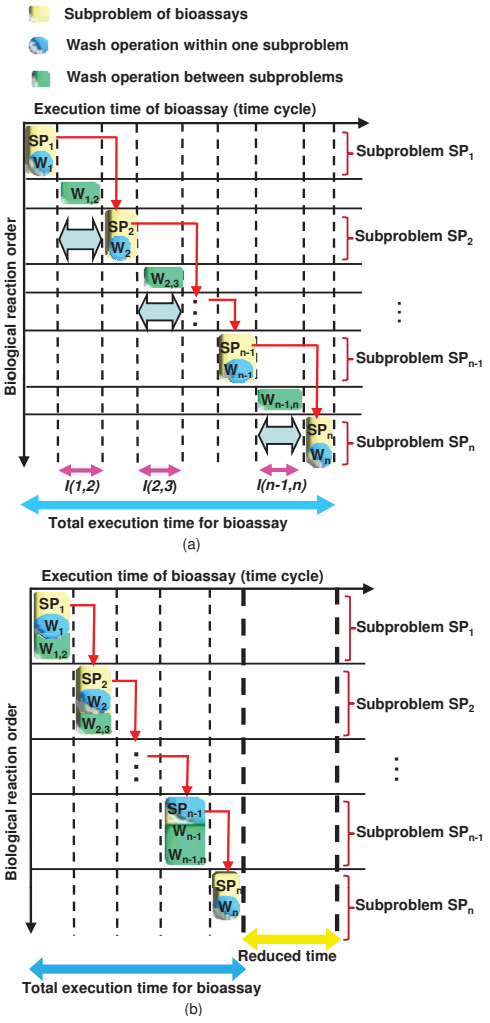

Figure 2: Illustration of different wash strategy for inter-contaminations. (a) Handle inter-contaminations between successive subproblems. (b) Handle intra- and inter-contaminations simultaneously within one subproblem.

The contamination constraint is to prevent the contamination problem from the erroneous bioassay outcome. This is because when two droplets successively pass through a same cell, the liquid residue left behind by the first droplet can contaminate the second droplet and may mislead the bioassay outcome. To characterize the contamination constraint, let the cell c_{ij} denotes the contaminated spot caused by the two routes of d_i and d_j, and $\hat{t}^i_{c_{ij}}$ ($\hat{t}^j_{c_{ij}}$) represent the corresponding arrival clock cycle at c_{ij} for d_i (d_j). To avoid the contamination problem, a wash droplet must clean (pass through) this contaminated spot in the time interval $[\hat{t}^i_{c_{ij}}, \hat{t}^j_{c_{ij}}]$ to perform the wash operation.

The contamination aware droplet routing problem on a 2D plane can be formulated as follows:

Input: A netlist of droplets, a set of wash droplets, the locations of blockages, the locations of reservoirs, and the timing constraint.

Objective: Route all droplets from their source cells to their target cells while minimizing the contaminated spots, the execution time, and the used cells for better fault tolerance. Route wash droplets to clean contaminated spots between different droplet routes while minimizing the cleaning time.

Constraint: Both fluidic and timing constraints are satisfied.

3. ALGORITHM OVERVIEW

Our contamination aware droplet routing algorithm which consists of three major stages: preprocessing, intra-contamination aware

TABLE I: NOTATIONS USED IN OUR ALGORITHM.

D	set of droplets
W	set of wash droplets
Blk	cell set of blockages
SP_p	subproblem p
B_i	bounding box from the source of d_i to its target location
CS	cell set of contaminated spots
Q_{eq}	priority queue for determination of routing order
Q_{cs}	priority queue for number of contaminated spots of each routing path
$CS_{(p,q)}$	cell set of non-washed contaminated spots between subproblems SP_p and SP_q. Note that $p = q$ represents the contaminated spots in one subproblem SP_p.
CS_p	cell set of non-washed contaminated spots in subproblem SP_p. Note that $CS_p = CS_{(r,p)}$, where $1 \leq r \leq p$.
$LA(v_i, v_j)$	set of non-washed look-ahead contaminated spots in the bounding box of nodes v_i and v_j.

routing, and inter-contamination aware routing. In preprocessing stage, we first construct the preferred routing tracks to make droplets route orderly on these tracks thereby minimizing the used cells and reduce the possibility of congestion. To determine routing priority of droplets, we propose a routing-resource-based equation to determine the routing order of droplets for better routability. In intra-contamination aware routing stage, a k-shortest path algorithm is proposed to reduce the contaminated spots within one subproblem. When all droplets are routed sequentially to their targets, a dynamic-programming-based routing compaction technique is presented to minimize the execution time of bioassay. Since the contamination problem may occur during these routing stages, to prevent contamination problem from interfering the assay outcomes, wash droplets must be appropriately scheduled and routed with reasonable routing time and used cells. In inter-contamination aware routing stage, to avoid the timing overhead which is caused by scheduling wash operation between successive subproblems, we propose a look-ahead routing scheme that simultaneously considers the contaminated spots within one subproblem and successive subproblems. To handle these contaminated spots, a minimum cost circulation (MCC) algorithm is adopted to simultaneously clean these contaminated spots while minimizing the used cells and the execution time. When all the wash droplets are routed, a final solution compaction technique is applied to minimizing the completion time of assay.

