A Progressive-ILP Based Routing Algorithm for Cross-Referencing Biochips *

Ping-Hung Yuh[1], Sachin Sapatnekar[2], Chia-Lin Yang[1], Yao-Wen Chang[3]
[1]Department of Computer Science and Information Engineering, National Taiwan University
[2]Department of Electrical and Computer Engineering, University of Minnesota
[3]Graduate Institute of Electronics Engineering and Department of Electrical Engineering, National Taiwan University
{r91089, yangc}@csie.ntu.edu.tw; sachin@umn.edu; ywchang@cc.ee.ntu.edu.tw

ABSTRACT

Due to recent advances in microfluidics technology, digital microfluidic biochips and their associated CAD 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, which uses row/column addressing scheme to activate electrodes. We propose the *first* droplet routing algorithm that directly solves the problem of routing in cross-referencing biochips. The main challenge of this type of biochips is the electrode interference which prevents simultaneous movement of multiple droplets. We first present a basic integer linear programming (ILP) formulation to optimally solve the droplet routing problem. Due to its complexity, we also propose a *progressive* ILP scheme to determine the locations of droplets at each time step. Experimental results demonstrate the efficiency and effectiveness of our progressive ILP scheme on a set of practical bioassays.

Categories and Subject Descriptors

B.7.2 [**Integrated Circuits**]: Design Aids

General Terms

Algorithm, Performance, Design

Keywords

Microfluidics, biochip, routing, progressive-ILP

1. INTRODUCTION

Recently, there have been many significant advances in microfluidic technologies [7]. Microfluidic biochips show numerous advantages over conventional assay methods, including portability and sample/reagent volume reduction, and offer a platform for developing clinical and diagnostic applications, such as healthcare of infants and point-of-care diagnostics of disease. This breadth of applicability implies that microfluidic biochips are increasingly used in laboratory procedures in molecular biology.

Lately, a new type of biochips, which are based on digitalizing continuous liquid flow into discrete liquid particles, called *droplets*, have been proposed [5]. These type of biochips are more suitable for large-scale and scalable systems due to their reconfigurability. Each droplet can be independently controlled by the electrohydrodynamic forces generated by electrodes. In this way, droplets movement can be controlled by a system clock. Due to this parallel with digital electronic systems, this type of biochips is referred to as "digital microfluidic biochips."

Typically, a digital microfluidic biochip has a 2D microfluidic array, which consists of a set of electrodes for fundamental operations, such as droplet mixing and transportation. 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 very 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 problem complicates the design process, and therefore, this architecture is only applicable to small-scale biochips [9].

To overcome these limitations, recently a new digital microfluidic biochip architecture has been proposed [3]. This architecture uses a row/column addressing scheme, where a set of electrodes in one row/column is connected to a 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 *cross-referencing biochips*. However, this architecture 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 architecture incurs higher droplet movement complexity than that of direct-addressing biochips. Moreover, the manipulation of more than two droplets causes electrode interference among droplets, which prevents multiple droplets to move at the same time. This performance limitation is a major drawback to 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 needs to be satisfied, and electrode interference patterns 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 a sample spent on a biochip, which is desirable to maintain the bioassay execution integrity.

1.1 Previous Work

Droplet routing is a critical step in biochip design automation. Previous routing approaches mainly focus on direct-addressing biochips [4, 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 re-

*This work was partially supported by the National Science Council of Taiwan under Grant No's. NSC 96-2221-E-002-250, NSC 96-2752-E-002-008-PAE, NSC 96-2628-E-002-248-MY3, NSC 96-2628-E-002-249-MY3, NSC 96-2221-E-002-245, and by the Excellent Research Projects of National Taiwan University, 96R0062-AE00-07.

DAC 2008, June 8–13, 2008, Anaheim, California, USA

sult. Griffith *et al.* [4] 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 to color this undirected graph. Xu *et al.* [8] 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. 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 so far, we believe that there is no routing algorithm that is directly targeted to cross-referencing biochips.

