A Progressive-ILP-Based Routing Algorithm for the Synthesis of Cross-Referencing Biochips

Ping-Hung Yuh, Sachin S. Sapatnekar, *Fellow, IEEE*, Chia-Lin Yang, *Member, IEEE*, and Yao-Wen Chang, *Member, IEEE*

***Abstract*—Due to recent advances in microfluidics technology, digital microfluidic biochips and their associated computer-aided-design problems have gained much attention, most of which has been devoted to direct-addressing biochips. In this paper, we solve the droplet routing problem under the more scalable cross-referencing biochip paradigm. We propose the first droplet routing algorithm that directly solves the problem of routing. We first present an optimal basic integer-linear-programming (ILP) formulation. Due to its complexity, we also propose a progressive-ILP scheme to determine the locations of droplets at each time step. Simulation results demonstrate the efficiency and effectiveness of our algorithm.**

***Index Terms*—Cross-referencing biochips, integer linear programming (ILP), progressive, routing.**

I. Introduction

RECENTLY, there have been many significant advances in microfluidic technologies [2]. Microfluidic biochips show numerous advantages over conventional assay methods, including portability, sample/reagent volume reduction, and faster analysis, and offer a platform for developing clinical and diagnostic applications, such as infant health care and point-of-care disease diagnostics. Other promising applications include environmental toxin monitoring and automated and parallel drug discovery, and this breadth of applicability implies that microfluidic biochips are increasingly used in laboratory procedures in molecular biology.

First-generation biochips are based on manipulating continuous liquid flows using external pressure sources (e.g., micropumps) and microchannels. Although they have been successfully applied to many biological applications, their lack of reconfigurability makes them unsuitable for large-scale and scalable systems. Second-generation biochips are based on digitalizing continuous liquid flow into discrete liquid particles, called droplets [3]. Each droplet can be independently controlled by the electrohydrodynamic forces generated by electrodes. In this way, droplet movement can be controlled by a system clock. Due to this parallel with digital electronic systems, second-generation biochips are referred to as "digital microfluidic biochips."

Typically, a digital microfluidic biochip has a 2-D microfluidic array, which consists of a set of electrodes. Droplets are routed through this array, and various fundamental operations, such as droplet merging, droplet splitting, and droplet transportation, are performed on them during this process. In the simplest and most common droplet control scheme, each electrode is directly addressed and controlled by a dedicated control pin, which allows each electrode to be individually activated. In this paper, we refer to these types of digital microfluidic biochips as direct-addressing biochips. While this architecture provides a high flexibility for droplet movement, it suffers from the major drawback that the number of control pins rapidly increases as the system complexity (i.e., the size of the array) increases. The large number of control pins affects product cost, and the control-wire routing problems complicate the design process, and therefore, this architecture is only applicable to small-scale biochips (fewer than 10×10 electrodes) [4].

To overcome these limitations, recently, a new digital microfluidic-biochip architecture has been proposed [5]. This architecture uses a row/column-addressing scheme, where all electrodes in one row (or column) are connected to a single control pin. Therefore, the number of control pins is greatly reduced: This number is now proportional to the perimeter of the chip rather than the area of the chip. We refer to this type of biochip architecture as the cross-referencing biochip architecture. However, this also introduces a new set of limitations. Since an electrode can potentially control the movement of all droplets in a row/column at the same time, this incurs higher droplet-movement complexity than that for direct-addressing biochips. Moreover, the manipulation of more than two droplets causes electrode interference among droplets, which prevents multiple droplets from moving at the same time. This limitation is a critical challenge in designing a low-cost cross-referencing biochip for high-performance applications, such as large-scale protein analysis.

In this paper, we tackle the problem of droplet routing in cross-referencing biochips. The main challenge of this routing problem is to ensure the correctness of droplet movement; the fluidic property which avoids unexpected mixing among droplets must be satisfied, and electrode-interference patterns

Manuscript received October 19, 2008; revised February 1, 2009. Current version published August 19, 2009. This work was supported in part by the National Science Council of Taiwan under Grants NSC 97-2221-E-002-242-MY3 and NSC 95-2221-E-002-098-MY3 and in part by the Excellent Research Projects of National Taiwan University under Project 97R0062-05. This paper was presented in part at the IEEE/Association for Computing Machinery (ACM) Design Automation Conference (DAC) 2008 [1]. This paper was recommended by Associate Editor K. Chakrabarty.

P.-H. Yuh and C.-L. Yang are with the Department of Computer Science and Information Engineering, National Taiwan University, Taipei 106, Taiwan (e-mail: r91089@csie.ntu.edu.tw; yangc@csie.ntu.edu.tw).

S. S. Sapatnekar is with the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN 55455 USA (e-mail: sachin@ece.umn.edu).

Y.-W. Chang is with the Department of Electrical Engineering and the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan (e-mail: ywchang@cc.ee.ntu.edu.tw).

Digital Object Identifier 10.1109/TCAD.2009.2023196

that prevent multiple droplets from moving at the same time must be avoided. The goal of droplet routing is to minimize the maximum droplet-transportation time and has several motivations. First, this minimization is critical to real-time applications, such as monitoring environmental toxins. Second, the minimized droplet-transportation time leads to shorter time that a sample spent on a biochip, which is desirable to maintain the bioassay execution integrity.

A. Related Work

Droplet routing is a critical step in biochip design automation. Previous routing approaches mainly focus on direct-addressing biochips [6]–[10]. Recently, the problem of manipulating droplets on a cross-referencing biochip has attracted some attention, but to our knowledge, all existing methods begin with a direct-addressing routing result and transform it to a cross-referencing routing result, cycle-by-cycle. Griffith *et al.* [8] proposed a graph-coloring-based method that is applied to each successive cycle of the direct-addressing solution. Each node in an undirected graph represents a droplet, and two nodes are connected by an edge if these two droplets cannot move at the same time. Therefore, the minimum time to move all droplets is equivalent to the minimum number of colors required to color this undirected graph. Xu *et al.* [11] proposed a clique-partitioning-based method that is also sequentially applied to the direct-addressing solution. Each clique is a set of droplets whose destination cells, i.e., the cell to which the droplets are supposed to move, are in the same row or column. The number of cliques is the maximum time to move all droplets. Since clique partitioning is NP-hard, they proposed an efficient heuristic approach to solve this problem. However, droplets with different destination cells can move at the same time. Therefore, their algorithm may potentially increase droplet-transportation time. Any algorithm that uses the direct-addressing solution as a starting point is limited by that solution, and to the best of our knowledge, there is no existing routing algorithm that is directly targeted to cross-referencing biochips.

B. Our Contribution

In this paper, we propose an integer-linear-programming (ILP)-based droplet routing algorithm for cross-referencing biochips. We derive the basic ILP formulation to *simultaneously* perform droplet routing and assign voltages to the cross-referencing electrodes, while minimizing the maximum droplet-transportation time. Moreover, we also model multipin nets in our ILP formulation for practical bioassays, where multiple droplets are merged during their transportation. To overcome the computational cost of the ILP, we also propose a progressive-ILP routing scheme, which is used to find the minimum-cost droplet locations at each time step using an ILP. Unlike the progressive-ILP scheme proposed in [12], which divides the original problem spatially, our algorithm divides the original routing problem temporally. In this way, the original problem is reduced to a manageable size, and we can practically apply an ILP-based method to find a good solution within reasonable CPU time. The major contributions of this paper include the following.

1) We propose the first routing algorithm that *directly* solves the routing problem in cross-referencing biochips. In contrast with previous works that start with an initial direct-addressing routing solution, our algorithm has higher flexibility and can obtain better solutions for droplet routing on cross-referencing biochips.
2) To tackle the complexity of the basic ILP formulation, we propose the progressive-ILP routing scheme. This scheme iteratively determines the locations of droplets at each time step by ILP formulation and, therefore, can obtain a high-quality solution within reasonable CPU times.
3) Unlike previous works that only move a subset of droplets at each time (for example, the algorithm proposed in [11] only moves droplets whose destination cells are in the same row/column), our algorithm maximizes the number of droplets that can simultaneously move at the same time, even if the destination cells are not in the same row/column. Therefore, our algorithm can obtain a routing solution with lower droplet-transportation time. This minimization is particularly important for cross-referencing biochips due to the electrode-interference problem.

Simulation results demonstrate the efficiency of our progressive routing scheme as compared with the basic ILP formulation. The basic ILP formulation requires more than five days while the progressive-ILP routing scheme requires no more than about 15 s for one bioassay. Simulation results also demonstrate the effectiveness of our algorithm as compared with previous work. For example, for the protein assay, our algorithm obtains 59.61% smaller maximum droplet-transportation time than the network-based method proposed in [10] plus the clique-partitioning-based algorithm proposed in [11].

