A Contamination Aware Droplet Routing Algorithm for the Synthesis of Digital Microfluidic Biochips

Tsung-Wei Huang, Chun-Hsien Lin, and Tsung-Yi Ho, *Member, IEEE*

***Abstract*—Recent advances of digital microfluidic biochips (DMFBs) have revolutionized the traditional laboratory procedures. By providing the droplet-based system, DMFB can perform real-time biological analysis and safety-critical biomedical applications. However, different droplets being transported and manipulated on the DMFB may introduce the contamination problem caused by liquid residue between different biomolecules. To overcome this problem, a wash droplet is introduced to clean the contaminations on the surface of the microfluidic array. However, current scheduling of wash droplet does not restrict the extra used cells and execution time of bioassay, thereby degrading the reliability and fault-tolerance significantly. In this paper, we propose a contamination aware droplet routing algorithm for DMFBs. To reduce the routing complexity 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 (MCC) algorithm for optimal wash-droplet routing to simultaneously minimize used cells and the cleaning time. Since the droplet routing problem consists of several subproblems, a look-ahead prediction technique is further 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 significantly. 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.**

***Index Terms*—Biochips, contaminations, dynamic programming, minimum cost circulation (MCC), routing.**

I. INTRODUCTION

DIGITAL microfluidic biochip (DMFB) is an emerging technology that aims to miniaturize and integrate droplet-handling on a chip. By handling fluidics with micro-volumes, the DMFB provides higher sensitivity and less human errors than the conventional laboratories. Furthermore, the miniaturization and automation offer less reagent consumption and more flexible control. Due to these advantages, DMFBs are expected to revolutionize many biological processes, especially for the immediate point-of-care diagnosis of diseases.

Fig. 1(a) shows the schematic view of a DMFB. A DMFB contains three components, the 2-D microfluidic array, the dispensing ports/reservoirs, and the optical detectors. The 2-D microfluidic array contains a set of basic cells which consist of two parallel glass plates [see Fig. 1(b) and (c)]. The bottom plate contains a patterned array of individually controllable electrodes, and the top plate is coated with a continuous ground electrode. The droplet is sandwiched between the two plates and moves within the filler medium. By independently controlling the voltage of electrodes, droplets can be moved along this line of electrodes due to the principle of electrowetting on dielectric (EWOD) [5], [15]. Therefore, many fluidic operations such as mixing and dilution can be performed anywhere in the 2-D array within different time intervals. Specifically, such operations are executed in the form of fluidic modules (i.e., virtual devices), that are simply obtained by grouping adjacent cells together [5]. This characteristic is also referred to as the *reconfigurability* [7]. Besides the 2-D microfluidic array, there are on-chip reservoirs, dispensing ports, and optical detectors. The dispensing ports are responsible for droplet generation while the optical detectors are used for droplet detection.

Recently, many on-chip laboratory procedures, such as immunoassay, real-time DNA sequencing, and protein crystallization, have all been successfully demonstrated on DMFBs. Researchers addressed these procedures with architectural-level and physical-level synthesis [14], [16]. In the architectural-level synthesis, the goal is to schedule the assay functions and bind them to a given number of resources so as to maximize the parallelism, thereby decreasing the process time. During the physical-level synthesis, modules of these resources must be placed in the 2-D microfluidic array to minimize the entire chip area [17], [20], [24]. Then the droplets must be appropriately routed to perform these prescheduled biological operations (e.g., sample mixing or detection) [21], [26].

Droplet routing on biochips is a key design issue in the physical-level synthesis, which schedules the movement of each droplet in a time-multiplexed manner. This physical synthesis is one of the most critical design challenges due to high

Manuscript received December 3, 2009; revised February 25, 2010 and June 8, 2010; accepted June 11, 2010. Date of current version October 20, 2010. This work was supported in part by the National Science Council of Taiwan, under Grant NSC 98-2220-E-006-013. A Preliminary version of this paper was presented at the 2009 IEEE/ACM International Conference on Computer-Aided Design (ICCAD'09), San Jose, CA, November 2009. This paper was recommended by Associate Editor K. Chakrabarty.

The authors are with the Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan 701, Taiwan (e-mail: twhuang@eda.csie.ncku.edu.tw; f7495134@mail.ncku.edu.tw; tyho@csie.ncku.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2010.2062770

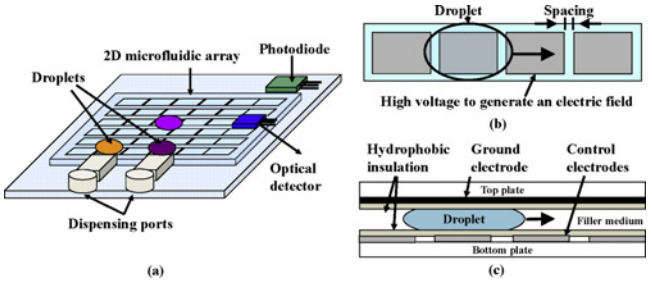

Fig. 1. DMFB [21]. (a) Schematic view of a DMFB. (b) Top view of the 2-D microfluidic array. (c) Side view of the 2-D microfluidic array.

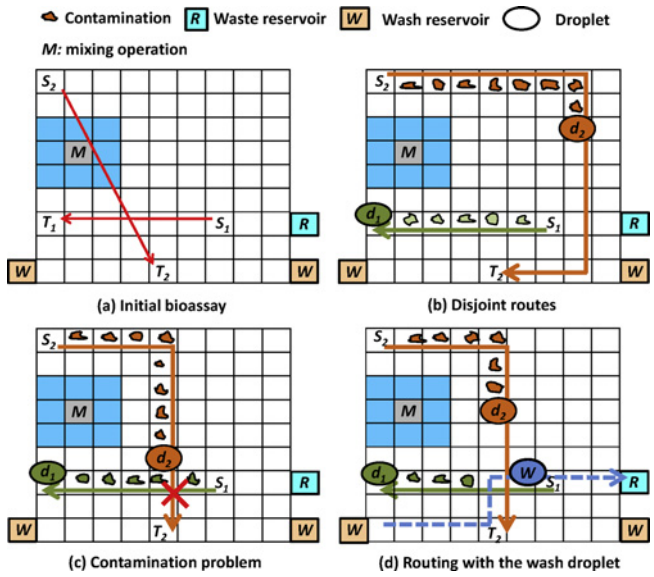

Fig. 2. Illustration of the contamination aware droplet routing. (a) Initial bioassay. (b) Disjoint routing for contamination avoidance. (c) Shortest path routing with a contaminated spot. (d) Wash-droplet routing.

complexity as well as large impacts on assay performance. During the droplet transportation, molecules and substances carried in droplets may potentially leave traces on the microfluidic array, which causes the contamination problem [13], [22], [27]. The problem occurs more frequently in many protein assays, since proteins tend to adsorb the hydrophobic surface and contaminate it [13], [22]. The particles and liquid residues left behind the microfluidic array potentially lead to an erroneous assay outcome. Moreover, the contaminations left between two adjacent electrodes may cause electrode short problems, which result in physical defects and produce incorrect behaviors in the electrical domain [6]. Although a filler medium, such as silicone oil, has been advocated to prevent contaminations, it has been proved that it is not sufficient for many types of proteins and heterogeneous immunoassays [9], [13].

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 allows more and more biological operations to be performed on a DMFB, finding disjoint routes has become more and more difficult. Furthermore, disjoint routes also restrict the spare cells needed for replacing faulty primary cells and ensuring the correctness of bioassay execution. Hence, the fault tolerance of bioassay is significantly reduced [23].