1.2 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 multi-pin 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 min-cost droplet locations at each time step using an ILP. Unlike the progressive ILP scheme proposed in [2], 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:

- 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.
- To tackle the complexity of the basic ILP formulation, we propose the *progressive* ILP routing scheme. We iteratively determine the locations of droplets at each time step by ILP formulation. Therefore, our algorithm can obtain a high-quality solution within reasonable CPU time.
- Unlike previous works that only move a subset of droplets at each time (for example, the algorithm proposed in [8] 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 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 especially important for cross-referencing biochips due to the electrode interference problem.

Experimental results demonstrate the efficiency of our progressive routing scheme compared with the basic ILP formulation. The ILP formulation needs more than five days while the progressive ILP routing scheme needs at most 15.36 seconds for one bioassay. Experimental results also demonstrate the effectiveness of our algorithm compared with previous work. For example, for the protein assay, our algorithm obtains 45.83% smaller maximum droplet transportation time than the network-based method proposed in [10], plus the clique partitioning based algorithm proposed in [8].

The reminder of this paper is organized as follows. Section 2 details routing on cross-referencing biochips and formulates the droplet routing problem. Section 3 presents the basic ILP formulation for droplet routing problem. Section 4 details the progressive ILP routing scheme, while Section 5 shows the experimental results. Finally, concluding remarks are provided in Section 6.

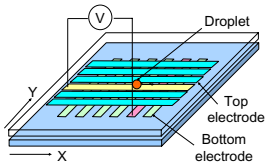

Figure 1: Top view of a cross-referencing biochip.

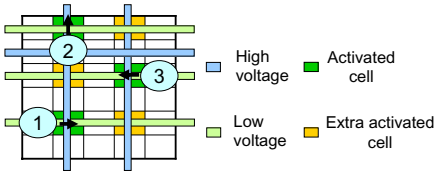

Figure 2: Illustration of electrode interference. Unselected rows/columns are left floating.

2. ROUTING ON CROSS-REFERENCING BIOCHIPS

In this section, we first show the architecture of cross-referencing biochips. Then we detail the unique electrode interference that results in incorrect droplet movement. Finally, we present the problem formulation of the droplet routing problem.

2.1 Cross-Referencing Biochips

Figure 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 (column) in the X-dimension (Y-dimension), and is assigned either a driving or reference voltage. Two sets of electrodes are orthogonally placed, one set each on the top and bottom plates as shown in Figure 1, and each electrode can be set to either a high or a low voltage. A grid point is "addressed" if there is a potential difference between the upper and lower electrode, 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 need only $\hat{W}+\hat{H}$ instead of $\hat{W}\times\hat{H}$ control wires for droplet movement, where $\hat{W}$ ($\hat{H}$) is the width (height) of a biochips, measured in terms of the number of electrodes in each dimension. Therefore, the package and fabrication costs are reduced.

2.2 Electrode Constraint

However, this improvement comes at the cost of reduced flexibility for droplet movement, as compared to direct-addressed 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. To see this, we present an illustration in Figure 2, where our goal is to move three droplets at the same time. The picture shows a set of voltage assignments to the electrodes that are required to move the three droplets to their desired location by activating the corresponding rows and columns and by setting one of the lines to high and the other to low voltage. 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 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 cells causes incorrect droplet movement: for example, the top-right extra cell has no effect on droplet movement since it has no neighboring droplets. Only the extra cells that are around a droplet and its destination cell may cause incorrect droplet movement.

The electrode constraint imposes serious restriction on the movement of multiple droplets. When more than two droplets are moved, we may need to stall some droplets to satisfy the elec-

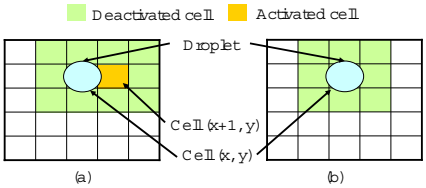

Figure 3: Modeling of electrode constraint when a droplet moves (a) or stays at its original location (b).

trode constraint, which will result in longer droplet transportation times. For example, in Figure 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.