The remainder of this paper is organized as follows. Section II details routing on cross-referencing biochips and formulates the droplet routing problem. Next, Section III presents the basic ILP formulation for droplet routing problem and the complexity analysis. Section IV details the progressive-ILP routing scheme, while Section V shows the simulation results. Finally, concluding remarks are provided in Section VI.

II. Routing on Cross-Referencing Biochips

In this section, we first show the architecture of cross-referencing biochips. Next, we detail the unique electrode interference that could result in incorrect droplet movement and the fluidic constraints that avoid this and guarantee correct droplet movement. Finally, we present the problem formulation of the droplet routing problem.

A. Cross-Referencing Biochips

Fig. 1 shows the top view of a cross-referencing biochip. A droplet is sandwiched between two plates. A set of electrodes spans a full row in the X-dimension and a full column in the Y-dimension and is assigned either a driving or reference

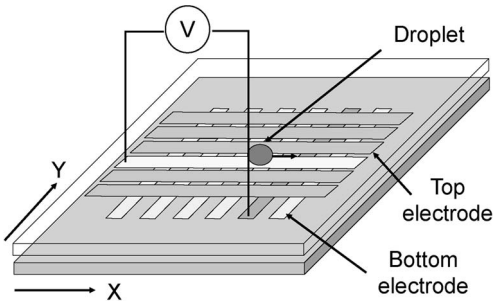


Fig. 1. Top view of a cross-referencing biochip.

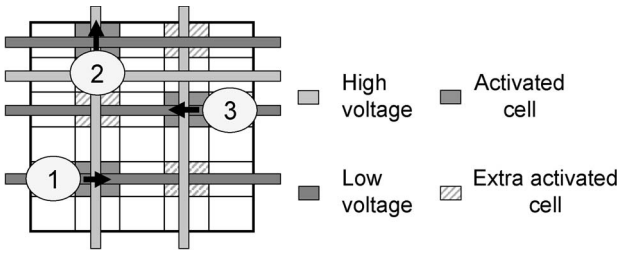


Fig. 2. Illustration of electrode interference. Unselected rows/columns are left floating.

voltage. Two sets of electrodes are orthogonally placed, one set each on the top and bottom plates, as shown in Fig. 1, and each electrode can be set to either a high or a low voltage level. A grid point is "addressed" if there is a potential difference between the upper and lower electrodes, i.e., one is high while the other is low. This causes a droplet at a neighboring grid point to move into this location.

The advantage of cross-referencing biochips is that we require only $\hat{W} + \hat{H}$ control wires for droplet movement, instead of the $\hat{W}\hat{H}$ control wires required for direct-addressed biochips, where $\hat{W}$ $(\hat{H})$ is the width (height) of a biochip, measured in terms of the number of electrodes in each dimension. Therefore, these biochips incur reduced package and fabrication costs.

B. Electrode Constraint

The improvement in packaging and fabrication costs, described earlier, comes at the cost of reduced flexibility for droplet movement, as compared to direct-addressing biochips. When moving multiple droplets simultaneously, we must set potential levels on multiple rows/columns as driving electrodes. However, since each set of electrodes in the $X(Y)$-direction spans the entire row (column), setting electrode-voltage values to move a set of droplets could imply that some droplets may be inadvertently and incorrectly moved. We show this idea in Fig. 2, where our goal is to move three droplets at the same time, to neighboring locations. The picture shows the set of voltage assignments to the electrodes to achieve this goal: These destinations of the droplets are selected by activating the corresponding rows and columns and by setting one of the lines to high and the other to low. However, due to the grid structure, several other locations are also accidentally activated. If either of these is adjacent to a droplet, it will cause an unwanted effect. For example, in this scheme, droplet 2 is now attracted to the

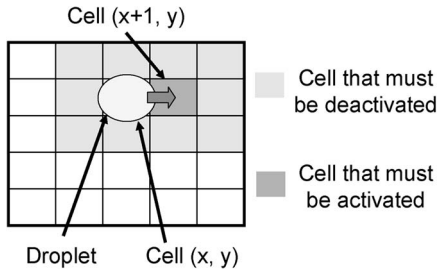


Fig. 3. Illustration of the modeling of the electrode constraint.

locations just above and just below it and is likely to split into two. This scenario is referred to as *electrode interference*, and it must be avoided during droplet transportation. The restriction that avoids electrode interference is referred to as an *electrode constraint*. Note that not every extra activated cell causes incorrect droplet movement: For example, the top-right activated cell has no effect on droplet movement, since it has no neighboring droplets. Only the extra activated cells that are around a droplet and its destination cell may cause incorrect droplet movement.

The electrode constraint imposes serious restrictions on the movement of multiple droplets. When more than two droplets are moved, we may have to stall some droplets to satisfy the electrode constraint, which will result in longer droplet-transportation times. For example, in Fig. 2, we can stall droplet 3 and move the other two, and this will avoid electrode interference. The manner in which electrode constraints are handled is a critical routing consideration in cross-referencing biochips and is important in minimizing the droplet-transportation time.

We now show how the electrode constraint can be modeled. Suppose that a droplet d moves from cell (x, y) to cell $(x + 1, y)$ at time $t + 1$, as shown in Fig. 3. Then, the cell (x, y), as well as the ten cells that surround $(x, y) \cup (x + 1, y)$, as shown in Fig. 3, must be deactivated. Otherwise, d is attracted by two electrohydrodynamic forces, so d may move incorrectly or be split into two smaller droplets. Note that in the case that d stays at its original location, cell (x, y) does not necessarily have to be activated, and all the neighboring cells of (x, y) must be deactivated, as demonstrated in [13].

C. Fluidic Constraint

Besides the electrode constraint, we must also satisfy the static and dynamic fluidic constraints for correct droplet movement [9], [10]. The static fluidic constraint states that the minimum spacing between two droplets must be one cell, while the dynamic fluidic constraint is related to two moving droplets. To satisfy the dynamic fluidic constraint, if a droplet d moves to cell (x, y) at time $t + 1$, then there must be no other droplet d' which can move from one of the eight neighboring cells of (x, y) to another cell at time $t + 1$.

Fig. 4 shows the dynamic fluidic constraint. In the figure, cell (x, y) must be activated for droplet d, since a droplet moves to a cell only if that cell is activated. However, the electrode constraint dictates that cell (x, y) cannot be activated; otherwise, droplet d' can move incorrectly. In other words, enforcing the electrode constraint for all drops ensures that the dynamic fluidic constraint is automatically satisfied (however,

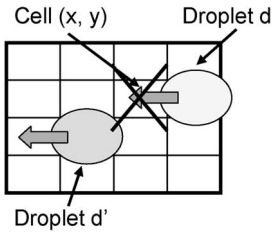


Fig. 4. Illustration of the dynamic fluidic constraint.

it is easy to show that the reverse is not true). As a result, our electrode constraints subsume the dynamic fluidic constraints, and we do not have to explicitly specify the latter.

D. Problem Formulation

In this paper, we focus on droplet-transportation problem. As stated in [9] and [10], the droplet-transportation problem can be represented in a 3-D space, where the third dimension corresponds to time. At each time step, the problem reduces to working with the droplet-movement problem in a 2-D plane. We focus the problem formulation on one 2-D plane, noting that similar constraints may be added for every other 2-D plane. When routing droplets on one 2-D plane, we consider the modules (and the surrounding segregation cells) that are active as obstacles. Besides, for practical bioassays, we must be able to handle three-pin nets to represent the fact that two droplets may have to be merged during their transportation for efficient mix operations [9], [14]. Therefore, the droplet routing problem for each 2-D plane can be formulated as follows:

Input: *A netlist of m nets $N = \{n_1, n_2, \ldots, n_m\}$, where each net n_i is a two-pin net (one droplet)or a three-pin net (two droplets), and the locations of pins and obstacles.*

Objective: *Route all droplets from their source pins to their target pins while minimizing the maximum time to route all droplets.*

Constraint: *Both fluidic and electrode constraints must be satisfied.*

III. ILP Formulation for Droplet Routing

In this section, we present the basic ILP formulation for one 2-D plane. We show how the ILP optimizes droplet routing and scheduling and voltage assignment on the electrodes, with the consideration of three-pin nets, and then, we analyze the computational complexity of the formulation and propose a practical approach for problem size reduction.