To cope with the contamination problem, a wash droplet is introduced to clean the contaminated spots on the surface of the microfluidic array. Let us consider an initial bioassay with two droplets d_1 and d_2, a mixer, and peripheral devices (e.g., reservoirs) as shown in Fig. 2(a). If we adopt the disjoint routes to avoid the contamination problem as shown in Fig. 2(b), the execution time and the number of used cells for routing d_1 and d_2 are 18 and 26, respectively (a droplet moves to one cell in one clock cycle). In Fig. 2(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 through this contaminated spot. As shown in Fig. 2(d), to ensure the correctness of wash operation, the wash droplet must clean the contaminated spot in the time interval $(\hat{t}^1_{cs}, \hat{t}^2_{cs})$, where $\hat{t}^1_{cs}$ and $\hat{t}^2_{cs}$ denote the arrival time at the contaminated spot of d_1 and d_2, respectively. If the wash droplet cannot arrive at the contaminated spot before $\hat{t}^2_{cs}$, $\hat{t}^2_{cs}$ 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. Therefore, to achieve fast and correct execution of bioassays, it is desirable to consider the wash operations and droplet routing simultaneously.

Furthermore, since DMFBs are dynamically reconfigurable, a series of 2-D placement configurations are obtained in different time spans during the module placement phase. In this way, the droplet routing problem is divided into a series of subproblems. In each subproblem, the nets to be routed from the source modules to the target modules are determined first. Only the microfluidic modules that are active during this time interval are considered as obstacles in droplet routing. These subproblems are addressed sequentially to obtain a complete solution for the droplet routing [21]. Therefore, contaminated spots occur not only within one subproblem (*intracontaminations*) but also between successive subproblems (*intercontaminations*). Contaminations in the previous subproblem are treated as blockages for the next subproblem. Additional wash droplets are needed to clean the intercontaminations, which may cause timing overhead for bioassays.

The main challenge of contamination aware droplet routing is to ensure the correct execution of bioassays. Unlike the traditional very large-scale integration routing, in addition to routing paths decision, the DMFB routing problem needs to address the issue of droplet scheduling under the practical constraints imposed by the fluidic property and timing restriction of synthesis results. As previously mentioned, when a droplet moves to a cell that is contaminated by a previous droplet, the contamination problem in this cell may cause an unexpected execution error of bioassays. Hence, a contamination constraint is proposed to avoid the contamination problem.

Fast droplet routing time can achieve real-time responses for most safe-critical biosystems and reduce the possibility of the electrode breakdown caused by long actuation voltage. Thus, minimizing the droplet routing time can maintain a reliable

system [21], [25]. On the contrary, a biochip contains primary cells for bioassay execution and spare cells for replacing faulty primary cells to ensure the correct bioassay execution [18]. Therefore, to maximize the number of spare cells for better fault tolerance, the number of cells used for droplet routing should be minimized.

A. Related Prior Work

Droplet routing problem has attracted much attention in the literature recently [3], [4], [11], [21], [25]. The first one is the prioritized A*-search algorithm [3]. By assigning each droplet with a priority, the A*-search algorithm is used to coordinate the routing path of each droplet. However, this method did not address the practical timing issue for throughput consideration. The second one is based on the open shortest path first routing protocol [11]. They use Dijkstra's shortest path algorithm to create a routing table, and define layout patterns of a biochip. Each layout pattern has a routing table which stores the routing information. Then the droplets can be routed based on these routing tables. However, this algorithm restricts the movements of droplets, which allows less reconfigurability of DMFBs. In [21], they proposed a two-stage algorithm. In the first stage, a set of shortest routing paths for each droplet is generated by maze routing. In the second stage, a scheduling heuristic is used to schedule droplets based on selected routing paths. This approach suffers from the initialization overhead either to build routing paths or to discover a feasible routing solution. Recently, a network-flow-based algorithm with negotiation is proposed in [25]. This algorithm adopts a two-stage technique of global routing followed by detailed routing. In the global routing, routing for a set of noninterfering nets is formulated into a network flow model. In the detailed routing, a negotiation-based method iteratively schedules the routing paths for other droplets. The drawback of this approach is that much more spare cells are required for the network flow formulation, which is bottlenecked by the distribution of the module placement. In [4], droplets are first routed by the bypassibility in the 2-D array. Then a greedy approach is introduced to transform the 2-D routing paths into 3-D routing paths. Since the bypassibility is only decided by local information, the proposed manner may suffer from the routability problem. In this case, many biological processes may not be successfully performed.

Although current state-of-the-art algorithms can handle the droplet routing problem, however, these approaches have a common detrimental effect on unrestricted sharing of used cells by various droplet routes. In [27], 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 (intercontaminations), a wash operation is also scheduled between successive subproblems to avoid the contamination problem. However, the extra execution time of the wash operation interrupts the continuous biological reactions between successive subproblems. In this regard, the total execution time of the bioassay may increase and cause a time-to-result effect, which is a practical issue for safe-critical applications requiring real-time response [19]. Fig. 3(a) illustrates the wash strategy for intercontaminations proposed in [27].

B. Our Contribution

In this paper, we propose a contamination aware droplet routing algorithm for DMFBs. Unlike the aforementioned routing algorithm of wash droplets, which cleans intra and intercontaminations separately, our algorithm simultaneously cleans them within one subproblem to reduce the execution time significantly [see Fig. 3(b) for an illustration]. To tackle the complexity issue of concurrently considering droplet routing and wash operations, our algorithm consists of three major stages: preprocessing, intracontamination aware routing, and intercontamination aware routing between successive subproblems. Different from the aforementioned work, our algorithm has the following distinguished features:

1) a global moving vector analysis for constructing preferred routing tracks to reduce the design complexity and minimize the number of used unit cells;
2) a k-shortest path routing technique to minimize the contaminated spots within one subproblem;
3) a routing compaction technique by dynamic programming to determine the time slot of contaminated spots within one subproblem and minimize the execution time of bioassays;
4) a look-ahead prediction technique to determine the contaminated spots between successive subproblems and minimize the total routing time of wash droplets;
5) a minimum cost circulation technique is adopted to simultaneously clean intra and intercontaminations to minimize the used cells and the execution time.

Experimental results demonstrate the robustness and efficiency of our algorithm. The evaluation performed on four widely used bioassays shows the completeness of all the droplet routing without any contamination problem. Our algorithm outperforms the previous work in minimizing the number of used cells for better fault tolerance (28% reduction). We also achieve a 12% fewer execution time cycles to route all assays. The experimental evaluation demonstrates that in terms of used cells and execution time, the proposed algorithm achieves the best results.

The rest of this paper is organized as follows. Section II formulates the contamination aware droplet routing problem, while Section III describes the overview of our algorithm. Our algorithm consists of three major stages: preprocessing, intracontamination aware routing, and intercontamination aware routing, which are given in Sections IV, V, and VI, respectively. Finally, the complexity analysis, experimental results, and the concluding remarks are provided in Sections VII, VIII, and IX.