4. PREPROCESSING STAGE

4.1 Preferred Routing Tracks Construction

The goal of droplet routing in a DMFB is to find an efficient schedule for each droplet from its source to target while both fluidic and timing constraints are satisfied. Furthermore, it is desirable to minimize the execution time among all droplets and the number of unit cells used during routing for better reliability and fault tolerance. However, with the increased design complexities, there may exist substantial macros (ex. mixer) that cause fatal routability problems. Moreover, as multiple droplets are routed in a time-multiplex manner, violations of fluidic constraints occur frequently, involving deadlock or detour overhead that increase the used unit cells and completion time of DMFBs. To remedy these deficiencies, we first construct the preferred routing tracks by analyzing each droplet's preferred moving direction and make droplets route on these specific tracks in order. Consider a droplet d_i located at (x_s^i, y_s^i) and its target located at (x_t^i, y_t^i) in a two-dimensional microfluidic array (if Cartesian coordinate system is used); we define the preferred moving vector $\overrightarrow{e^i}$ to be the connection from the point (x_s^i, y_s^i) to the point (x_t^i, y_t^i). Since droplets can only move horizontally or vertically in a microfluidic array, to obtain precise routing tracks information, we decompose $\overrightarrow{e^i}$ into two linearly dependent vectors $\overrightarrow{e_x^i}$ and $\overrightarrow{e_y^i}$ along the x and y axis. Then, we define the preferred routing tracks on rows and columns as follows:

- Track on row r: $\overrightarrow{tk_r} = \sum_{\forall B_i \cap r \neq \phi} \overrightarrow{e_x^i}$
- Track on column c: $\overrightarrow{tk_c} = \sum_{\forall B_i \cap c \neq \phi} \overrightarrow{e_y^i}$

By analyzing the preferred moving vector of each droplet in the bounding box, we can construct the preferred routing tracks. Since the fluidic constraints require a minimum spacing between two droplets, the preferred routing tracks are constructed on non-adjacent rows and non-adjacent columns.

The intuition behind our preferred routing tracks construction is similar to traffic control, as each droplet can be regarded as a car. If most of the cars have common driving direction on the global routing track, we will assign this routing track to the preferred driving direction, which is beneficial to the traffic control (avoid the conflicts of fluidic constraints). If these cars did not drive on the preferred routing track, they can still drive the alternative shoulders adjacent to it or even the opposite direction of it, but they will be charged for additional costs (minimize the used unit cells).

We model the routing resource as a routing graph $G = (V, E)$. A node in the routing graph represents a cell in the microfluidic array, whereas an edge denotes the connection between two adjacent cells. We define the routing result that from v_s to v_t as $R_{(v_s,v_t)} = \{c \in V | c$ is the cell chosen for routing$\}$. For the traffic control, we define the cost function of the droplet routing result $R_{(v_s,v_t)}$ as follows:

$$Cost(R_{(v_s,v_t)}) = \sum_{c \in R_{(v_s,v_t)}} (\alpha \cdot C_{legal} + \beta \cdot C_{illegal} + \gamma \cdot C_{shoulder}) \quad (1)$$

where α, β, and γ are user-specified parameters. C_{legal}, $C_{illegal}$, and $C_{shoulder}$ are the cost of the cell that is along the preferred routing tracks, against the preferred routing tracks, and in non-preferred routing tracks associated with d_i, respectively. The goal is to make droplets route along the preferred routing tracks and find the minimum cost path. If there is a tie in terms of cost, we encourage it to share the paths taken by the previous droplets to improve routability as well as fault tolerance.

Motivated from [4], we also implement the feature of routing concession control. Once a droplet is routed to its target cell, it will be frozen and become a 3×3 blockage till the experiment is finished. This phenomenon will cause lots of congestion regions and have detrimental effects on satisfying both fluidic and timing constraints. To increase the flexibility during routing, we will route the droplet to the available cells that are adjacent to its target cell (named A-cells) instead of its target cell for concession control. Thus, if a routed droplet d_j located in its A-cell v_{d_j} that blocks the routing path of the droplet d_i, we can move d_j from v_{d_j} to the concessive cell v_c while minimizing the routing cost $\delta \cdot Cost(R_{(v_{d_j},v_c)})$ where δ is user-defined constant to penalize the routing detour. The concessive cell is the available cell that is away from the congested region and it can be found by using maze searching.