A. Basic ILP Formulation

Based on the modeling of electrode constraint shown in Fig. 3, the electrode constraint can be modeled by the following three rules.

1) If a droplet is at cell (x, y) at time t, then the four diagonally adjacent cells cannot be activated at time $t + 1$.
2) If a droplet moves to cell (x', y') at time $t + 1$, then the eight neighboring cells of (x', y') cannot be activated at $t + 1$.

TABLE I
Notations Used in the ILP Formulation

$\hat{W}/\hat{H}$	biochip width/height
T	maximum droplet transportation time
N'	set of 3-pin nets
D	set of droplets
d_j^i	j-th droplet of net n_i; $j = \{1, 2\}$
$(s_x^{i,j}, s_y^{i,j})$	location of the source of d_j^i
$(\hat{t}_x^i, \hat{t}_y^i)$	location of the sink of net n_i
$(x_t^{i,j}, y_t^{i,j})$	location of droplet d_j^i at time t
C	set of available cells (cells that can be used for routing)
$E(x, y)$	set of cell (x, y) and its four adjacent cells
$E'(x, y)$	set of cell (x, y)'s four adjacent cells
$\hat{E}(x, y)$	set of available neighboring cells of (x, y)
$E'_x(x, y)/E'_y(x, y)$	set of available cells adjacent to (x, y) with the same x- (y-) coordinate
$\hat{D}(x, y)$	set of available diagonal cells of (x, y)
T_l	actual maximum droplet transportation time
$p_j^i(x, y, t)$	a 0-1 variable to represent that droplet d_j^i is located at (x, y) at time t
$L_x^c(t)$	a 0-1 variable represents that column x is set to voltage low at time t
$H_x^c(t)$	a 0-1 variable represents that column x is set to voltage high at time t
$L_y^r(t)$	a 0-1 variable represents that row y is set to voltage low at time t
$H_y^r(t)$	a 0-1 variable represents that row y is set to voltage high at time t
$a_t^{x,y}$	a 0-1 variable represents that (x, y) is activated at time t
m_t^i	a 0-1 variable represents that d_0^i and d_1^i are merged at time t

3) If a droplet is at cell (x, y) at time t, at most one cell can be activated among cells (x, y) and its four adjacent cells.

Note that both rules 1) and 2) state that the cells diagonally adjacent to cell (x, y) cannot be activated at time $t + 1$. This redundancy reduces the size of the basic ILP formulation; otherwise, extra variables are required to represent the moving direction of each variable $p_j^i(x, y, t)$ and extra constraints to determine the values of these extra variables.

In the following sections, we introduce the objective function and constraints of the basic ILP formulation. The notations used in the proposed ILP formulation are shown in Table I.

1) Objective function: The goal is to minimize the latest time a droplet reaches its sink. Therefore, the objective function is defined by the following:

$$\text{Minimize} : T_l. \tag{1}$$

2) Constraints: There are totally eight constraints in the basic ILP formulation.

Objective-function computation: If a droplet reaches its sink at time $t + 1$, then the time when it reaches its sink can be computed as $t + 1$ times the difference of the two variables $p_j^i(\hat{t}_x^i, \hat{t}_y^i, t + 1)$ and $p_j^i(\hat{t}_x^i, \hat{t}_y^i, t)$. Therefore, the objective function can be computed by the following constraint:

$$(t + 1)\left(p_j^i\left(\hat{t}_x^i, \hat{t}_y^i, t + 1\right) - p_j^i\left(\hat{t}_x^i, \hat{t}_y^i, t\right)\right) \leq T_l \quad \forall d_j^i \in D,\ 0 \leq t < T. \tag{2}$$

Source and sink requirements: We assume that at time zero, all droplets are at their source locations. All droplets must

reach their sinks. Once a droplet reaches its sink, it remains there. Therefore, the earlier requirements can be represented by the following constraints:

$$p_j^i\left(s_x^{i,j}, s_y^{i,j}, 0\right) = 1 \qquad \forall\, d_j^i \in D \tag{3}$$

$$\sum_{t=0}^{T-1} p_j^i\left(\hat{t}_x^i, \hat{t}_y^i, t\right) \geq 1 \qquad \forall\, d_j^i \in D \tag{4}$$

$$p_j^i\left(\hat{t}_x^i, \hat{t}_y^i, t\right) - p_j^i\left(\hat{t}_x^i, \hat{t}_y^i, t+1\right) \leq 0 \qquad \forall\, d_j^i \in D,\ 0 \leq t < T. \tag{5}$$

Exclusivity constraint: The exclusivity constraint states that, at each time step, a droplet only has one location and can be represented by the following constraint:

$$\sum_{(x,y)} p_j^i(x,y,t) = 1, \qquad (x,y) \in C\, \forall\, d_j^i \in D,\ 0 \leq t < T. \tag{6}$$

Static fluidic constraint: To satisfy the static fluidic constraint, the minimum spacing between two droplets must be one cell. In other words, there are no other droplets in the 3×3 region centered by a droplet. The static fluidic constraint can be modeled by the following constraint:

$$p_j^i(x,y,t) + p_{j'}^{i'}(x',y',t) \leq 1 \qquad \forall\, d_j^i,\ d_{j'}^{i'} \in D,\ (x,y) \in C, (x',y') \in \hat{E}(x,y),\ 0 \leq t < T. \tag{7}$$

Voltage assignment and cell activation: Each row/column can be assigned one voltage (high or low), or left floating, at any time. A cell is activated if and only if it is in the intersection of a row with high (low) voltage and a column with low (high) voltage. Therefore, this constraint can be modeled by the following constraints:

$$L_x^c(t) + H_x^c(t) \leq 1, \qquad 0 \leq x < \hat{W},\ 0 \leq t < T \tag{8}$$

$$L_y^r(t) + H_y^r(t) \leq 1, \qquad 0 \leq y < \hat{H},\ 0 \leq t < T \tag{9}$$

$$\left(L_x^c(t) \text{ and } H_y^r(t)\right) \text{ or } \left(H_x^c(t) \text{ and } L_y^r(t)\right) \leftrightarrow a_t^{x,y} \qquad \forall (x,y) \in C,\ 0 \leq t < T. \tag{10}$$

We use the following five equations and two additional variables to linearize constraint (10):

$$L_x^c(t) + H_y^r(t) - 2b_1^{x,y}(t) \geq 0 \qquad \forall\, (x,y) \in C,\ 0 \leq t < T \tag{11}$$

$$L_x^c(t) + H_y^r(y) - b_1^{x,y}(t) \leq 1 \qquad \forall\, (x,y) \in C,\ 0 \leq t < T \tag{12}$$

$$H_x^c(t) + L_y^r(t) - 2b_2^{x,y}(t) \geq 0 \qquad \forall\, (x,y) \in C,\ 0 \leq t < T \tag{13}$$

$$H_x^c(t) + L_y^r(y) - b_2^{x,y}(t) \leq 1 \qquad \forall\, (x,y) \in C,\ 0 \leq t < T \tag{14}$$

$$a_t^{x,y} - b_1^{x,y}(t) - b_2^{x,y}(t) = 0 \qquad \forall\, (x,y) \in C,\ 0 \leq t < T \tag{15}$$

where $b_1^{x,y}(t)$ and $b_2^{x,y}(t)$ are two binary variables. $b_1^{x,y}(t) = 1$ ($b_2^{x,y}(t) = 1$) means that cell (x,y) is activated, and column x is assigned to low (high) voltage, and row y is assigned to low (high) voltage. Note that $b_1^{x,y}(t)$ and $b_2^{x,y}(t)$ cannot be one at the same time, since a row/column can be assigned to only one voltage at a time.

Droplet-movement constraint: A droplet can only move to an adjacent cell, which must be activated before the movement occurs. On the other hand, if a droplet is to stay at its original location, the corresponding cell may or may not be activated. Therefore, the droplet-movement constraint can be represented by the following constraints:

$$\sum_{(x',y') \in E(x,y)} p_j^i(x',y',t+1) - p_j^i(x,y,t) \geq 0 \qquad \forall\, d_j^i \in D,\ (x,y) \in C,\ 0 \leq t < T-1 \tag{16}$$

$$p_j^i(x,y,t+1) + \sum_{(x',y') \in E'(x,y)} p_j^i(x',y',t) - a_{t+1}^{x,y} \leq 1 \qquad \forall d_j^i \in D,\ (x,y) \in C,\ 0 \leq t < T-1. \tag{17}$$

Note that activating a cell does not imply that a droplet will move to this cell, as shown in the case of the top-right extra activated cell in Fig. 2. Therefore, constraint (17) does not state an "if-and-only-if" relation between cell activation and droplet movement.