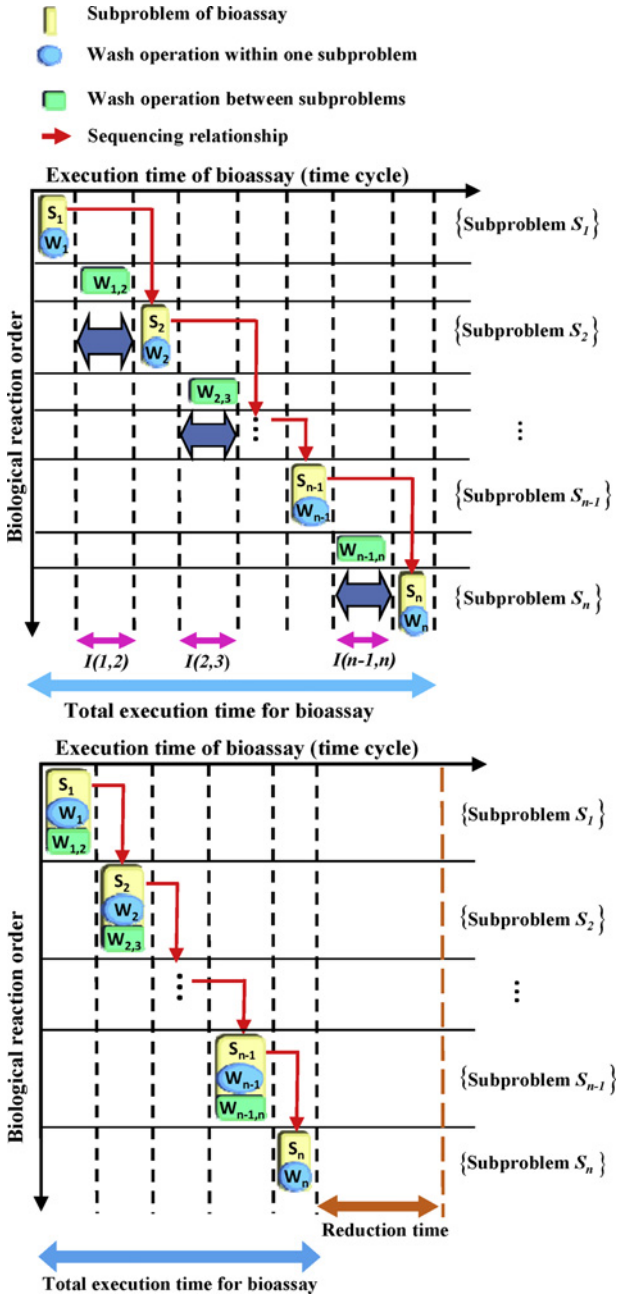

Fig. 3. Illustration of different wash strategy for intercontaminations. (a) Handle intercontaminations between successive subproblems. (b) Handle intra and intercontaminations, simultaneously, within one subproblem.

II. Problem Formulation

There are three routing constraints in contamination aware droplet routing: the fluidic constraints, the timing constraint, and the contamination constraint. The fluidic constraints are used to avoid the unexpected mixing between two droplets of different nets during their transportation. Fluidic constraints can be further divided into the static and dynamic fluidic constraints [21]. Let d_i at $(x_i^{\hat{t}}, y_i^{\hat{t}})$ and d_j at $(x_j^{\hat{t}}, y_j^{\hat{t}})$ denote two independent droplets at time $\hat{t}$. Then, the following constraints should be satisfied for any $\hat{t}$ during routing:

1) static constraint: $|x_i^{\hat{t}} - x_j^{\hat{t}}| > 1$ or $|y_i^{\hat{t}} - y_j^{\hat{t}}| > 1$;
2) dynamic constraint: $|x_i^{\hat{t}+1} - x_j^{\hat{t}}| > 1$ or $|y_i^{\hat{t}+1} - y_j^{\hat{t}}| > 1$ or $|x_i^{\hat{t}} - x_j^{\hat{t}+1}| > 1$ or $|y_i^{\hat{t}} - y_j^{\hat{t}+1}| > 1$.

The static fluidic constraints state that the minimum spacing between two droplets is one cell for any $\hat{t}$ 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 constraints, 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.

The contamination constraint needs to be enforced in order to prevent an erroneous bioassay outcome due to the possible contamination of droplets. 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} denote the contaminated spot caused by the two routes of d_i and d_j, and $\hat{t}^i_{c_{ij}}$ ($\hat{t}^j_{c_{ij}}$) represents 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 2-D plane can be formulated as follows.

1) *Input*: a netlist of droplets, a set of wash droplets, the locations of blockages, the locations of reservoirs, and the timing constraint.
2) *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.
3) *Constraint*: all fluidic, timing, and contamination constraints are satisfied.

III. Algorithm Overview

Our contamination aware droplet routing algorithm consists of three major stages: *preprocessing*, *intracontamination aware routing*, and *intercontamination aware routing*.

In preprocessing stage, the goal is to determine an initial 2-D routing path for each droplet. We first construct the preferred routing tracks to make droplets route orderly on these tracks thereby minimizing the used cells and reducing the possibility of congestion. By routing droplets along these tracks, the design complexity can be reduced, thereby decreasing the violation of the fluidic constraints. To reach better routability, we also propose a routing-resource-based equation to determine the routing priority for droplets.

In intracontamination aware routing stage, the major goal is to minimize the contaminated spots for routing paths of

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. Note that the bounding is defined as the rectangle formed by two diagonal points, and the two sides are parallel with the x and y-axis.
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 nonwashed 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 nonwashed contaminated spots in subproblem SP_p. Note that $CS_p = CS_{(r,p)}$, where $1 \le r \le p$.
$LA(v_i, v_j)$	Set of nonwashed look-ahead contaminated spots in the bounding box of nodes v_i and v_j.

droplets. To tackle this problem, a k-shortest path algorithm is proposed to reduce the contaminated spots within one subproblem. Then a dynamic-programming-based routing compaction technique is presented to minimize the execution time of the bioassay. In this case, the occurring clock cycle of each contaminated spot can be estimated.

In intercontamination aware routing stage, the goal is to achieve more washing efficiency for wash droplets. To avoid the timing overhead caused by scheduling wash operations 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, an 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 minimize the completion time of assay. The notations used in our algorithm are shown in Table I.

IV. PREPROCESSING STAGE

There are two main steps in the preprocessing stage, preferred routing tracks construction and routing priority calculation. In constructing the preferred routing tracks, we first analyze each droplet's moving vector, and set the routing direction for nonadjacent rows and columns on the microfluidic array. For the purpose of minimizing the used cells and routing complexity, a cost function is presented to guide the droplets to move along these tracks. To achieve better routability, we propose a routing-resource-based equation that considers the congestion issue between nets to determine the routing priority.

A. 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, in the droplet routing

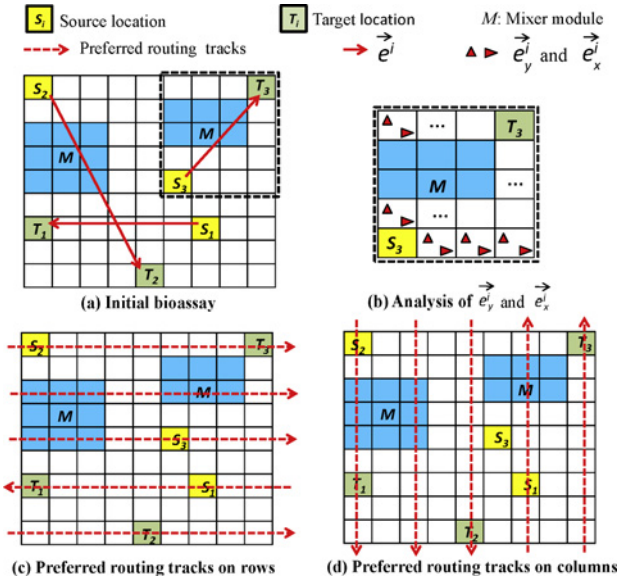

Fig. 4. Illustration of the preferred routing tracks construction. (a) Initial bioassay. (b) Analysis of the two linearly dependent vectors for net 3. (c) Preferred routing tracks on nonadjacent rows. (d) Preferred routing tracks on nonadjacent columns.

stage, there may exist some fluidic modules (e.g., mixer or diluter) decided by the previous placement stage. These modules are treated as blockages and may cause the routability problem in the routing stage. 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.

Fig. 4 illustrates the overall process of the preferred routing tracks construction. 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 2-D 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) [see Fig. 4(a)]. 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 [see Fig. 4(b)]. Then, we define the preferred routing tracks on rows and columns as follows.

1) Track on row r: $\overrightarrow{tk_r} = \sum_{\forall B_i \cap r \neq \phi} \overrightarrow{e_x^i}$.
2) 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 nonadjacent rows and nonadjacent columns [see Fig. 4(c) and (d)].