2.3 Problem Formulation

In this paper, we focus on the droplet transportation problem. As stated in [6, 10], the droplet transportation problem can be represented in a 3D space, where the third dimension corresponds to time. At each time step, the problem reduces to working with the droplet movement problem in a 2D plane. We focus the problem formulation on one 2D plane, noting that similar constraints may be added for every other 2D planes. When routing droplets on one 2D plane, we consider the modules (and the surrounding segregation cells) that are active as obstacles.

Besides the electrode constraint, we also need to satisfy the static and dynamic fluidic constraints for correct droplet movement [6]. The static fluidic constraint states that the minimum spacing between two droplets is one cell. Note that once electrode constraint is satisfied, the dynamic fluidic constraint is automatically satisfied, since the neighboring cells of the destination cell and the cell where a droplet is originally at will not be activated. Therefore, we do not need to explicitly handle the dynamic fluidic constraint on cross-referencing biochips. Besides the above constraint, for practical bioassays, we must be able to handle 3-pin nets to represent the fact that two droplets may have to be merged during their transportation for efficient mix operations [6]. Therefore, the droplet routing problem for each 2D 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 2-pin net (one droplet) or a 3-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.*

3. ILP FORMULATION FOR DROPLET ROUTING

In this section, we present the basic ILP formulation for one 2D plane. We show how the ILP optimizes droplet routing and scheduling and voltage assignment on the electrodes, with the consideration of 3-pin nets.

3.1 Basic ILP Formulation

One of the major challenges in the formulation of this problem is in modeling the electrode constraint within an ILP. Figure 3 illustrates the electrode constraint. When a droplet moves from cell (x, y) to cell $(x+1, y)$ at time $t+1$, as shown in Figure 3 (a), all neighboring cells of (x, y) and $(x+1, y)$ must be deactivated, except for cell $(x+1, y)$, which must be activated. Note that in the case that a droplet stays at its original location, cell (x, y) does not necessarily need to be activated and all the neighboring cells of (x, y) must be deactivated, as demonstrated in [3]. Figure 3 (b) shows this case. Therefore, 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 1: Notations used in our ILP formulation.

W/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
$E(x, y)$	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	maximum droplet transportation time
$p_j^i(x, y, t)$	a 0-1 variable to represent that droplet d_j^i locates 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 rule #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, we would need extra variables 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 our basic ILP formulation. The notations used in our ILP formulation are shown in Table 1.

3.1.1 Objective Function

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

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

3.1.2 Constraints

There are total eight constraints in our basic ILP formulation.

1. *Objective function computation:* If a droplet reaches its sink at time $t+1$, then the time 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:

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

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 above requirements can be represented by the following constraints:

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

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

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

3. *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:

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

4. *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:

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

5. *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:

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

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

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

6. *Droplet movement constraint:* A droplet can only move to an adjacent cell, which needs to 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:

$$\sum_{(x',y') \in E(x,y)} p_j^i(x',y',t+1) - p_j^i(x,y,t) \ge 0, \forall d_j^i \in D, (x,y) \in C, 0 \le t < T-1 \quad (11)$$

$$p_j^i(x,y,t+1) + \sum_{(x',y') \in E'(x,y)} p_j^i(x',y',t) - a_{t+1}^{x,y} \le 1, \forall d_j^i \in D, (x,y) \in C, 0 \le t < T-1. \quad (12)$$

Note that an activated cell does not imply that a droplet will move to this cell, as can be seen in the case of the top-right extra activated cell in Figure 2. Therefore, constraint (12) does not state an "if-and-only-if" relation between cell activation and droplet movement.

7. *Electrode constraint:* The three rules explained earlier can be represented as the following constraints:

$$p_j^i(x,y,t) + a_{t+1}^{x',y'} \le 1, \forall d_j^i \in D, (x,y) \in C, (x',y') \in \hat{D}(x,y), 0 \le t < T-1 \quad (13)$$

$$9p_j^i(x,y,t) + \sum_{(x',y') \in \hat{E}(x,y)} a_t^{x',y'} \le 9, \forall d_j^i \in D, (x,y) \in C, 0 \le t < T \quad (14)$$

$$6p_j^i(x,y,t) + \sum_{(x',y' \in E(x,y)} a_{t+1}^{x',y'} \le 7, \forall d_j^i \in D, (x,y) \in C, 0 \le t < T-1, \quad (15)$$

where constraint (13) represents rule #1, constraint (14) represents rule #2, and constraint (15) represents rule #3.