Electrode constraint: The three rules explained earlier can be represented as the following constraints, where constraint (18) represents rule 1), constraint (19) represents rule 2), and constraint (20) represents rule 3)

$$p_j^i(x,y,t) + a_{t+1}^{x',y'} \leq 1 \qquad \forall\, d_j^i \in D,\ (x,y) \in C,\ (x',y') \in \hat{D}(x,y),\ 0 \leq t < T-1 \tag{18}$$

$$9p_j^i(x,y,t) + \sum_{(x',y') \in \hat{E}(x,y)} a_t^{x',y'} \leq 9 \qquad \forall d_j^i \in D,\ (x,y) \in C,\ 0 \leq t < T \tag{19}$$

$$6p_j^i(x,y,t) + \sum_{(x',y') \in E(x,y)} a_{t+1}^{x',y'} \leq 7 \qquad \forall d_j^i \in D,\ (x,y) \in C,\ 0 \leq t < T-1. \tag{20}$$

Three-pin nets: We use the following three constraints to handle the three-pin nets:

$$\sum_{(x,y)} \left(p_0^i(x,y,t) - p_1^i(x,y,t)\right) = 0 \leftrightarrow m_t^i = 1 \qquad \forall n_i \in N',\ (x,y) \in C,\ 1 \leq t < T \tag{21}$$

$$\sum_{t=1}^{T-1} m_t^i \geq 1 \qquad \forall\, n_i \in N' \tag{22}$$

$$\sum_{t=1}^{T-2} (t+1)\left(m_{t+1}^i - m_t^i - p_0^i\left(\hat{t}_x^i, \hat{t}_y^i, t+1\right) + p_0^i\left(\hat{t}_x^i, \hat{t}_y^i, t\right)\right) \leq -1 \qquad \forall\, n_i \in N'. \tag{23}$$

Constraint (21) is used to determine whether two droplets are merged, i.e., in the same physical location. Constraint (22) is used to guarantee that two droplets must be merged during their transportation by restricting and that their physical location must be the same for at least one time step. Constraint (23) states that these two droplets must be merged before reaching

their sink. We use the following five equations and two additional variables to linearize constraint (21):

$$\sum_{(x,y)} \left(p_0^i(x,y,t) - p_1^i(x,y,t)\right) - (\hat{W}\hat{H}) \times \hat{m}_1^i(t) \geq 1 - (\hat{W}\hat{H}) \quad \forall\, n_i \in N',\ (x,y) \in C,\ 1 \leq t < T \tag{24}$$

$$\sum_{(x,y)} \left(p_0^i(x,y,t) - p_1^i(x,y,t)\right) - (\hat{W}\hat{H}) \times \hat{m}_1^i(t) \leq 0 \quad \forall\, n_i \in N',\ (x,y) \in C,\ 1 \leq t < T \tag{25}$$

$$\sum_{(x,y)} \left(p_0^i(x,y,t) - p_1^i(x,y,t)\right) + (\hat{W}\hat{H}) \times \hat{m}_2^i(t) \geq (\hat{W}\hat{H}) - 1 \quad \forall\, n_i \in N',\ (x,y) \in C,\ 1 \leq t < T \tag{26}$$

$$\sum_{(x,y)} \left(p_0^i(x,y,t) - p_1^i(x,y,t)\right) + (\hat{W}\hat{H}) \times \hat{m}_2^i(t) \leq 0 \quad \forall\, n_i \in N',\ (x,y) \in C,\ 1 \leq t < T \tag{27}$$

$$m_t^i + \hat{m}_1^i(t) + \hat{m}_2^i(t) = 1 \tag{28}$$

where $\hat{m}_1^i(t)$ and $\hat{m}_2^i(t)$ are two binary variables. $\hat{m}_1^i(t)$ is one if $\sum_{(x,y)}(p_0^i(x,y,t) - p_1^i(x,y,t))$ is greater than or equal to zero. Similarly, $\hat{m}_2^i(t)$ is one if $\sum_{(x,y)}(p_0^i(x,y,t) - p_1^i(x,y,t))$ is less than or equal to zero. Therefore, these two variables cannot be one at the same time. Note that once these two droplets are merged, they will always move together by the droplet-movement constraint. Therefore, these two droplets are not splitting once they are merged.

B. Complexity Analysis

It is easy to verify that the number of variables that represent the droplet location dominates the total number of variables. The number of variables that represents the locations of droplets at each time is $O(|D|\hat{W}\hat{H}T)$, and therefore, this is the order of the total number of variables in the basic ILP formulation.

It can be verified that the fluidic constraint (7) introduces the maximum number of constraints, since it models the relationship between every pair of droplets. This amounts to a total of $O(|D|^2\hat{W}\hat{H}T)$ constraints, and this is the order of the number of constraints in the basic ILP formulation.

Based on the earlier discussion, we present the following observation: *Given a set of nets N of one 2-D plane, a biochip of the width (height) $\hat{W}$ $(\hat{H})$, and the maximum droplet-transportation time T, the number of variables and constraints of the basic ILP formulation for one 2-D plane are $O(|D|\hat{W}\hat{H}T)$ and $O(|D|^2\hat{W}\hat{H}T)$, respectively.*

C. Implementational Issues

We now present techniques for practically reducing the number of variables/constraints, without affecting the optimality of the basic ILP formulation. We use the concept of the idle interval presented in [10] to reduce unnecessary variables. The idle interval $I_j^i(x,y) = [T^m(x,y,d_j^i), T^M(x,y,d_j^i)]$ represents all possible times a droplet d_j^i will be at cell (x,y) with a time limitation on the latest time when d_j^i must reach its sink. Here, $T^m(x,y,d_j^i)$ represents the earliest time when d_j^i reaches (x,y) from its source, and $T^M(x,y,d_j^i)$ represents the latest time when d_j^i can stay at (x,y) without violating the time limitation. If T is the time limitation, we can eliminate variables $p_j^i(x,y,t)$ if $t < T^m(x,y,d_j^i)$ or $t > T^M(x,y,d_j^i)$, since d_j^i cannot reach (stay at) cell (x,y) earlier than $T^m(x,y,d_j^i)$ (later than $T^M(x,y,d_j^i)$).

With this approach, we can also eliminate unnecessary constraints. For example, for every variable $p_j^i(x,y,t)$, if all variables $p_{j'}^{i'}(x',y',t)$, $(x',y') \in \hat{E}(x,y)$ are eliminated, then we can eliminate the static fluidic constraint associated with $p_j^i(x,y,t)$. Note that if all variables $p_j^i(x',y',t+1)$, $(x',y') \in \hat{E}(x,y)$ are eliminated, then we add one extra constraint $p_j^i(x,y,t) = 0$ in the basic ILP formulation. Otherwise, we may obtain an infeasible solution with $p_j^i(x,y,t) = 1$ and $p_j^i(x',y',t+1) = 1$, but $(x',y') \notin E(x,y)$, since the droplet-movement constraint is eliminated for $p_j^i(x,y,t)$.

Algorithm: Progressive-ILP routing
Let T_l be the maximum Manhattan distance of all droplets from their sources to sinks;
begin
while not all droplets reach their sinks
  Compute_droplet_movement_cost();
  Construct_ILP();
  Solve_ILP();
  Update_droplet_position();
  if it is not possible to route all droplets within T_l
    T_l += $\alpha\times$ maximum Manhattan distance;
  Set T_l as the maximum droplet transportation time;
end

Fig. 5. Overview of progressive-ILP method.

IV. Progressive-ILP Routing Scheme

Although the basic ILP formulation presented in the previous section can solve the droplet-transportation problem, it may incur high runtimes. Hence, it may be hard to directly apply the basic ILP formulation to practical bioassays. In this section, we present a progressive-ILP routing scheme to solve the droplet-transportation problem. The main idea is to divide the original problem into a set of subproblems corresponding to each time step. The goal of each subproblem is to find min-cost locations of the droplets at next time step by ILP. In the following sections, we first overview the progressive-ILP routing scheme and, then, present the ILP formulation. Next, we present approaches for handling three-pin nets and determining the droplet-movement cost. Finally, we present a complexity analysis and some implementation details.