The intuition behind our preferred routing tracks construction is similar to traffic control, as each droplet can be regarded

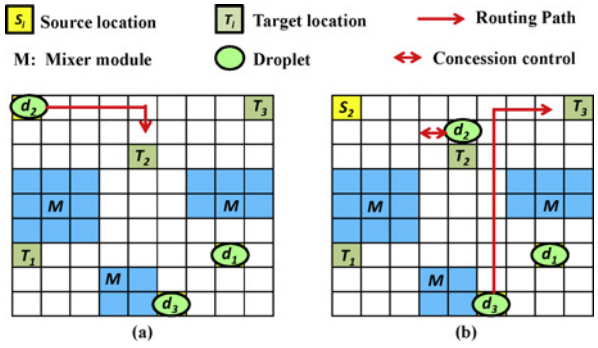

Fig. 5. Illustration of the concession control. (a) Route the droplet d_2 to the A-cell of t_2. (b) Route the droplet d_3 to the A-cell of t_3 while there is a concession of d_2.

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).

To guide the droplet move along the direction of preferred routing tracks, 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 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:

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

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 nonpreferred routing tracks, 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. If a routed droplet blocks the routing for another droplet, a concession for the routed droplet is adopted to make the routing successful. In detail, once a droplet is routed to its target cell, it may stall on this cell for a specific optical detection. To satisfy the fluidic constraints, the cell should be frozen and become a 3×3 blockage till the process 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 [see Fig. 5(a)]. 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 cells are the available cells away from the congested region and they can be found by using maze searching [see Fig. 5(b)].

B. 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 SRC_i^+ and unavailable routing resource SRC_i^- for droplet d_i as follows:

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

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

where $E_5(t_j)$ represents the 5×5 cell set center by the location of target t_j, and $E_3(d_j)$ represents the 3×3 cell set center by the location of droplet d_j. There are two main reasons behind the definition. When a droplet is routed to and stall on the target cell for some biological processes (e.g., testing or detection), the target cell will become a 3×3 blockage till the process is finished. In this case, unexpected violations of fluidic constraints can be avoided, i.e., only the outer cells of the 5×5 cell set center by the target can be used by other droplets. Furthermore, during the droplet routing, the minimum spacing should also be maintained to prevent the violation of fluidic constraints. In other words, for a droplet d_i, there must not be other droplets that locate on the 3×3 cell set center by d_i.

Then, we define the routing-resource-based equation as follows:

$$SRC_i^{\text{eq}} = \frac{SRC_i^+ - SRC_i^-}{SRC_i^+}. \quad (4)$$

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. When those droplets are routed to the target cells inside the bounding box B_i, the target cells will become blockages which may cause the routability problem for droplet d_i. As aforementioned, since droplets may stall on targets for some detection processes, we attempt to route droplet d_i first that has many target cells inside the bounding box B_i. Similarly, the cells of blockages and unrouted droplets inside the bounding box B_i will have detrimental effects on routability due to the module blockages and fluidic constraints.

For example, Fig. 6 demonstrates the overall calculation of routing priority for droplet d_2. The source and target location for droplet d_2 are (0, 8) and (4, 0), respectively. Thus, the $|B_2|$ is 45 (see the dashed line area). There is one target cell located at cell (0, 2) inside B_2, and the influence range is the rectangular area from (0, 0) to (2, 4). Therefore, the value of

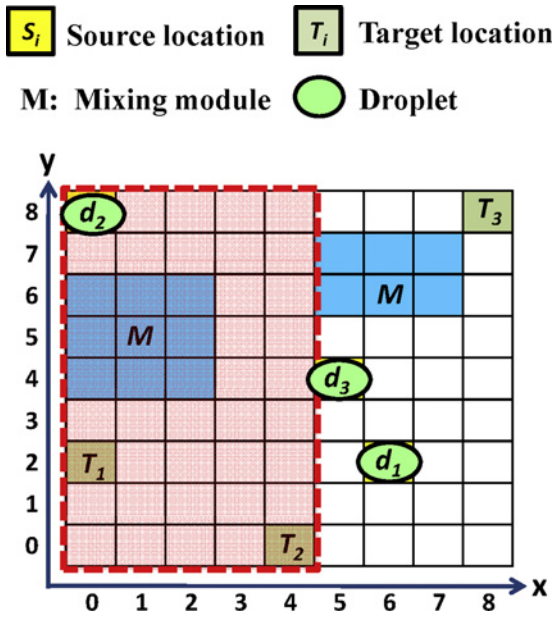

Fig. 6. Illustration of the routing priority calculation for droplet d_2.

SRC_2^+ can be calculated as 45+15 = 60. On the contrary, there are one mixing module located from (0, 4) to (2, 6) and one droplet located at (5, 4) whose influence range is from (4, 3) to (6, 5). The corresponding cardinalities of intersections with B_2 are 9 and 3, respectively. Then the SRC_2^- can be calculated as 9 + 3 = 12. Finally, the SRC_2^{eq} is $(60 - 12)/60$.

To determine the routing order, we calculate the SRC_i^{eq} for each unrouted droplet d_i. The higher value of SRC_i^{eq} means that there is less congested inside the routing region, which enables more possibility to route this droplet successfully. Therefore, the droplet d_i with maximum SRC_i^{eq} represents the highest routing priority. For the sake of efficiency, we use a priority queue Q_{eq} to find the routing order of unrouted droplets. Then, we iteratively pop an unrouted droplet d_i with the highest SRC_i^{eq} to route. If we cannot successfully route this selected droplet d_i due to the reason that a droplet d_j blocks the routing path of d_i, the proposed concession control method will be adopted to solve this problem. Finally, we dynamically update the SRC_k^{eq} of unrouted droplets d_k left in Q_{eq}.

V. Intracontamination Aware Routing Stage

In this section, we present an intracontamination aware routing stage to handle the intracontaminated spots. We first minimize the intracontaminated spots by a k-shortest path-based technique. To minimize the execution time of bioassays, a dynamic programming approach is proposed to transform a series of 2-D routing paths into the 3-D routing paths. According to the dynamic programming, the initial occurring clock cycles of intracontaminated spots can be estimated. Then, we formulate the entire problem into a minimum cost circulation flow network, and solve the flow problem to obtain an optimal wash scheduling. Finally, two proofs are followed to prove the optimality and correctness of our algorithm.

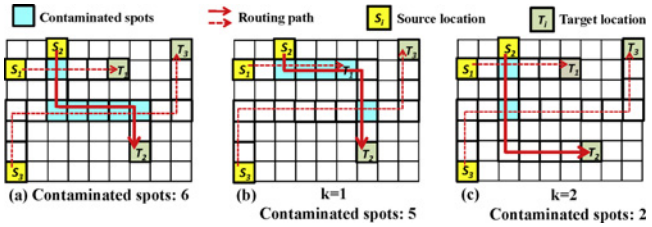

Fig. 7. Adopt the k-shortest path algorithm to reduce the cross-section. (a) Original routing solution of d_2. (b) First shortest path of d_2 in the remodeled routing graph. (c) Second shortest path of d_2 in the remodeled routing graph.

A. Routing Path Modification by k-Shortest Path

Although the above proposed scheme can schedule a high-throughput routing paths in the 2-D plane, it may cause a large amount of contaminated spots. As previously mentioned, the proposed algorithm routes the droplets orderly along the preferred routing tracks. In other words, multiple droplets may be routed along the same tracks to minimize the used cells. Therefore, contamination occurs and causes the erroneous outcome for bioassay. To overcome these drawbacks, we proposed a k-shortest-path-based algorithm [8] to reduce the number of contaminated spots. As the aforementioned algorithm, we iteratively select an unrouted droplet d_i with the highest SRC_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 $R_{(d_i,t_i)}$. Then, we update the number of contaminated spots for the droplet d_j $(j \neq i)$ in Q_{cs}, where $R_{(d_i,t_i)} \cap R_{(d_j,t_j)} \neq \phi$. The iteration terminates when 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 nodes 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, Fig. 7(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 Fig. 7(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 Fig. 7(b), which causes only slight 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.