4.2 Routing Priority Calculation

A key issue in the droplet routing problem is the determination of the droplet routing order. If droplets route in disorder, it will cause fatal routability problem, which increases the routing complexity. To solve this problem, we propose a routing-resource-based equation which considers the congestion of routing region globally and interference with other nets to determine the routing order of droplets for better routability. We first define the available routing resource Res_i^+ and unavailable routing resource Res_i^- for droplet d_i as follows:

$$Res_i^+ = |B_i| + |\{E_5(t_j) \cap B_i\}/E_5(t_i)|, \forall d_j \in D/d_i \quad (2)$$

$$Res_i^- = |Blk \cap B_i| + |\{E_3(s_j) \cap B_i\}/E_3(s_i)|, \forall d_j \in D/d_i \quad (3)$$

where $E_5(t_j)$ represent the 5×5 cell set center by the location of target t_j and $E_3(d_j)$ represent the 3×3 cell set center by the location of droplet d_j. Then, we define the routing-resource-based equation as follows:

$$Res_i^{eq} = \frac{Res_i^+ - Res_i^-}{Res_i^+} \quad (4)$$

The intuition behind the definition of the routing resource can be described as follows. The available cells inside the bounding box B_i are possibly used frequently when routing d_i. We attempt to route droplet d_i first that have many target cells t_j inside the bounding box B_i due to these target cells will become blockages if their droplets d_j are routed. On the contrary, the cells of blockages and unrouted droplets inside the bounding box B_i of droplet d_i will have detrimental effects on routability due to the module blockage and fluidic constraints.

To determine the routing order, we calculate the Res_i^{eq} for each un-routed droplet d_i, then the maximum one represents that it is less congested inside its routing region and has the highest routing priority. In the detail implementation, we use a priority queue Q_{eq} to find the routing order of un-routed droplets efficiently, where the priority of Q_{eq} is the Res_i^{eq} of each un-routed droplets. We iteratively pop a un-routed droplet with the highest Res_i^{eq} to route, and update the Res_j^{eq} of un-routed droplets d_j left in Q_{eq}.

5. INTRA-CONTAMINATION AWARE ROUTING STAGE

5.1 Routing Path Modification by K-Shortest Path

Although the above proposed scheme can schedule a high throughput routing paths in the 2D plane. However, it may cause a large amount of contaminated spots. This is because the droplets are orderly routed on these preferred routing tracks, multiple droplets may be routed along the same tracks to minimize the used cells. Therefore, contamination occurs and causes erroneous outcome for bioassay. To overcome these drawbacks, we proposed a k-shortest-path-based algorithm [5] to reduce the number of contaminated spots. As the aforementioned algorithm, we iteratively select an un-routed droplet d_i with the highest Res_i^{eq} from Q_{eq}. We then model the routing path from d_i to its target t_i as $R_{(d_i,t_i)}$ and search a minimum cost path. After d_i is routed, we push d_i into a priority queue Q_{cs}, where the priority is the number of contaminated spots within $R_{(d_i,t_i)}$. And we update the number of contaminated spots for the droplet d_j $(j \neq i)$ in Q_{cs}, where $R_{(d_i,t_i)} \cap R_{(d_j,t_j)} \neq \phi$. Iteration terminates until all droplets are routed.

When all droplets are routed, we iteratively select a routed-droplet d_i with the highest number of contaminated spots from Q_{cs}. We then remodel the cost of the edge $(v_x, v_y) \in R_{(d_i,t_i)}$ with a higher value for penalty, where node v_x and v_y are contaminated spots. Based on the remodeled routing graph, we adopt the k-shortest path algorithm to reduce the number of contaminated spots. For example, Figure 3 (a) shows the original routing paths and the routing path of d_2 has the highest number of contaminated spots. We then select d_2 by adopting the k-shortest path algorithm to the remodeled routing graph. As shown in Figure 3 (b) and (c), when k is set to be one, the number of contaminated spots is reduced to five; when k is set to be two, the number of contaminated spots is reduced to two. There are two essential ideas behind our k-shortest path algorithm. Since we still want to make the droplet route on tracks to minimize the used cells, the k-shortest path algorithm can be adopted to find the k-minimum cost path. However, if we only modify the original path once $(k = 1)$, it may still overlap with other routed path as shown in Figure 3 (b), which causes only slightly reduction of contaminated spots. Thus, it is necessary to find another minimum cost path to avoid this situation. In this paper, the value of k is set to be three.