A. Progressive Routing Algorithm Overview

Fig. 5 shows the overview of the progressive routing scheme. The essential intuition is that this scheme finds the optimal movement of droplets progressively, one time step at a time. We first set T_l as the maximum Manhattan distance of all droplets from their sources and sinks. Until all droplets reach their sinks, we first calculate the droplet-movement cost, and then, we construct the progressive-ILP formulation and solve it by an ILP solver. Finally, if it is found that some droplets

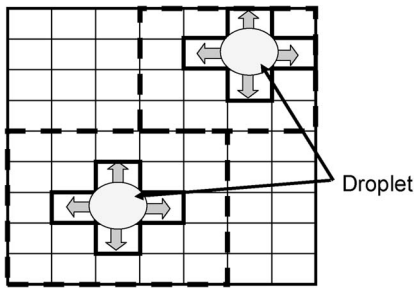


Fig. 6. Illustration of the locality of droplet movement and electrode constraint when locations of droplets are known.

cannot be routed within time T_l, then T_l is increased by α times the maximum Manhattan distance, where α is a user-specified constant and is set to be 0.2.

B. Progressive-ILP Formulation

The key idea of the progressive-ILP method is to reduce the problem to a manageable size by solving a series of subproblems. The problem size can be further reduced by observing that the electrode and static fluidic constraints and droplet movement have only local effects when the locations of the droplets are known. Fig. 6 shows a biochip with two droplets, where the arrows represent possible directions in which the droplets can move. Since the movement of a droplet d_j^i can only be affected by its neighboring cells and their adjacent cells, the progressive ILP does not have to consider the effects of activated cells outside the 5×5 region around a droplet. To satisfy the static fluidic constraint, we observe that, after a droplet moves, two droplets are adjacent to each other only if a cell with more than one droplet nearby is activated. Therefore, we do not activate this cell in the progressive-ILP formulation. Finally, each droplet d_j^i can either move to one of its four adjacent cells or stay at its original location. That is, the possible position of a droplet can be represented as a "cross" shown in Fig. 6.

Therefore, variables $a_{t+1}^{x,y}$ are required to represent the cell $(x_t^{i,j}, y_t^{i,j})$ and its four adjacent cells. Other cells either must be deactivated (within the 5×5 region) or have no effect on droplet movement (outside the 5×5 region). In this way, unnecessary variables can be eliminated. We now present the progressive-ILP formulation in the following sections. The notations used in the progressive-ILP formulation are also listed in Table I.

1) Objective function: The goal of the progressive-ILP formulation is to find the min-cost droplet movement. Therefore, the objective function can be represented as the following:

$$\text{Minimize} \sum_{d_j^i \in D} \sum_{(x,y) \in E\left(x_t^{i,j i}, y_t^{i,j}\right)} w_j^i(x,y) p_j^i(x,y,t+1) \quad (29)$$

where $w_j^i(x,y)$ is the cost when a droplet d_j^i moves to cell (x,y).

2) Constraints: There are five constraints in the progressive-ILP formulation.

Droplet-movement constraint: A droplet can move to one of its four adjacent cells if and only if this cell is activated. Therefore, this constraint can be expressed by the following constraint:

$$p_j^i(x,y,t+1) \leftrightarrow a_{t+1}^{x,y} \qquad \forall d_j^i \in D \quad \forall (x,y) \in E'\left(x_t^{i,j}, y_t^{i,j}\right). \quad (30)$$

The above constraint can easily be linearized by the following:

$$p_j^i(x,y,t+1) \le a_{t+1}^{x,y} \qquad \forall d_j^i \in D \; \forall (x,y) \in E'(x_t^{i,j}, y_t^{i,j}) \quad (31)$$

$$a_{t+1}^{x,y} \le p_j^i(x,y,t+1) \qquad \forall d_j^i \in D \; \forall (x,y) \in E'(x_t^{i,j}, y_t^{i,j}). \quad (32)$$

Note that since only the four adjacent cells are associated with cell-activation variables, the cell activation and the droplet movement must be modeled as an "if-and-only-if" relation, unlike the "if" relation in the basic ILP formulation. Moreover, a cell stays at its original location if and only if all its four adjacent cells are deactivated. Therefore, we use the following constraints to represent this situation:

$$p_j^i\left(x_t^{i,j}, y_t^{i,j}, t+1\right) + \sum_{(x,y) \in E'\left(x_t^{i,j}, y_t^{i,j}\right)} a_{t+1}^{x,y} \ge 1 \qquad \forall d_j^i \in D \quad (33)$$

$$4p_j^i\left(x_t^{i,j}, y_t^{i,j}, t+1\right) + \sum_{(x,y) \in E'\left(x_t^{i,j}, y_t^{i,j}\right)} a_{t+1}^{x,y} \le 4 \qquad \forall d_j^i \in D. \quad (34)$$

Electrode constraint: Rule 1), which states that the diagonal cells cannot be activated, can be represented as

$$L_x^c(t+1) + H_y^r(t+1) \le 1 \qquad \forall (x,y) \in \hat{D}\left(x_t^{i,j}, y_t^{i,j}\right) \quad (35)$$

$$H_x^c(t+1) + L_y^r(t+1) \le 1 \qquad \forall (x,y) \in \hat{D}\left(x_t^{i,j}, y_t^{i,j}\right). \quad (36)$$

The following two constraints, similar to constraint (19), are used for rule 2):

$$\text{if } p_j^i(x,y,t+1) = 1 \rightarrow L_{x'}^c(t+1) + H_{y'}^r(t+1) \le 1 \text{ or } H_{x'}^c(t+1) + L_{y'}^r(t+1) \le 1$$
$$\forall d_j^i \in D \; \forall (x,y) \in E_x'\left(x_t^{i,j}, y_t^{i,j}\right) \; \forall (x',y') \in E_y'(x,y) \quad (37)$$

$$\text{if } p_j^i(x,y,t+1) = 1 \rightarrow L_{x'}^c(t+1) + H_{y'}^r(t+1) \le 1 \text{ or } H_{x'}^c(t+1) + L_{y'}^r(t+1) \le 1$$
$$\forall d_j^i \in D \; \forall (x,y) \in E_y'\left(x_t^{i,j}, y_t^{i,j}\right) \; \forall (x',y') \in E_x'(x,y). \quad (38)$$

As shown in Fig. 3, if d_j^i moves to cell $(x+1,y)$, then cells $(x+2,y)$, $(x+2,y-1)$, and $(x+2,y+1)$ cannot be activated. A similar condition holds for cell $(x-1,y)$. Constraint (37) is used to represent these two cases, and constraint (38) is used to represent the case when d_j^i moves to cell $(x_t^{i,j}, y_t^{i,j}+1)$

or $(x_t^{i,j}, y_t^{i,j} - 1)$. We use the following two equations to linearize constraint (37):

$$p_j^i(x,y,t+1) + L_{x'}^c(t+1) + H_{y'}^r(t+1) \leq 2 \quad \forall d_j^i \in D\, \forall (x,y) \in E_x'\left(x_t^{i,j}, y_t^{i,j}\right)\ \forall (x',y') \in E_y'(x,y) \tag{39}$$

$$p_j^i(x,y,t+1) + H_{x'}^c(t+1) + L_{y'}^r(t+1) \leq 2 \quad \forall d_j^i \in D\, \forall (x,y) \in E_x\left(x_t^{i,j}, y_t^{i,j}\right)\ \forall (x',y') \in E_y'(x,y). \tag{40}$$

Similarly, the same method can be used to linearize constraint (38). Finally, for rule 3), we impose the following constraint:

$$\sum_{(x',y') \in E\left(x_t^{i,j}, y_t^{i,j}\right)} a_{t+1}^{x',y'} \leq 1 \qquad \forall d_j^i \in D. \tag{41}$$

Static fluidic constraint: Since the locations of the droplets are known, we model the static fluidic constraint in a simpler way. The fluidic constraint is violated only if a cell with more than one droplet nearby is activated. Therefore, instead of using constraint (7), we represent the fluidic constraint as follows:

if more than one droplet is around cell (x,y)

$$a_{t+1}^{x,y} = 0 \qquad \forall (x,y) \in E'\left(x_t^{i,j}, y_t^{i,j}\right)\ \forall d_j^i \in D. \tag{42}$$

Sink requirement: Once a droplet reaches its sink, it must stay there. The sink requirement can be modeled by the following constraint:

$$p_j^i\left(x_t^{i,j}, y_t^{i,j}, t\right) - p_j^i\left(x_{t+1}^{i,j}, y_{t+1}^{i,j}, t+1\right) \leq 0 \qquad \forall d_j^i \in D \tag{43}$$

which means that if d_j^i reaches its sink at time t, then it must stay there at time $t+1$.

Voltage assignment and cell activation: This constraint is the same as that in the basic ILP formulation.