There are many related works on the k-shortest path algorithm, such as A* search, Dijkstra-based modification, path deletion, and so on. We modify the algorithm discussed in [8], which is based on a shortest-path tree and heap-order-tree structure to find the k-minimum cost path.

B. 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, 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. As discussed before, contamination is likely to occur when multiple droplet routes cross or overlap 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 2-D routing and determine the occurring clock cycle of each contaminated spot. There are at least two advantages of 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 2-D routing paths to routing compaction, we encode each routing path into a corresponding 1-D moving string by using four direction characters u, d, l, and r, which 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 recursive formula as follows:

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

Our routing compaction step by dynamic programming approach can be summarized in Algorithm 1. First, we encode the first routing path P_1 into a moving string MS_1 (lines 2–3). Then we try to compact the other moving strings with MS_1 and use an array π to record the optimal compacted routing path. During the compaction, we should check the legality of the compacted string, i.e., the fluidic constraints should be satisfied for each droplet. Based on the proposed optimal substructure, if it is legal to compact MS_1 with 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 recorded table previously, plus one unit length. Otherwise, it will set to be infinite (lines 4–15). During the droplet routing, a droplet may block many routing paths of other droplets. To increase the routability, the concession control enables the droplet to move back and forth for concession. In this case, the concession control may cause some duplicate movements for this droplet. Thus, we delete the duplicate droplet movements and route droplets from A-cells to targets. Finally, we obtain the optimal compacted string, that represents the routing compaction results (lines 16–18).

Algorithm 1: Routing compaction by dynamic programming

```
input : 2-D routing paths p_i for N droplets
output: 3-D routing paths for N droplets
begin
    construct two 2-D arrays C and π;
    MS_1 ← encode P_1 into a moving string;
    for k ← 2 to N do
        MS_2 ← encode P_k into a moving string;
        l_i ← string length of MS_1;
        l_j ← string length of MS_2;
        for i ← 0 to l_i do
            for j ← 0 to l_j do
                C[i][j] ← optimal compacted length;
                π ← optimal compacted path;
            end
        end
        MS_1 ← optimal compacted string from π[i][j];
    end
    delete the duplicate movement in MS_1;
    route all droplets to targets ;
    return MS_1
end
```

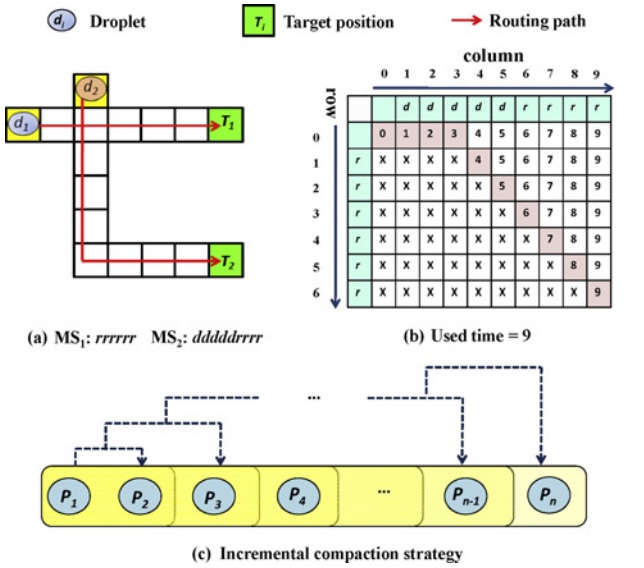

Fig. 8. Illustration of routing compaction by dynamic programming. (a) Two encoded moving strings MS_1 and MS_2 for the routing paths of droplet d_1 and d_2. (b) Optimal compaction that stored in the table. (c) Incremental strategy-based dynamic programming.

In detail, Fig. 8 shows the two major phases of our routing compaction. Given two 2-D routing paths as shown in Fig. 8(a), we first decode the 2-D routing paths into 1-D moving string. By using the four direction characters u, d, l, and r, the routing paths of droplet d_1 and d_2 can be encoded into $MS_1 = rrrrrr$ and $MS_2 = dddddrrrr$, respectively. Then, we use the dynamic programming approach to record the optimal compacted string in the table. For example, in Fig. 8(b), the entry with row 1 and column 3 in the table is set to be infinite because there exists an unexpected mixing within the

movements that d_1 moves one unit and d_2 moves three units. In the similar manner, if d_1 stalls in the original cell, d_2 can move three or four units without any violation of fluid constraints. The entries (0, 3) and (0, 4) are consequently assigned by 3 and 4. In the calculation of the entry (1, 4), the optimal value will be the minimum value among entries (0, 3), (0, 4), and (1, 3) in the table, plus one unit. By tracing the table, we can obtain the optimal routing paths with minimum compaction time [see Fig. 8(b)]. Fig. 8(c) describes the incremental compaction strategy. In each iteration, we treat the previous compacted strings as one string that stores the information of movements for multiple droplets. Then, we compact an uncompacted string of the routing path P_i with this string, and use dynamic programming to trace the optimal compaction. The iteration stops when the last compaction is finished.

C. Minimum Cost Circulation Flow Technique

Wash droplets can remove the liquid residue and nontarget molecules that have been adsorbed onto the surface to prevent the contamination problem. Although the wash droplet can clean many particles left behind the surface of the microfluidic array, 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 MCC-based algorithm for optimal wash-droplet routing to simultaneously minimize the number of the used cells and the cleaning time.

1) *Introduction to Minimum Cost Circulation Problem:* We first briefly introduce the MCC problem. 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:

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

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

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

where l_{ij} (u_{ij}) represents the lower (upper) bound on flow from node i to j, C_{ij} represents the cost per unit flow from node i to j, and x_{ij} represents the flow value from node i to j. The MCC problem can be solved strongly in 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 [10].

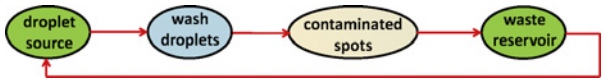

Fig. 9. Four phases of our minimum cost circulation construction.

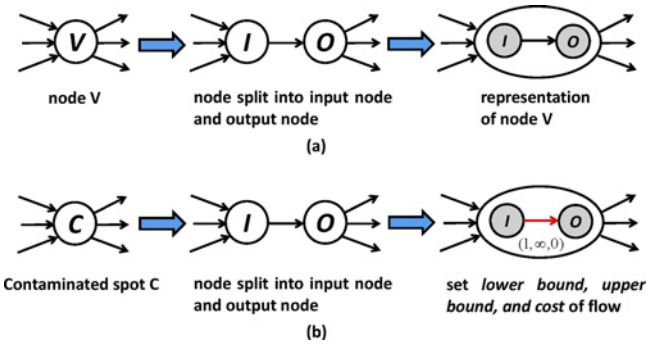

Fig. 10. Node capacity assignment. (a) Node split technique. (b) Adopt the node split technique to the contaminated spots.

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 to 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. To tackle the contamination constraint and achieve multiobjective optimizations, we propose a MCC-based algorithm to solve these problems. 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}) denotes the lower (upper) bound on the flow of this edge, and C_{ij} represents the associated cost per unit flow on this edge. As shown in Fig. 9, there are four major phases in our MCC formulation, which are a dummy source, a set of wash droplets, a 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.

1) *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^i_{in} and v^i_{out}, and an edge is connected from v^i_{in} to v^i_{out}. Fig. 10(a) shows the technique. 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 $(v_{\text{in}}^i, v_{\text{out}}^i)$ is set to be $(1, \infty, 0)$, as shown in Fig. 10(b). Note that u_{ij} is set to ∞ due to multiple wash droplets that can pass through the same contaminated spot.