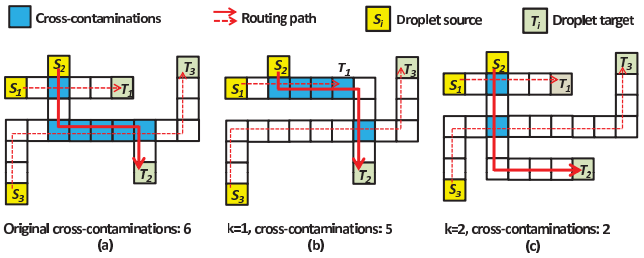

Figure 3: Adopt the k-shortest path algorithm to reduce the cross section. (a) The original routing solution of d_2. (b) The first shortest path of d_2 in the remodeled routing graph. (c) The second shortest path of d_2 in the remodeled routing graph.

5.2 Routing Compaction by Dynamic Programming

One of the major challenges is to convert the routing paths from sequential to concurrent manner. Since all droplets are routed to A-cells sequentially in the order determined by the routing-resource-based equation, however, the completion time may violate the timing constraint. Thus, it is necessary to perform routing compaction that minimize the execution time for fast bioassay execution and better reliability. Another difficult challenge is the determination of the occurring clock cycle of contaminated spots. Since contamination is likely to occur when multiple droplet routes cross or overlap with each other. At the contaminated spot, a droplet that arrives at a later clock cycle can be contaminated by the residue left behind by another droplet that passed through at an earlier clock cycle. Therefore, to obtain simultaneous wash operations within the droplet routing, it is desirable to obtain the occurring clock cycle of each contaminated spot. To achieve these goals, we propose a dynamic programming based approach to compact the 2D routing and determine the occurring clock cycle of each contaminated spot. There are at least two advantages by using this approach. First, the original routing paths can be preserved on the preferred routing tracks to minimize the routing detour and the used cells for better bioassay reliability. Second, an efficient and robust routing scheduling for concurrent routing among droplets can be derived due to the elegant property of dynamic programming. Since it is hard to directly apply the 2D routing paths to routing compaction, we encode each routing path into a corresponding 1D moving string by using four direction characters u, d, l, and r, where represent up, down, left, and right, respectively. Generally, consider two moving strings MS_1 and MS_2, the routing compaction problem is transformed to minimize the length of the compacted string without violating any fluidic constraints. To characterize the optimal substructure of the compacted string, we define $C[i][j]$ to be the compacted string length using i prefixes of MS_1 and j prefixes of MS_2. The optimal substructure of the routing compaction gives the following recursive formula.

$$C[i][j] = \begin{cases} min\{C[i-1][j], C[i][j-1], C[i-1][j-1]\} + 1 & , if\ legal \\ \infty & , otherwise \end{cases} \quad (5)$$

Based on the routing order, we first encode the first routing path P_1 into the corresponding moving string. Then we iteratively encode the next routing path, and compact it with the previous compacted moving string. During the compaction, we should check the legality of the compacted string, that is, it should satisfy the fluidic constraints for each droplet. If it is legal to compact MS_1 and MS_2, the optimal string length $C[i][j]$ will be the minimum length among $C[i-1][j]$, $C[i][j-1]$, and $C[i-1][j-1]$, which are stored in the table previously, plus the unit length. Otherwise, it will set to be infinite. Finally, iteration terminate when all moving string are compacted.

5.3 Minimum Cost Circulation Flow Technique

Wash droplets can remove the liquid residue and non-target molecules that have been adsorbed onto the surface to prevent the contamination problem. However, if we only adopt one wash droplet to perform the wash operation, the overloading of contaminated particles may reduce the purity of wash droplet thereby decreasing the washing rate. Furthermore, wash droplets are sensitive to the environmental temperature and may suffer from the evaporation problem due to the long execution time. Therefore, it is desirable to design a wash-droplet routing algorithm that fully utilizes multiple wash droplets to shorten the execution time for assay thereby increasing the reliability of DMFBs. Therefore, we propose the first minimum-cost-circulation-based algorithm (MCC) for optimal wash-droplet routing to simultaneously minimize used cells and the cleaning time.