C. Droplet-Movement Cost

A key issue in the progressive-ILP routing scheme is the determination of the droplet-movement cost. Since the goal is to minimize the maximum droplet-transportation time, the objective is to move a droplet toward its sink at each time step. Furthermore, congestion must be avoided among droplets; otherwise, either a detour occurs or some droplets stall for a period of time, to satisfy both the fluidic and electrode constraints. Therefore, the droplet-transportation time is potentially increased. In this section, we detail how to determine the droplet-movement cost.

The droplet-movement cost $w_j^i(x,y)$ when a droplet d_j^i moves to cell (x,y) consists of routing and congestion costs and is defined as follows:

$$w_j^i(x,y) = \beta r_j^i\left(x,y,\hat{t}_x^i,\hat{t}_y^i\right) + \gamma g_j^i(x,y,\hat{t}_x^i,\hat{t}_y^i) \tag{44}$$

where $r_j^i(x,y,\hat{t}_x^i,\hat{t}_y^i)$ is the routing cost and $g_j^i(x,y,\hat{t}_x^i,\hat{t}_y^i)$ is the congestion cost. β and γ are user-specified constants. In our implementation, we empirically set $\beta = 15$ and $\gamma = 0.1$. Fig. 7 shows the droplet-movement cost computation. Routing cost is the distance from cell (x,y) to the sink of d_j^i and is defined as follows:

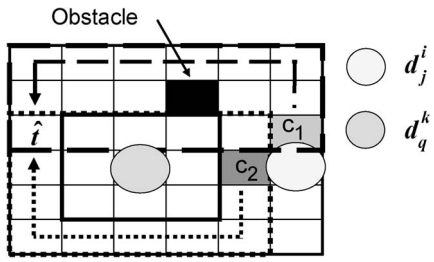


Fig. 7. Droplet-movement cost computation illustration. c_1 and c_2 are two possible destinations of a droplet. Black regions are obstacles.

$$r_j^i\left(x,y,\hat{t}_x^i,\hat{t}_y^i\right) = \frac{d^r\left(x,y,\hat{t}_x^i,\hat{t}_y^i\right)}{\hat{W}\hat{H}} \tag{45}$$

where $d^r(x,y,\hat{t}_x^i,\hat{t}_y^j)$ is the real routing distance from cell (x,y) to cell $(\hat{t}_x^i,\hat{t}_y^j)$. We use the A^*-search algorithm to determine this routing distance while considering all other droplets except d_j^i as a 3×3 obstacle due to static fluidic constraint. The routing cost of a cell (x,y) is the actual distance between the source and (x,y) and the half perimeter of the bounding box defined by (x,y) and the target pin. The use of the A^*-search permits us to obtain an accurate estimation of the routing distance. For example, in Fig. 7, the dotted line represents the routing path obtained by the A^*-search algorithm. Since the 3×3 region centered by the droplet d_q^k is considered as an obstacle, the minimum distance between cell c_2 and the sink for d_j^i is nine.

Note that one special case for routing cost computation is that another droplet $d_{j'}^{i'}$ is temporarily located at the neighboring cells of the sink of d_j^i. Therefore, it is not possible to reach the sink of d_j^i due to $d_{j'}^{i'}$ being modeled a 3×3 obstacle. In this case, we use the minimum of the real distance from the sink to a cell plus the Manhattan distance from this cell to the sink as $d^r(x,y,\hat{t}_x^i,\hat{t}_y^j)$. In this way, we can accurately estimate the routing cost if the sink of d_j^i is temporarily not reachable.

The goal of the congestion-cost component is to reduce the probability of the violations of the electrode and fluidic constraints. In this paper, we also use the idle interval $I_j^i(x,y) = [T^m(x,y,d_j^i), T^M(x,y,d_j^i)]$ for congestion-cost computation. If T_l is set to be the time limitation, the congestion cost of a cell can be measured as the length of the idle interval of d_j^i on cell (x,y) over the summation of the length of the idle intervals of other droplets on cell (x,y). Instead of using the source location of a droplet, as in [10], we use the possible destination cell (x,y) to compute the idle interval. The values of $T^m(x,y,d_j^i)$ and $T^M(x,y,d_j^i)$ for a subproblem at time t can be computed by the following:

$$T^m\left(x,y,d_j^i\right) = t + d^m\left(x,y,\hat{t}_t^x,\hat{t}_t^y\right) \tag{46}$$

$$T^M\left(x,y,d_j^i\right) = T_l - d^m\left(x,y,\hat{t}_t^x,\hat{t}_t^y\right) \tag{47}$$

where $d^m(x,y,x',y')$ is the Manhattan distance between cells (x,y) and (x',y'). The congestion information is updated at every time step for the latest congestion information.

To obtain more accurate congestion information, it is not sufficient to only consider the congestion information of the possible destination cell (x, y). In this paper, we consider the congestion information of all cells within the bounding box defined by cell (x, y) and its sink. Note that we use the real bounding box computed by the A^*-search algorithm. For example, as shown in Fig. 7, when computing the congestion cost of cell c_1, we consider all cells within the dashed region, which represents the real bounding box. Therefore, the congestion cost $g_j^i(x, y, \hat{t}_x^i, \hat{t}_y^i)$ when droplet d_j^i moves to cell (x, y) is defined by the following:

$$g_j^i\left(x, y, \hat{t}_x^i, \hat{t}_y^i\right) = \sum_{(x',y')\in b_j^i(x,y)} \frac{\left|I_j^i(x', y')\right|}{\sum_{d_{j'}^{i'}\in D}\left|I_{j'}^{i'}(x, y)\right|} \tag{48}$$

where $b_j^i(x, y)$ is the bounding box obtained by the A^*-search algorithm and $|I_j^i(x, y)|$ is the length of the idle interval defined as the difference of its two end points plus one. Note that if $T^m(d_j^i) > T^M(d_j^i)$, then $|I_j^i(x, y)|$ is zero. If the congestion cost is large, it is likely that the electrode and fluidic constraints will be violated if d_j^i moves to cell (x, y). Therefore, the proposed routing algorithm will not tend to move a droplet to cell (x, y).

D. Handling Three-Pin Nets

For a three-pin net n_i, the two droplets must be mixed during their transportation. To guarantee that these two droplets are mixed, first, these two droplets route toward each other. After they merged, we consider them as one droplet and route them toward their sink. Therefore, for droplet-movement-cost computation, the sink of d_0^i (d_1^i) is the location of d_1^i (d_0^i) before merging.

E. Complexity Analysis

For each droplet, five variables are used to represent its position at next time step, and five variables are used to capture cell activation. The number of variables for row/column-voltage assignment is $O(\hat{W}\hat{H})$. Therefore, the total number of variables for the progressive-ILP formulation is $O(|D| + \hat{W}\hat{H})$.

There are a total of $O(\hat{W}\hat{H})$ constraints for row/column-voltage assignment. It is easy to see that the number of other constraints is proportional to the number of droplets. Therefore, the number of constraint in the progressive-ILP formulation is $O(|D| + \hat{W}\hat{H})$.

Based on the earlier analysis, we have the following observation: *Given a set of droplets D and a biochip of the width (height) $\hat{W}$ ($\hat{H}$), the number of variables and constraints of the progressive-ILP formulation for one problem instance (one time step) are $O(|D| + \hat{W}\hat{H})$ and $O(|D| + \hat{W}\hat{H})$, respectively.*

F. Implementational Issues

We now present several implementation details associated with obtaining a fast and reasonable solution to this problem. To further reduce the problem size, we add a restriction that a droplet can only move toward its sink, except in the following two cases. First, if a droplet cannot move toward its sink due to obstacles, we allow the droplet to move in all possible directions. Second, if the cell with minimum droplet-movement cost is not toward the sink, we allow d_j^i to move to this cell.