2) *Assignment* 2 *(edge cost assignment):* another important issue is to determine the routing paths 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 IV-A 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.

1) *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. To avoid incorrect diagnosis outcomes, it is desirable to clean the cell before being used for routing other droplets. 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 denote the occurring time of contaminated spot v_i. Given a set of contaminated spots in CS_p, $(v_1, \hat{t}_1)$, $(v_2, \hat{t}_2)$, and $\ldots$ $(v_n, \hat{t}_n)$, 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 $\hat{t}_i \leq \hat{t}_j$. The associated 3-tuple of the edge is assigned 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. The associated 3-tuple of the edge is assigned by $(0, \infty, Cost(R_{(v_i,v_j)}))$. There are two major reasons for the assignment of the 3-tuple. First, to achieve high efficiency of the wash operation, multiple wash droplets are allowed to perform the wash operations. When a wash droplet cleans the contaminated spot, it can be directly routed to the waste reservoir. Another contaminated spots left behind can be cleaned by other wash droplets. Therefore, the lower bound of the 3-tuple is set to be zero. Second, since the cells used by wash droplets should be also minimized, the wash droplets are routed along the preferred routing tracks. In other words, the routing paths of the wash droplets may share some cells. To ensure the correctness, the upper bound of the 3-tuple is set to be infinite. Briefly, contamination problem occurs when two droplets pass through the same spot in different clock cycle. The timing-based topology demonstrates that the wash operation should be performed between the successive 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.

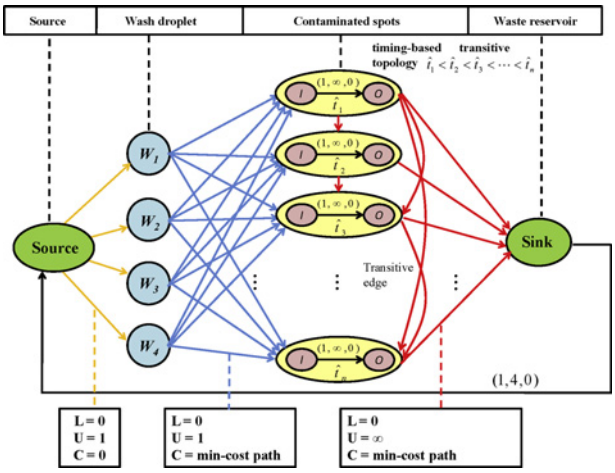

Fig. 11. Illustration of our MCC construction for the four phases: source node of dispensing port, node set of wash droplets, contaminated spots, and sink node of waste reservoir, respectively.

2) *Rule 2 (connection strategy between phases):* there are four major phases in our MCC formulation as shown in Fig. 9. 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 connect the sink back to the dummy source, with the associated 3-tuple $(1, 4, 0)$. Note that the maximum number of available wash droplets is four in this paper.

Fig. 11 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.

Proof: A feasible solution of the circulation problem is a set of flows on the flow graph G_f that satisfies all the constraints (conservation constraint and bounded constraint). Generally, we only discuss the nontrivial contamination problem ($CS_p \neq \phi$). In the flow formulation, we enhance at least one flow from the sink back to the dummy source, meaning there is at least one flow from the source to the wash-droplet set. To characterize the proof, we assume this flow that leaves from dummy source is x_f. As shown in Fig. 10, we adopt the node split technique to split the nodes $v_i \in CS_p$ into two intermediate nodes, v_{in}^i and v_{out}^i, with a flow lower bound

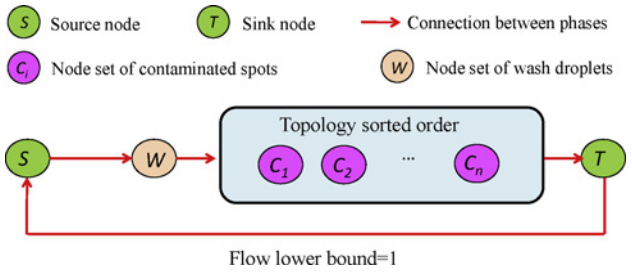

Fig. 12. Illustration of the proof for Theorem 1.

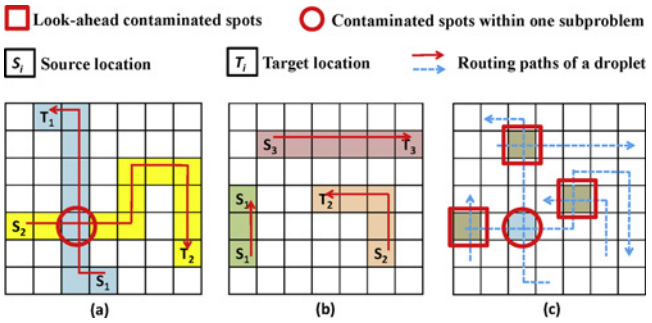

Fig. 13. Illustration of look-ahead routing scheme. (a) Routing solution of subproblem SP_i. (b) Routing solution of subproblem SP_{i+1}. (c) Both intracontamination within SP_i and intercontaminations between SP_i and SP_{i+1}.

by one. Based on the flow formulation rule 1, there exists an edge that connects v_{out}^i and v_{in}^j, iff $t_i \leq t_j$. To achieve the bounded constraint in $(v_{\text{in}}^i, v_{\text{out}}^i)$, we construct the order $(v_1, v_2, \ldots, v_n)$ by topological sort. Then we can make x_f to pass through the path P_{x_f} by the order $(v_{\text{in}}^1, v_{\text{out}}^1, \ldots, v_{\text{in}}^k, v_{\text{out}}^k)$, where $2 \leq k \leq n$, thereby passing through all the nodes in CS_p. In other words, the bounded constraint of these edges $(v_{\text{in}}^i, v_{\text{out}}^i)$ can be achieved. Since there is at least one edge that connects w_j to v_{in}^1, and there is an edge that connects v_{out}^n to the sink (waste reservoir), thus, there exists x_f from dummy source $\rightarrow w_j \rightarrow P_{x_f} \rightarrow$ sink, and then back to the dummy source, thereby forming a feasible circular flow.

Fig. 12 demonstrates the entire proof in brief. Based on the proposed construction, the flow network enhances at least one flow from the sink back to the source, meaning at least one flow from the source to the node set of wash droplets. In other words, at least one wash droplet will perform the wash operation. By tracing all the node set of contaminated spots in a topological sorted order, the flow lower bound of the node set can be achieved. Therefore, there exists a feasible solution under the two basic assignments and two flow formulation rules. ■

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

Proof: The most important issue is that we should schedule the correct wash operations on contaminated spots. In other words, to prevent the contamination problem, the wash droplets must be routed to clean the residue left behind the surface. Theorem 1 has showed the feasibility of our MCC formulation. Under the proposed flow formulation rules, the flow graph $G'_f = (V_f, E_f/(v_{sink}, v_{source}))$ forms a direct acyclic graph. This feature ensures that the feasible flow must pass through the source and target nodes, which ensures that the contaminated spots can only be correctly cleaned by wash droplets. Furthermore, we make multiple wash droplets route on the preferred routing tracks to minimize the used cells. Based on our construction, the MCC algorithm will obtain a feasible flow with minimum cost that represents the optimal scheduling of wash operations. ■

VI. Intercontamination Aware Routing Stage

In this section, we present the intercontamination aware routing to further improve the wash efficiency. We propose a look-ahead routing scheme that simultaneously considers the contaminated spots within one subproblem and successive subproblem. To clean these contaminated spots, the wash operation is formulated into a typical traveling salesman problem (TSP). Finally, a theorem proves that the proposed algorithm can correctly schedule the droplet routing and wash operations simultaneously without any redundant insertion of wash operations between successive subproblems.