8. *3-pin nets:* We use the following three constraints to handle the 3-pin nets:

$$\sum_{(x,y)} (p_0^i(x,y,t) - p_1^i(x,y,t)) = 0 \leftrightarrow m_t^i = 1, \forall n_k \in N', (x,y) \in C, 1 \le t < T \quad (16)$$

$$\sum_{t=1}^{T-1} m_t^i \ge 1, \forall n_k \in N' \quad (17)$$

$$(t+1) \times \sum_{t=1}^{T-2} (m_{t+1}^i - m_t^i - p_0^i(\hat{t}_x^i, \hat{t}_y^i, t+1) + p_0^i(\hat{t}_x^i, \hat{t}_y^i, t)) \le -1. \quad (18)$$

Algorithm: Progressive ILP routing
Let T_l be the maximum Manhattan distance of all droplets from their sources to sinks;
begin
1 **while** not all droplets reach their sinks
2 Compute_droplet_movement_cost();
3 Construct_ILP();
4 Solve_ILP();
5 Update_droplet_position();
6 **if** it is not possible to route all droplets within T_l
7 T_l += $\alpha\times$ maximum Manhattan distance;
end

Figure 4: Overview of progressive ILP method.

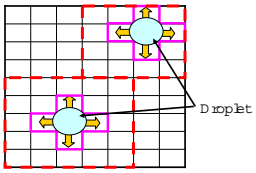

Figure 5: Illustration of the locality of droplet movement and electrode constraint when locations of droplets are known.

Constraint (16) is used to determine whether two droplets are merged; i.e., in the same physical location. Constraint (17) is used to guarantee that two droplets must be merged during their transportation by restricting that their physical location must be the same for at leat one time step. Constraint (18) states that these two droplets must be merged before reaching their sink. Note that once these two droplets are merged, they will always move together by the droplet movement constraint. Therefore, these two droplets are not splitted once they are merged.

4. PROGRESSIVE ILP ROUTING SCHEME

Although the basic ILP formulation presented in the previous section can solve the droplet transportation problem, it may incur high run times. 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 subsections, we first overview our progressive ILP routing scheme, and then present the ILP formulation. Then, we present how to handle 3-pin nets and how to determine droplet movement cost.

4.1 Progressive Routing Algorithm Overview

Figure 4 shows the overview of our 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 cannot be routed within T_l, 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.

4.2 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. We can further reduce the problem size by observing that the electrode and static fluidic constraints and droplet movement have only local effects when the locations of the droplets are known. Figure 5 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 neighbor-

ing cells and their adjacent cells, the progressive ILP does need not 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 droplets nearby is activated. Therefore, we do not activate this cell in our 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 Figure 5.

Therefore, we only need variables $a_{t+1}^{x,y}$ 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 1.

4.2.1 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 equation:

$$Minimize \quad \sum_{d_j^i \in D} \sum_{(x,y) \in E(x_t^{i,j i}, y_t^{i,j})} w_j^i(x,y) p_j^i(x,y,t+1), \quad (19)$$

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

4.2.2 Constraints

There are five constraints in the progressive ILP formulation.