5.3.1 Introduction to Minimum Cost Circulation Problem

We first briefly introduce the minimum cost circulation problem (MCC). The circulation problem and its variants is a generalization of network flow problems, with the added constraint of a lower bound on edge flows, and with flow conservation also being required for the source and sink. Each arc has an associated cost and the objective function is to find a feasible flow through this graph with the minimum cost, such that the sum over all arcs of the multiplication of the flow in each arc and the corresponding cost is minimum. Given a flow graph $G_f = (V_f, E_f)$, the minimum cost circulation problem can be represented as follows:

$$Minimize: \quad z = \sum_{(i,j)\in E_f} C_{ij} \cdot x_{ij}$$

$$Subject\ to: \quad Bounded\ constraint: \quad l_{ij} \leq x_{ij} \leq u_{ij},\ \forall (i,j) \in E_f$$

$$Conservation\ constraint: \quad \sum_{(i,j)\in E_f} x_{ij} = \sum_{(j,i)\in E_f} x_{ji}$$

where l_{ij} (u_{ij}) represent the lower (upper) bound on flow from node i to j, C_{ij} represent the cost per unit flow from node i to j, and x_{ij} represent the flow value from node i to j. The MCC problem can be solved in strongly polynomial algorithm by using linear programming or more commonly by using faster and more efficient network algorithms. In this paper, we solve the MCC problem by using negative cycles canceling techniques [6].

5.3.2 Circulation Flow Formulation

The key concept behind our minimum cost circulation formulation is to schedule a routing method for high throughput wash operations. The most difficult challenge is to model the contaminated spots into the flow constraints and formulate correct wash operations. To ensure the real-time response and reduce the time-to-result effects, we should minimize the cleaning time used for wash operations. Furthermore, to improve the fault-tolerance of DMFBs, we

should minimize the used cells for wash-droplet routing. Consider the multi-objective optimizations, we propose the MCC-based algorithm to solve the wash-droplet routing problem. We first construct the flow graph $G_f = (V_f, E_f)$, where V_f is the set of nodes, and E_f is the set of edges. Different from the general network flow problem, each edge (i, j) in the flow graph G_f can be represented as a 3-tuple (l_{ij}, u_{ij}, C_{ij}), where l_{ij} (u_{ij}) denote the lower (upper) bound on the flow of this edge, and C_{ij} represent the associated cost per unit flow on this edge. As shown in Figure 4, there are four major phases in our MCC formulation, which are a dummy source, set of wash droplets, set of contaminated spots, and a sink (reservoir), respectively. There are two basic assignments and two formulation rules in our MCC formulation, and we assume that the subproblem SP_p is under formulation.

Assignment 1: Node Capacity Assignment

The most important issue is that the contaminated spot should be cleaned by the wash droplet to prevent the contamination problem. To perform the wash operation correctly (e.g. to schedule the wash droplet to pass through the node), we use the node split technique [1] to solve the contamination problem. We decompose each node $v_i \in CS$ into two intermediate nodes v_{in}^i and v_{out}^i, and an edge is connected from v_{in}^i to v_{out}^i. Since we formulate the wash operation as the flow problem, to ensure at least one wash droplet to pass through the node v_i, the 3-tuple of the edge $(v_{in}^i, v_{out}^i$ is set to be $(1, \infty, 0)$. Note that u_{ij} is set to ∞ due to multiple wash droplets can pass through the same contaminated spot.

Assignment 2: Edge Cost Assignment

Another important issue is to determine the routing path of wash droplets. We also make the wash droplets route on the preferred routing tracks to minimize the used cells. Therefore, we use the same model $R_{(v_i,v_j)}$ which is discussed in section 4.1 to represent the routing path from node v_i to v_j, and the associated cost of this edge (v_i, v_j) is $Cost(R_{(v_i,v_j)})$.

After the two basic assignments, the circulation flow can be constructed by the following two flow formulation rules.

Rule 1: Timing-Based Transitive Topology

Within one subproblem, the contamination happens in a cell when a droplet arrives at a later clock cycle that is contaminated by the residue left behind by another droplet that passed through at an earlier clock cycle. Before a droplet arrive at the cell that is contaminated by the previous droplet, it is desirable to wash the surface to avoid incorrect diagnosis outcomes. By adopting the routing compaction technique, the occurring time and location of contaminated spots can be determined. We use a 2-tuple $(v_i, \hat{t_i})$, to denotes the occurring time of contaminated spot v_i. Given a set of contaminated spots in CS_p, $(v_1, \hat{t_1})$, $(v_2, \hat{t_2})$, ... $(v_n, \hat{t_n})$, we construct the timing based transitive topology in the following descriptions.

(a)Timing-based topology: For each pair (v_i, v_j) where $\{v_i, v_j\} \in CS_p$, there exists a edge connection from v_{out}^i to v_{in}^j if and only if $\hat{t_i} \leq \hat{t_j}$. We assign the associated 3-tuple of the edge by $(0, \infty, Cost(R_{(v_i,v_j)}))$.

(b)Transitive closure: For each triple $(v_i, v_k, v_j) \in CS_p$, if there exists edge connections from v_{out}^i to v_{in}^k and from v_{out}^k to v_{in}^j, there also exists a edge connection from v_{out}^i to v_{in}^j. We assign the associated 3-tuple of the edge by $(0, \infty, Cost(R_{(v_i,v_j)}))$.