A. Look-Ahead Routing Scheme

By using the MCC-based algorithm, the wash droplets can be optimally scheduled to clean the intracontaminations within one subproblem. However, some contaminations left behind this subproblem will be treated as blockages for the next subproblem (i.e., intercontaminations). To avoid the contamination problem, extra wash operations should be performed between the successive subproblems, which may cause timing overhead. To remedy these deficiencies, we propose a look-ahead routing scheme that predicts the intercontaminations for the successive subproblems. By using the MCC-based algorithm, both the intra and intercontaminations can be simultaneously cleaned within one subproblem. Consequently, the execution time is significantly reduced. Fig. 13(a) shows the routing solution of subproblem SP_i with an intracontamination. Before cleaning this intracontamination by using wash droplets, we also derive the routing solution of subproblem SP_{i+1} as shown in Fig. 13(b). Based on the two routing solutions, the intercontaminations between subproblem SP_i and SP_{i+1} can be derived and are handled with intracontaminations in subproblem SP_i, as shown in Fig. 13(c).

1) *TSP Formulation:* Consider these intercontamination nodes $v_k \in LA(v_i, v_j)$, where (v_i, v_j) is an edge in the MCC flow graph. We perform the wash operation to clean these nodes. In other words, 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

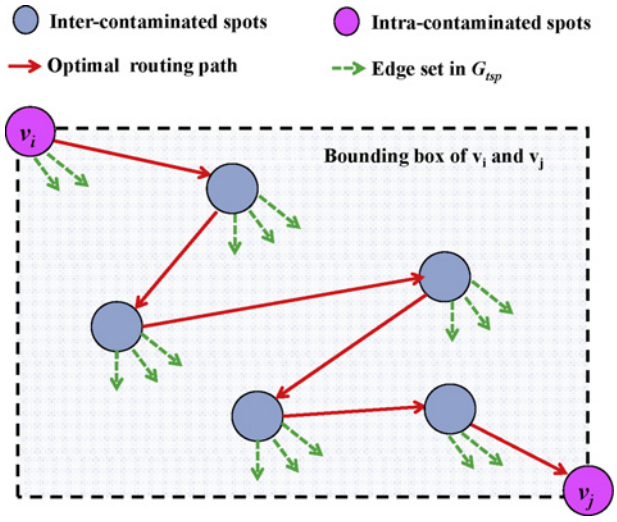

Fig. 14. TSP-optimization-based algorithm for scheduling the routing path of the wash droplet.

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 TSP. Assume the optimal routing path is $R^{tsp}_{(v_i,v_j)}$ and the corresponding minimum cost is $Cost(R^{tsp}_{(v_{k_1},v_{k_2})})$. We reformulate the cost of the edge (v_i, v_j) in G_f by $Cost(R^{tsp}_{(v_{k_1},v_{k_2})})$, and the corresponding routing path of (v_i, v_j) is $R^{tsp}_{(v_{k_1},v_{k_2})}$. The dynamic-programming-based approach for solving the TSP problem is used in this paper [2]. However, the dynamic programming runs exponential time, $O(|c|^2 2^{|c|})$, where c represents the set of intercontaminated spots. To avoid run time overhead, we restrict the set of intercontaminated spots with a maximum cardinality by 10.

Fig. 14 illustrates the proposed method in this routing stage. For the edges (v_i, v_j) in the flow graph, the intercontaminated spots v_k inside the bounding box of v_i and v_j are considered. Then we adopt the aforementioned construction method to construct a TSP graph. Note that the underlying graph of the TSP graph is a complete graph. By implementing the TSP optimization, an optimal path that starts from v_i to v_j while passing through all the node set of contaminated spots can be scheduled.

2) *Final Compaction by Dynamic Programming:* 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 contaminated spots are cleaned before being used for routing other droplets. Finally, the overall algorithm leads to the following theorem.

Theorem 3: The proposed algorithm can correctly schedule the droplet routing and wash operations simultaneously without any redundant insertion of wash operations between successive subproblems.

Proof: For any given subproblem SP_p, we handle the two types of contaminated spots, intra and intercontaminated spots. In the look-ahead routing scheme, given an edge (v_i, v_j) in the flow graph, we only consider the intercontaminated spots inside the bounding box of v_i and v_j. Therefore, the core of the proof is to show that the intercontaminated spots outside the bounding box should be also cleaned by satisfying the contamination constraint. As the discussed MCC formulation rules in Section V, we handle contaminated spots in CS_p within one subproblem, where $CS_p = CS(r, p)$, $1 \le r \le p$. In other words, the intercontaminated spots that are not formulated into the G_{tsp} will be considered in the afterward occurring subproblems. Since Theorems 1 and 2 have proved the feasibility and correctness of our MCC formulation, the proposed algorithm can simultaneously clean the intra and intercontaminated spots without any redundant insertion of wash operations between successive subproblems. ■

TABLE II
STATISTICS OF THE ROUTING BENCHMARKS

Bioassay	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

Size: Size of the microfluidic array
#Net: Total input nets
#Sub: Number of subproblems
#W: Number of wash droplets
#D_{max}: Maximum number of droplets with one subproblem

VII. Run Time Complexity Analysis

In this section, we discuss the run time complexity of the proposed algorithm. The run time bottleneck of our algorithm is to solve the MCC formulation. Let W_c/H_c denote the width/height of a biochip. In the MCC flow graph $G_f = (V_f, E_f)$, the node set mainly depends on the number of contaminated spots between different droplet routes. Thus, the size of V_f is $O(|D|(W_cH_c))$, where D represents the droplet set. As constructed by flow formulation rules, the size of E_f is $O(|V_f|^2)$, i.e., $O(|D|^2(W_cH_c)^2)$. Since we solve the MCC problem by implementing [10], the time complexity of solving the MCC problem is consequently $O(|D|^5(W_cH_c)^5(log(|D|W_cH_c))^2)$. Finally, let N_s represent the total subproblems for a given bioassay, the overall time complexity is $O(N_s|D|^5(W_cH_c)^5(log(|D|W_cH_c))^2)$.

VIII. Experimental Results

We have implemented our contamination aware droplet router in the C++ language on a 2 GHz 64-bit Linux machine with 16 GB memory. In this paper, the parameters α, β, and γ are set to be 1, 4, and 2, respectively. We perform experiments to verify the efficiency and effectiveness of our algorithm on four widely used bioassays from [21] and [25]. These bioassays represents two biological processes, in-vitro diagnostics and protein reactions, respectively. The locations of modules (e.g., mixer, diluter) and other components (e.g., I/O port, optical detection device) are decided by the previous placement stage [17]. Droplets routing should be well-scheduled to achieve high-throughput outcomes. Table II shows the

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.)

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

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

$\#C_{intra}$: Number of intra-contaminations
#UC: Number of used cells for routing
$\#C_{inter}$: Number of inter-contaminations
T_{exe}: Execution time for the bioassays
CPU: CPU time (s)

TABLE V
COMPARISONS OF OUR ALGORITHM WITH DIFFERENT NUMBER OF WASH DROPLETS

Bioassay	Ours (#W = 1)			Ours (#W = 2)			Ours (#W = 3)			Ours (#W = 4)		
	#UC	T_{exe}	CPU	#UC	T_{exe}	CPU	#UC	T_{exe}	CPU	#UC	T_{exe}	CPU
in-vitro_1	473	312	0.20	409	230	0.41	370	206	0.50	351	193	0.58
in-vitro_2	382	297	0.21	331	237	0.37	301	211	0.31	281	191	0.39
protein_1	2723	1976	2.11	2415	1690	2.31	2283	1477	2.37	2213	1394	2.58
protein_2	1914	1624	1.31	1612	1428	1.14	1442	1178	1.41	1362	1108	1.49
Total	5492	4209	3.83	4767	3585	4.23	4396	3072	4.59	4207	2886	5.04

T_{exe}: Execution time for the bioassays
#UC: Number of used cells for routing
CPU: CPU time (s)

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, and "#W" denotes the number of wash droplets.