The knowledge that d_j^i will definitely not move to some cells can be used to further reduce the number of variables/constraints. For example, if d_j^i will not move to cell $(x_t^{i,j} - 1, y_t^{i,j})$, then we can eliminate the variables $p_j^i(x_t^{i,j} - 1, y_t^{i,j}, t + 1)$ and $a_t^{x_t^{i,j}-1, y_t^{i,j}}$. We can also eliminate the electrode constraint for these cells. Finally, the voltage assignment and cell-activation constraint is replaced by the following two constraints to ensure that this cell is not activated:

$$L^c_{x_t^{i,j}-1}(t + 1) + H^r_{y_t^{i,j}}(t + 1) \le 1 \tag{49}$$

$$H^c_{x_t^{i,j}-1}(t + 1) + L^r_{y_t^{i,j}}(t + 1) \le 1. \tag{50}$$

TABLE II
STATISTICS OF THE ROUTING BENCHMARKS

Circuit	Chip dimension	#2D planes	#Tnets
Diagnostics_1	16×16	11	28
Diagnostics_2	14×14	15	35
Protein_1	21×21	64	181
Protein_2	13×13	78	178

TABLE III
COMPARISON BETWEEN BASIC ILP AND PROGRESSIVE-ILP FORMULATIONS

Circuit	max. #vari.		max. #const.		CPU time	
	Basic ILP	Prog. ILP	Basic ILP	Prog. ILP	Basic ILP	Prog. ILP
Diagnostics_1	24889	138	176580	388	>120h	2.55s
Diagnostics_2	18061	134	126983	398	>120h	2.53s
Protein_1	49561	134	350645	357	>120h	15.36s
Protein_2	22238	132	252989	385	>120h	6.70s

V. SIMULATION RESULTS

Our progressive-ILP algorithm was implemented in the C++ language, and GLPK [15] is used as our ILP solver. For purposes of comparison, we have also implemented the clique-partitioning-based [11] and the graph-coloring-based [8] algorithms. All of the above algorithms and ILP formulations are executed on a 1.2-GHz SUN Blade-2000 machine with 8-GB memory. We evaluated our routing algorithm on the two practical bioassays: the *in vitro* diagnostics [9] and the colorimetric protein assay [16]. Table II shows the statistics of each benchmark [10]: column two shows the chip dimension, column three lists the total number of 2-D planes, and column four lists the total number of nets of all 2-D planes.

We performed two experiments to verify the efficiency and effectiveness of the progressive-ILP algorithm. In the first experiment, we compared the basic ILP with the reduction method presented in Section III-C and progressive-ILP algorithms. We generated both the basic and progressive-ILP formulations for each 2-D plane. The simulation results are listed in Table III. Columns two and three (columns four and five) list the maximum number of variables (constraints) of all problem instances. For the basic ILP, the problem instance is one 2-D plane, and for

TABLE IV
COMPARISON BETWEEN OUR APPROACH AND TWO METHODS BASED ON MAPPING DIRECT-ADDRESSING SOLUTIONS

Circuit	[10] + [11]		[10] + [8]		Progressive-ILP	
	Max/avg T_l (cycle)	CPU time (sec.)	Max/avg T_l (cycle)	CPU time (sec.)	Max/avg T_l (cycle)	CPU time (sec.)
Diagnostics_1	40 / 16.72	0.05	47 / 20.18	0.06	24 / 13.09	2.55
Diagnostics_2	35 / 13.46	0.05	52 / 16.80	0.06	22 / 11.00	2.53
Protein_1	48 / 19.32	0.26	55 / 24.40	0.30	26 / 16.15	15.36
Protein_2	36 / 11.00	0.20	53 / 14.33	0.21	26 / 10.23	6.70

the progressive-ILP, the problem instance is one time step. The results show that the basic ILP requires at least five days to solve all 2-D planes of one benchmark, which is not feasible for this problem; in contrast, the proposed progressive-ILP algorithm requires at most 15.36 s due to the significantly smaller problem size. For example, as shown in Table III, for the protein_1 benchmark, the progressive-ILP routing scheme can reduce the number of variables (constraints) by 99.72% (99.89%) for the largest problem instance. This result demonstrates the efficiency of the progressive-ILP formulation.

The proposed progressive-ILP formulation can obtain a near-optimal solution with much less CPU time as compared with the basic ILP formulation. For example, for one 2-D plane of the protein_1 benchmark with four droplets, the progressive-ILP formulation obtains a solution of 21 cycles in 0.40 s while the basic ILP formulation obtains a 20-cycle solution in 19 632 s. Since it is not feasible to solve the basic ILP for practical bioassays, it is not possible to compare the solution quality of these two algorithms directly. Nevertheless, it is observed that the difference in the solution quality for the two algorithms is at most three cycles for 41 2-D planes completed by the basic ILP formulation, and the average solution difference is only 1.68%. The result indicates that the progressive-ILP algorithm is close to the optimal solution.

In the second experiment, we verified the quality of the proposed progressive-ILP routing scheme. Since previous works require a direct-addressing routing solution for a fair comparison, we first generated the direct-addressing routing solution using the network-flow-based algorithm [10] and applied the previous approaches to obtain the final routing solution. Moreover, we modified the network-flow-based algorithm to minimize the droplet-transportation time. Table IV shows the simulation result. We report the maximum and average T_l of all 2-D planes and the CPU time to route all 2-D planes. The CPU times for [10] + [11] and [10] + [8] are the total CPU times to obtain the cross-referencing routing solution. As shown in this table, the progressive-ILP routing algorithm can obtain much smaller droplet-transportation times (in cycles) than the previous approaches, under reasonable CPU times. For example, compared with the [10] + [11] approach, the progressive-ILP algorithm can obtain up to 45.83% smaller maximum droplet-transportation time of all 2-D planes (48 cycles versus 26 cycles) with longer CPU time (0.26 s versus 15.36 s) for the protein_1 benchmark. Compared with the [10] + [8] approach, the progressive-ILP algorithm can obtain up to 59.61% smaller maximum droplet-transportation time of all 2-D planes (52 cycles versus 22 cycles) with longer CPU time (0.06 s versus 2.53 s) for the diagnostics_2 benchmark. This result

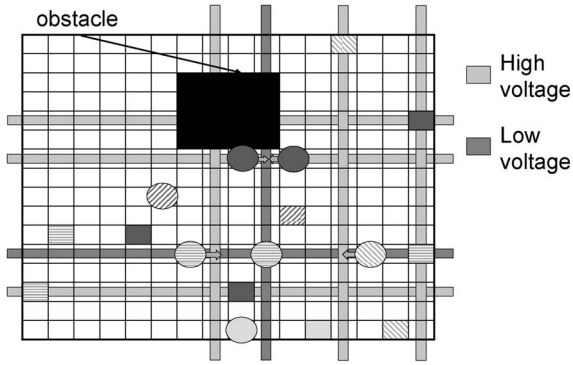


Fig. 8. Result of the diagnostics_1 benchmark.

demonstrates that the proposed algorithm is very effective for droplet routing on cross-referencing biochips. Fig. 8 shows the routing result of one 2-D plane of the diagnostics_1 benchmark at cycle eight.

There are two reasons why the progressive-ILP-based routing algorithm can obtain an improved solution as compared with previous works. The first is that our algorithms directly target droplet routing on cross-referencing biochips without going through the intermediate step of finding a solution that is optimal for direct-addressing routing. Therefore, this method has more flexibility than previous works in exploring the solution space. The second is that the proposed algorithm maximizes the number of droplet movements at each cycle by using the real routing distance as part of droplet-movement cost. To minimize cost, droplets tend to move to their sinks at each cycle. Moreover, our algorithm incorporates a probability-based congestion cost that avoid droplets to move to congested regions and, therefore, avoids additional detours or stalls that increase droplet-transportation time. In contrast, both [8] and [11] require a direct-addressing routing solution, and they are less flexible than our algorithm. Moreover, the technique proposed in [11] iteratively moves a set of droplets whose destination cells are in the same row/column. However, droplets whose destination cells are not in the same row/column can still move at the same time as long as they do not violate electrode constraint. Therefore, their algorithm may lose the optimality of minimizing droplet-transportation time.

Another difference is that the graph-coloring-based algorithm [8] assumes that four cells are activated when two droplets move at the same time. However, by properly assigning high/low voltages to each row/column, only two instead of four cells activate. For example, as shown in Fig. 2, the two extra cells can be eliminated by assigning high voltage to the row containing droplet three and low voltage to the column where

two extra activated cells are originally at. In other words, the algorithm in [8] does not exploit the ability of a biochip to assign voltages to each row/column to maximize the number of droplets movement at the same time.

VI. Conclusion

In this paper, we have proposed the first droplet routing algorithm that operates directly on cross-referencing biochips without a direct-addressing routing solution. We have also presented the basic ILP formulation to minimize the droplet-transportation time and the *progressive*-ILP routing scheme to iteratively determine the minimum-cost positions of the droplets at each time step. Simulation results have shown the efficiency and effectiveness of our algorithm.

The target biochip can be viewed as a passive-matrix-addressing biochip, since we must activate a row/column to activate a cell. There are other types of biochips that use active matrix addressing [17]. Each electrode has its unique addressing; therefore, we can uniquely activate an electrode without introducing the electrode-interference problem. However, the hardware and package costs may increase due to the extra control circuitry. We are currently trying to apply our droplet routing algorithm to these types of biochips.