Contamination problem occurs when two droplet pass through the same spot in different clock cycle, the timing-based topology shows that the wash operation should be performed before the second droplet arrives at this contaminated spot. The transitive closure property allows the multiple wash droplets to perform the wash operations, while the timing-based topology should be also followed.

Rule 2: Connection Strategy Between Phases

There are four major phases in our MCC formulation. We first construct edge connections from the dummy source to wash droplets and the associated 3-tuple of each edge is set to $(0, 1, 0)$. From the second phase to the third phase, for each wash droplet $w_j \in W$ and $v_i \in CS_p$, we connect w_j with v_{in}^i and assign the associated 3-tuple of this edge by $(0, 1, Cost(R_{(v_{w_j},v_i)}))$, where v_{w_j} is the location of wash droplet w_j. Then, for each node $v_i \in CS_p$, there exists an edge connection from v_{out}^i to the sink v_{sink} (waste reservoir), with the associated 3-tuple $(0, \infty, Cost(R_{(v_i,v_{sink})}))$. Finally, since we should use at least one wash droplet to perform the wash operation, we connects the sink back to the dummy source, with the associated 3-tuple $(1, 4, 0)$. Note that in this paper, the maximum number of available wash droplets is four.

Figure 4 shows the flow construction. Based on the aforementioned basic assignments (node capacity and edge cost) and the two flow formulation rules, we have the following two theorems.

THEOREM 1. *There exists a feasible solution under the two basic assignments and two flow formulation rules.*

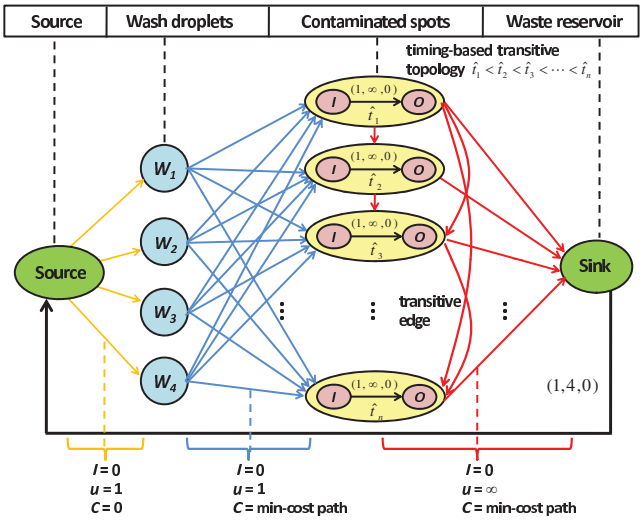

Figure 4: Illustration of our minimum cost circulation construction.

THEOREM 2. *Under the proposed flow construction, we can adopt the minimum cost circulation algorithm to schedule optimal and correct wash operations.*

6. INTER-CONTAMINATION AWARE ROUTING STAGE

6.1 Look-Ahead Routing Scheme

Although wash droplets clean the intra-contaminations within one subproblem optimally by using the MCC-based algorithm, some contaminations that treated as blockages for the next subproblem (i.e. inter-contaminations) may incur timing overhead since extra execution time for wash operations should be perform between successive subproblems. To remedy these deficiencies, we propose a look-ahead routing scheme that predicts the inter-contaminations for the next subproblem then simultaneously clean both intra- and inter-contaminations by using the MCC-based algorithm to reduce the execution time significantly. For example, before cleaning the intra-contamination of subproblem SP_i by using wash droplets, we also derive the routing solution of subproblem SP_{i+1}. Based on both routing solutions, the inter-contaminations between subproblem SP_i and SP_{i+1} can be derived and are handled with intra-contaminations in subproblem SP_i.

6.1.1 *Travelling Salesman Problem Formulation*

Consider each edge (v_i, v_j) in the flowing graph, we perform the wash operation for those inter-contamination nodes $v_k \in LA(v_i, v_j)$. Since each v_k should be cleaned by a wash droplet, which means a wash droplet should pass through all the nodes v_k. Furthermore, we also encourage the wash droplet to route on the preferred routing tracks to minimize the total used cells. To tackle these problems, we construct a TSP graph $G_{tsp} = (V_{tsp} \cup \{v_i, v_j\}, E_{tsp})$, where the vertex set of V_{tsp} is the node $v_k \in LA(v_i, v_j)$, the E_{tsp} is the edge set, and the v_i (v_j) is the starting (ending) point of the TSP graph. We connect each pair (v_{k_1}, v_{k_2}) where $\{v_{k_1}, v_{k_2}\} \in V_{tsp}$, with the associated cost $Cost(R_{(v_{k_1},v_{k_2})})$. Then, we connect the starting point v_i with each node v_k and connect each node v_k with the ending point v_j. The goal is to schedule a minimum cost routing path of the wash droplet that starts from v_i to v_j while passing through all the nodes v_k, which is a variant of travelling salesman problem (TSP). Assume the optimal routing path is $R_{(v_i,v_j)}^{tsp}$ and the corresponding minimum cost is $Cost(R_{(v_{k_1},v_{k_2})}^{tsp})$. We reformulate the cost of the edge (v_i, v_j) in G_f by $Cost(R_{(v_{k_1},v_{k_2})}^{tsp})$, and the corresponding routing path of (v_i, v_j) is $R_{(v_{k_1},v_{k_2})}^{tsp}$. The dynamic-programming-based approach for solving the TSP problem is used in this paper [2].