In the first experiment, we evaluated the reduction of intracontaminations 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 intracontaminations by 48% but also reduced the respective used cells and execution time by 8% and 13% with small increase in the central processing unit (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 can predict the intercontaminations for the next subproblem. Then the MCC-based algorithm is adopted to simultaneously clean both intra and intercontaminations. The total execution time is accordingly reduced 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.

TABLE VI
COMPARISONS BETWEEN [27] AND OUR ALGORITHM

Bioassay	[27]				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

$\#C_{intra}$: Number of intra contaminations
#UC: Number of used cells for routing
T_{exe}: Execution time for the bioassays
CPU: CPU time (s)

In the third experiment, we demonstrate the results of our algorithm by using different number of wash droplets. Table V demonstrates the used cells, execution time, and CPU time by using one, two, three, and four wash droplets, respectively. Since the capacity of the wash reservoir is limited, we set the maximum available wash droplets up to four. The experimental results depict that using four wash droplets achieve the best results in terms of used cells and completion time with slightly increased CPU time.

In the fourth experiment, we compared the effectiveness of our contamination aware droplet routing algorithm with the state-of-the-art algorithm [27] in Table VI. For comparison purpose, we implement the state-of-the-art algorithm [27]. 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 intracontaminations 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.

IX. CONCLUSION

In this paper, we proposed a contamination aware droplet routing algorithm for DMFBs. To reduce the routing complexity and used cells, we first constructed 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 applied a k-shortest path routing technique to minimize the contaminated spots. Then, to take advantage of multiple wash droplets, we adopted a MCC algorithm 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 simultaneously cleaned both contaminations within one subproblem and 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.

ACKNOWLEDGMENT

The authors would like to thank Y. Zhao and Prof. K. Chakrabarty of the Duke University, Durham, NC, for providing the authors with benchmarks and manuscripts for the comparative studies.

REFERENCES

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

[2] R. Bellman, "Dynamic programming treatment of the travelling salesman problem," *J. ACM*, vol. 9, pp. 61–63, Jan. 1962.

[3] K. F. Böhringer, "Modeling and controlling parallel tasks in droplet based microfluidic systems," *IEEE Trans. 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. CAD*, vol. 27, no. 10, pp. 1714–1724, Oct. 2008.

[5] S. K. Cho, S.-K. Fan, H. Moon, and C.-J. Kim, "Toward digital microfluidic circuits: Creating, transporting, cutting and merging liquid droplets by electrowetting-based actuation," in *Proc. MEMS Conf.*, Jan. 2002, pp. 32–35.

[6] K. Chakrabarty, "Toward fault-tolerant digital microfluidic lab-on-chip: Defects, fault modeling, testing, and reconfiguration," in *Proc. IEEE Int. Conf. BioCAS*, Dec. 2008, pp. 329–332.

[7] J. Ding, K. Chakrabarty, and R. B. Fair, "Scheduling of microfluidic operations for reconfigurable 2-D electrowetting arrays," *IEEE Trans. CAD*, vol. 20, no. 12, pp. 1463–1468, Dec. 2001.

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

[9] R. B. Fair, A. Khlystov, T. D. Tailor, V. Ivanov, R. D. Evans, P. B. Griffin, S. Vijay, V. K. Pamula, M. G. Pollack, and J. Zhou, "Chemical and biological applications of digital-microfluidic devices," *IEEE Des. Test Comput.*, vol. 24, no. 1, pp. 10–24, Jan. 2007.

[10] A. V. Goldberg and R. E. Tarjan, "Finding minimum cost circulations by canceling negative cycles," *J. ACM*, vol. 36, no. 4, pp. 873–886, Oct. 1989.

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

[12] C. C.-Y. Lin and Y.-W. Chang, "Cross-contamination aware design methodology for pin-constrained digital microfluidic biochips," in *Proc. IEEE/ACM DAC*, Jun. 2010, pp. 641–646.

[13] H. Moon, A. R. Wheeler, R. L. Garrell, J. A. Loo, and C. J. Kim, "An integrated digital microfluidic chip for multiplexed proteomic sample preparation and analysis by MALDI-MS," *Lab Chip*, vol. 6, pp. 1213–1219, Feb. 2006.

[14] E. Maftei, P. Paul, and J. Madsen, "Tabu search-based synthesis of dynamically reconfigurable digital microfluidic biochips," in *Proc. CASES*, Oct. 2009, pp. 195–203.

[15] M. G. Pollack, A. D. Shenderov, and R. B. Fair, "Electrowetting-based actuation of droplets for integrated microfluidics," *Lab Chip*, vol. 2, pp. 96–101, May 2002.

[16] F. Su and K. Chakrabarty, "Architectural-level synthesis of digital microfluidics-based biochips," in *Proc. IEEE/ACM ICCAD*, Nov. 2004, pp. 223–228.

[17] F. Su and K. Chakrabarty, "Unified high-level synthesis and module placement for defect-tolerant microfluidic biochips," in *Proc. IEEE/ACM DAC*, Jun. 2005, pp. 825–830.

[18] F. Su and K. Chakrabarty, "Design of fault-tolerant and dynamically reconfigurable microfluidic biochips," in *Proc. IEEE/ACM DATE*, Mar. 2005, pp. 223–228.

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

[20] F. Su and K. Chakrabarty, "Module placement for fault-tolerant microfluidics-based biochips," *ACM TODAES*, vol. 11, no. 3, pp. 682–710, Jul. 2006.

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

[22] J. Y. Toon and R. L. Garrell, "Preventing biomolecular adsorption in electrowetting-based biofluidic chips," *Anal. Chem.*, vol. 75, no. 19, pp. 5097–5102, Oct. 2003.

[23] T. Xu and K. Chakrabarty, "Integrated droplet routing and defect tolerance in the synthesis of digital microfluidic biochips," *ACM JETC*, vol. 4, no. 3, pp. 1–24, Aug. 2008.

[24] P.-H. Yu, C.-L. Yang, and Y.-W. Chang, "Placement of defect-tolerant digital microfluidic biochips using the T-tree formulation," *ACM JETC*, vol. 3, no. 3, pp. 1–31, Nov. 2007.

[25] 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. CAD*, vol. 27, no. 11, pp. 1928–1941, Nov. 2008.

[26] P.-H. Yuh, S. Sapatnekar, C.-L. Yang, and Y.-W. Chang, "A progressive-ILP based routing algorithm for cross-referencing biochips," in *Proc. IEEE/ACM DAC*, Jun. 2008, pp. 284–289.

[27] Y. Zhao and K. Chakrabarty, "Cross-contamination avoidance for droplet routing in digital microfluidic biochips," in *Proc. IEEE/ACM DATE*, Apr. 2009, pp. 1290–1295.

[28] Y. Zhao and K. Chakrabarty, "Synchronization of washing operations with droplet routing for cross-contamination avoidance in digital microfluidic biochips," in *Proc. IEEE/ACM DAC*, Jun. 2010, pp. 635–640.

Tsung-Wei Huang is an undergraduate student with the Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan.

His current research interests include physical design automation for biochips.

Mr. Huang received the first place in the ACM SIGDA Student Research Competition in 2010.

Chun-Hsien Lin is an undergraduate student with the Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan.

His current research interests include physical design automation for biochips.

Tsung-Yi Ho (M'08) received the M.E. degree in computer science from the National Chiao-Tung University, Hsinchu, Taiwan, in 2001, and the Ph.D. degree in electrical engineering from the National Taiwan University, Taipei, Taiwan, in 2005.

He was a Visiting Scholar with the University of California, Santa Barbara, from 2003 to 2004, with Waseda University, Kitakyushu, Japan, in 2005, and with Synopsys, Mountain View, CA, in 2008. Since 2007, he has been with the Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan, where he is currently an Assistant Professor. His current research interests include physical design automation for nanometer integrated circuits and biochips.