References

[1] P.-H. Yuh, C.-L. Yang, and Y.-W. Chang, "A progressive-ILP based routing algorithm for cross-referencing biochips," in *Proc. Des. Autom. Conf.*, Jul. 2008, pp. 284–289.

[2] E. Verpoorte and N. F. de Rooij, "Microfluidics meets MEMS," *Proc. IEEE*, vol. 91, no. 6, pp. 930–953, Jun. 2003.

[3] F. Su, K. Chakrabarty, and R. B. Fair, "Microfluidic-based biochips: Technology issues, implementation platforms, and design-automation challenges," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 2, pp. 211–223, Feb. 2006.

[4] T. Xu and K. Chakrabarty, "Droplet-trace-based array partitioning and a pin assignment algorithm for the automated design of digital microfluidic biochips," in *Proc. Int. Conf. Hardware-Software Codes. Syst. Synthesis*, Oct. 2007, pp. 112–117.

[5] S.-K. Fan, C. Hashi, and C.-J. Kim, "Manipulation of multiple droplets on N × M grid by cross-reference EWOD driving scheme and pressure-contact packaging," in *Proc. Int. Conf. Micro Electro Mech. Syst.*, Jan. 2003, pp. 694–697.

[6] K. F. Böhringer, "Modeling and controlling parallel tasks in droplet-based microfluidic systems," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 2, pp. 334–344, Feb. 2006.

[7] M. Cho and D. Z. Pan, "A high-performance droplet router for digital microfluidic biochips," in *Proc. Int. Symp. Phys. Des.*, Apr. 2008, pp. 200–206.

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

[9] F. Su, W. Hwang, and K. Chakrabarty, "Droplet routing in the synthesis of digital microfluidic biochips," in *Proc. Des. Autom. Test Eur.*, 2006, pp. 323–328.

[10] P.-H. Yuh, C.-L. Yang, and Y.-W. Chang, "BioRoute: A network-flow based routing algorithm for digital microfluidic biochips," in *Proc. Int. Conf. Comput.-Aided Des.*, Nov. 2007, pp. 752–757.

[11] T. Xu and K. Chakrabarty, "A cross-referencing-based droplet manipulation method for high-throughput and pin-constrained digital microfluidic arrays," in *Proc. Des. Autom. Test Eur.*, Apr. 2007, pp. 552–557.

[12] M. Cho and D. Z. Pan, "Boxrouter: A new global router based on box expansion and progressive ILP," in *Proc. Des. Autom. Conf.*, Jul. 2006, pp. 373–378.

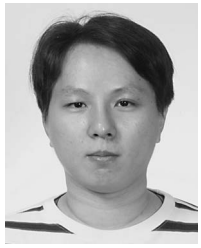
[13] J. Gong, S.-K. Fan, and C.-J. Kim, "Portable digital microfluidics platform with active but disposable lab-on-chip," in *Proc. Int. Conf. Micro Electro Mech. Syst.*, 2004, pp. 355–358.

[14] V. Srinivasan, V. Pamula, and R. Fair, "An integrated digital microfluidic lab-on-a-chip for clinical diagnostics on human physiological fluids," *Lab Chip*, vol. 4, no. 4, pp. 310–315, Aug. 2004.

[15] *The Network Simulator—ns-2*. [Online]. Available: http://www.gnu.org/software/glpk/

[16] F. Su and K. Chakrabarty, "Unified high-level synthesis and module placement for defect-tolerant microfluidic biochips," in *Proc. Des. Autom. Conf.*, Jun. 2005, pp. 825–830.

[17] P. Caillat, D. David, M. Belleville, F. Clerc, C. Massit, F. Revol-Cavalier, P. Peltie, T. Livache, G. Bidan, A. Roget, and E. Crapez, "Biochips on CMOS: An active matrix address array for DNA analysis," *Sens. Actuators B, Chem.*, vol. 61, no. 1–3, pp. 154–162, Dec. 1999.



Ping-Hung Yuh received the B.S. degree in computer science from National Chiao Tung University, Hsinchu, Taiwan, in 2002 and the Ph.D. degree in computer science from National Taiwan University (NTU), Taipei, Taiwan, in 2008.

From 2007 to 2008, he was a Visiting Scholar at the University of Minnesota, Minneapolis. He is currently with the Department of Computer Science and Information Engineering, NTU. His current research interests include temporal floorplanning for reconfigurable computing and biochip design automation, including placement and routing.

Sachin S. Sapatnekar (S'86–M'93–F'03) received the Ph.D. degree from the University of Illinois at Urbana–Champaign, Urbana, in 1992.

He is currently with the faculty of the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, where he holds the Henle Chair and the Distinguished McKnight University Professorship. He has authored several books and papers in the areas of timing and layout and has held positions on the editorial board of the IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, the IEEE Transactions on Circuits and Systems VII, IEEE Design and Test, and the IEEE Transactions on VLSI Systems. He has served on the technical program committees for various conferences, as Technical Program and General Chair for Tau and International Symposium on Physical Design, and is currently the Vice Chair for the Design Automation Conference 2009. He was the recipient of the NSF Career Award, five conference best paper awards, and the SRC Technical Excellence award.

Chia-Lin Yang (M'02) received the B.S. degree from National Taiwan Normal University, Taipei, Taiwan, in 1989, the M.S. degree from the University of Texas, Austin, in 1992, and the Ph.D. degree from the Department of Computer Science, Duke University, Durham, NC, in 2001.

In 1993, she joined VLSI Technology Inc. (currently Philips Semiconductors) as a Software Engineer. She is currently an Associate Professor with the Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan. Her research interests include energy-efficient microarchitectures, memory hierarchy design, and multimedia workload characterization.

Dr. Yang was the recipient of a 2000–2001 Intel Foundation Graduate Fellowship Award and the 2005 IBM Faculty Award.

Yao-Wen Chang (S'94–A'96–M'96) received the B.S. degree in computer science from National Taiwan University (NTU), Taipei, Taiwan, in 1988 and the M.S. and Ph.D. degrees in computer science from the University of Texas, Austin, in 1993 and 1996, respectively.

In the summer of 1994, he was with IBM T. J. Watson Research Center. From 1996 to 2001, he was with National Chiao Tung University (NCTU), Hsinchu, Taiwan. He is currently a Professor with the Department of Electrical Engineering and the Graduate Institute of Electronics Engineering, NTU. He is currently also a Visiting Professor at Waseda University, Kitakyushu, Japan. His current research interests include very large scale integration (VLSI) physical design, design for manufacturability/reliability, and design automation for biochips. He has been working closely with industry in these areas. He has coedited one textbook on electronic design automation and coauthored one book on routing and over 140 Association for Computing Machinery (ACM)/IEEE conference/journal papers in these areas.

Dr. Chang was a winner of the 2008 ACM International Symposium on Physical Design (ISPD) Global Routing Contest and the 2006 ACM ISPD Placement Contest. He was the recipient of a Best Paper Award at the International Conference on Computer Design (ICCD) 1995 and 12 Best Paper Award Nominations from Design Automation Conference (DAC) (four times), International Conference on Computer-Aided Design (ICCAD) (twice), ISPD (three times), ACM Transactions on Design Automation and Electronic Systems, Asia and South Pacific DAC (ASP-DAC), and ICCD in the past eight years. He was the recipient of many research awards, such as the 2007 Outstanding Research Award, the Inaugural 2005 First-Class Principal Investigator Award, and the 2004 Dr. Wu Ta You Memorial Award, all from National Science Council of Taiwan. He was also the recipient of the 2004 MXIC Young Chair Professorship from the MXIC Corporation and excellent teaching awards from NTU (four times) and NCTU. He is currently an Associate Editor for the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS (TCAD) and an Editor of the *Journal of Information Science and Engineering* (JISE). He has served on the ICCAD Executive Committee, the ACM/Special Interest Group on Design Automation (SIGDA) Physical Design Technical Committee, the ACM ISPD and IEEE Field Programmable Technology (FPT) Organizing Committees, and the technical program committees of ASP-DAC, DAC, Design Automation and Test on Europe, The International Conference on Field Programmable Logic and Applications (FPL), FPT, Great Lake Symposium on VLSI, ICCAD, ICCD, Annual Conference of IEEE Industrial Electronics (IECON), ISPD, System-on-Chip Conference, IEEE Region 10 Conference (TENCON), and VLSI Design, Automation, and Test. He is currently an Independent Board Director of Genesys Logic, Inc., a member of the board of governors of Taiwan IC Design Society, and a member of the IEEE Circuits and Systems Society, ACM, and ACM/SIGDA.