After the wash droplets are scheduled, we compact the routing paths of wash droplets with the previous compacted paths by using the same dynamic programming technique. In addition to checking the fluidic constraints, we add the contamination constraint to ensure that the wash droplet should arrive before the contaminated spots.

7. EXPERIMENTAL RESULTS

TABLE II: STATISTICS OF THE ROUTING BENCHMARKS

Circuit	Size	#Sub	#Net	#D_{max}	#W
in-vitro_1	16 x 16	11	28	5	4
in-vitro_2	14 x 14	15	35	6	4
protein_1	21 x 21	64	181	6	4
protein_2	13 x 13	78	178	6	4

We have implemented our contamination aware droplet router in the C++ language on a 2-GHz 64-bit Linux machine with 8GB memory. We perform experiments to verify the efficiency and effectiveness of our algorithm on four widely used bioassays from [9, 11]. Table II shows the statistics of each benchmark. In the table, "Size" gives the size of microfluidic array, "#Sub" denotes the number of subproblems, "#Net" represents the number of nets, "D_{max}" gives the maximum number of droplets among subproblems, "#W" denotes the number of wash droplets.

In the first experiment, we evaluated the reduction of intra-contaminations by using the k-shortest path technique. Table III shows the comparison results of our algorithm without and with the k-shortest path technique. By applying the k-shortest path technique, our algorithm not only reduced the intra-contaminations by 48%, but also reduced the respective used cells and execution time by 8% and 13% with small increase in the CPU time.

In the second experiment, we evaluated the reduction of the execution time by using the look-ahead routing scheme. Table IV shows the comparison results of our algorithm without and with the look-ahead routing scheme. The look-ahead routing scheme predicts the inter-contaminations for the next subproblem then simultaneously cleans both intra- and inter-contaminations by using the MCC-based algorithm to reduce the execution time significantly. By applying the look-ahead routing scheme, our algorithm not only reduced the execution time by 9%, but also reduced the used cells by 11% with small increase in the CPU time.

In the third experiment, we compared the effectiveness of our contamination aware droplet routing algorithm with the state-of-the-art algorithm [12] in Table V. For comparison purpose, we implement the state-of-the-art algorithm [12]. Overall, our algorithm reduced the respective used cells and execution time by 28% and 12% with small increase in the CPU time. The increase in the intra-contaminations is as expected, because our droplet router has to route on the preferred routing tracks to minimize the usage of used cells. By our contamination aware droplet routing algorithm, we can effectively clean all contaminations by optimal wash-droplet routing while minimizing the used cells and the execution time for better reliability and fast bioassay execution.

TABLE III: COMPARISONS OF OUR ALGORITHM WITHOUT AND WITH APPLYING K-SHORTEST PATH TECHNIQUE

Bioassay	Ours (non k-SP)				Ours (k-SP)			
	#C_{intra}	#UC	T_{exe}	CPU	#C_{intra}	#UC	T_{exe}	CPU
in-vitro_1	53	388	225	0.18	21	351	193	0.58
in-vitro_2	27	291	217	0.13	5	281	191	0.39
protein_1	138	2418	1592	1.47	82	2213	1394	2.58
protein_2	106	1453	1280	0.71	61	1362	1108	1.49
Total	324	4550	3314	2.49	169	4207	2886	5.04

■ #C_{intra}: The number of intra-contaminations. ■ #UC: The number of used cells for routing.
■ T_{exe}: The execution time for the bioassays. ■ CPU: The CPU time (sec).

8. CONCLUSION

In this paper, we proposed a contamination-aware droplet routing algorithm for DMFBs. To reduce the routing complexities and used cells, we first construct preferred routing tracks by analyzing the global moving vector of droplets to guide the droplet routing. To cope with contaminations within one subproblem, we first apply a k-shortest path routing technique to minimize the contaminated spots. Then, to take advantage of multiple wash droplets, we adopt a minimum cost circulation algorithm (MCC) for optimal wash-droplet routing to simultaneously minimize used cells and the cleaning time. Furthermore, a look-ahead prediction technique is used to determine the contaminations between successive subproblems. After that, we can simultaneously clean both contaminations within one subproblem and those between successive subproblems by using the MCC-based algorithm to reduce the execution time significantly. Based on four widely used bioassays, our algorithm reduced the used cells and the execution time significantly compared with the state-of-the-art algorithm.

TABLE IV: COMPARISONS OF OUR ALGORITHM WITHOUT AND WITH APPLYING LOOK-AHEAD ROUTING SCHEME

Bioassay	Contaminations		Ours (non look-ahead)			Ours (look-ahead)		
	#C_{intra}	#C_{inter}	#UC	T_{exe}	CPU	#UC	T_{exe}	CPU
in-vitro_1	21	19	446	227	0.32	351	193	0.58
in-vitro_2	5	8	267	210	0.24	281	191	0.39
protein_1	82	190	2493	1569	2.11	2213	1394	2.58
protein_2	61	141	1498	1172	0.47	1362	1108	1.49
Total	169	358	4704	3178	3.14	4207	2886	5.04

■ #C_{intra}: The number of intra-contaminations. ■ #UC: The number of used cells for routing.
■ #C_{inter}: The number of inter-contaminations. ■ T_{exe}: The execution time for the bioassays.
■ CPU: The CPU time (sec).

TABLE V: COMPARISONS BETWEEN [12] AND OUR ALGORITHM

Bioassay	[12]				Ours			
	#C_{intra}	#UC	T_{exe}	CPU	#C_{intra}	#UC	T_{exe}	CPU
in-vitro_1	4	621	268	0.06	21	351	193	0.58
in-vitro_2	0	423	224	0.03	5	281	191	0.39
protein_1	18	3215	1508	0.23	82	2213	1394	2.58
protein_2	11	1574	1287	0.14	61	1362	1108	1.49
Total	33	5833	3287	0.46	169	4207	2886	5.04

■ #C_{intra}: The number of intra-contaminations. ■ #UC: The number of used cells for routing.
■ T_{exe}: The execution time for the bioassays. ■ CPU: The CPU time (sec).

9. ACKNOWLEDGMENT

The authors would like to thank Yang Zhao and Prof. Krishnendu Chakrabarty of the Duke University for providing the authors with benchmarks and manuscripts for the comparative studies.

10. REFERENCES

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[2] R. Bellman, "Dynamic programming treatment of the travelling salesman problem," *J. ACM*, pp. 61–63, Jan. 1962.

[3] K. F. Böhringer, "Modeling and controlling parallel tasks in droplet based microfluidic systems," *IEEE Trans. on CAD*, vol. 25, no. 2, pp. 334–344, Feb. 2006.

[4] M. Cho and D. Z. Pan, "A high-performance droplet routing algorithm for digital microfluidic biochips," *IEEE Trans. on CAD*, vol. 27, no. 10, pp. 1714–1724, Oct. 2008.

[5] D. Eppstein, "Finding the k shortest paths," *Proc. IEEE FOCS*, pp. 154–165, Feb. 1994.

[6] A. V. Goldberg and R. E. Tarjan, "Finding minimum cost circulations by canceling negative cycles," *J. ACM* , pp. 873–886, Oct. 1989.

[7] E. J. Griffith, S. Akella, and M. K. Goldberg, "Performance characterization of a reconfigurable planar-array digital microfluidic system," *IEEE Trans. on CAD*, vol. 25, no. 2, pp. 345–357, Feb. 2006.

[8] F. Su, K. Charkrabarty, and R. B. Fair, "Microfluidics based biochips: Technology issues, implementation platforms, and design-automation challenges," *IEEE Trans. on CAD*, vol. 25, no. 2, pp. 211–223, Feb. 2006.

[9] F. Su, W. Hwang, and K. Chakrabarty, "Droplet routing in the synthesis of digital microfluidic biochips," *Proc. IEEE/ACM DATE*, pp. 1–6, Mar. 2006.

[10] T. Xu and K. Chakrabarty, "Automated design of digital microfluidic lab-on-chip under pin-count constraints," *Proc. ACM ISPD*, pp. 90–98, Apr. 2008.

[11] P. H. Yuh, C. L. Yang, and Y. W. Chang, "BioRoute: A network flow based routing algorithm for the synthesis of digital microfluidic biochips," *IEEE Trans. on CAD*, vol. 27, no. 11, pp. 1928–1941, Nov. 2008.

[12] Y. Zhao and K. Chakrabarty, "Cross-contamination avoidance for droplet routing in digital microfluidic biochips," *IEEE/ACM DATE*, Apr. 2009.