1. *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:

$$p_j^i(x,y,t+1) \leftrightarrow a_{t+1}^{x,y}, \forall d_j^i \in D, \forall (x,y) \in E'(x_t^{i,j}, y_t^{i,j}). \quad (20)$$

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 a "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 can use the following constraints to represent this situation:

$$p_j^i(x_t^{i,j}, y_t^{i,j}, t+1) + \sum_{(x,y) \in E'(x_t^{i,j}, y_t^{i,j})} a_{t+1}^{x,y} \geq 1, \forall d_j^i \in D \quad (21)$$

$$4p_j^i(x_t^{i,j}, y_t^{i,j}, t+1) + \sum_{(x,y) \in E'(x_t^{i,j}, y_t^{i,j})} a_{t+1}^{x,y} \leq 4, \forall d_j^i \in D. \quad (22)$$

2. *Electrode constraint:* Rule #1, which states that the diagonal cells cannot be activated, can be represented by the following constraints:

$$L_x^c(t+1) + H_y^r(t+1) \leq 1, \forall (x,y) \in \hat{D}(x_t^{i,j}, y_t^{i,j}) \quad (23)$$

$$H_x^c(t+1) + L_y^r(t+1) \leq 1, \forall (x,y) \in \hat{D}(x_t^{i,j}, y_t^{i,j}). \quad (24)$$

The following two constraints, similar to constraint (14), are used for rule #2:

$$\text{if } p_j^i(x,y,t+1) = 1 \rightarrow$$
$$L_{x'}^c(t+1) + H_{y'}^r(t+1) \leq 1 \text{ or } H_{x'}^c(t+1) + L_{y'}^r(t+1) \leq 1$$
$$\forall d_j^i \in D, \forall (x,y) \in E'_x(x_t^{i,j}, y_t^{i,j}), \forall (x',y') \in E'_y(x,y) \quad (25)$$

$$\text{if } p_j^i(x,y,t+1) = 1 \rightarrow$$
$$L_{x'}^c(t+1) + H_{y'}^r(t+1) \leq 1 \text{ or } H_{x'}^c(t+1) + L_{y'}^r(t+1) \leq 1$$
$$\forall d_j^i \in D, \forall (x,y) \in E'_y(x_t^{i,j}, y_t^{i,j}), \forall (x',y') \in E'_x(x,y). \quad (26)$$

As shown in Figure 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 (25) is used to represent these two cases, and constraint (26) is used to represent the case when d_j^i moves to cell $(x_t^{i,j}, y_t^{i,j}+1)$ or $(x_t^{i,j}, y_t^{i,j}-1)$. Finally, for rule #3, we impose the following constraint:

$$\sum_{(x',y') \in E(x_t^{i,j}, y_t^{i,j})} a_{t+1}^{x',y'} \leq 1, \forall d_j^i \in D. \quad (27)$$

3. *Static fluidic constraint:* Since the locations of the droplets are known, we can 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:

$$\text{if more than one droplet are around cell } (x,y)$$
$$a_{t+1}^{x,y} = 0, \forall (x,y) \in E'(x_t^{i,j}, y_t^{i,j}), \forall d_j^i \in D. \quad (28)$$

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

$$p_j^i(x_t^{i,j}, y_t^{i,j}, t) - p_j^i(x_{t+1}^{i,j}, y_{t+1}^{i,j}, t+1) \leq 0, \forall d_j^i \in D, \quad (29)$$

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

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

4.3 Droplet Movement Cost

A key issue in the progressive ILP routing scheme is the determination of the droplet movement cost. Since our goal is to minimize the maximum droplet transportation time, our objective is to move a droplet toward its sink at each time step. Furthermore, we need to avoid congestion among droplets; otherwise, we may either have to detour or stall some droplets for certain time, to satisfy both the fluidic and electrode constraints. Therefore, the droplet transportation time is potentially increased. In this subsection, 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:

$$w_j^i(x,y) = \beta\ r_j^i(x,y,\hat{t}_x^i, \hat{t}_y^i) + \gamma\ g_j^i(x,y,\hat{t}_x^i, \hat{t}_y^i), \quad (30)$$

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 this paper, we empirically set $\beta = 15$ and $\gamma = 0.1$. The routing cost is the real distance computed by the maze routing algorithm from cell (x,y) to the sink of d_j^i over the area of a biochips (for normalization). We consider all other droplets except d_j^i as 3×3 obstacles when performing the maze routing algorithm due to the static fluidic constraint.

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 use the concept of the idle interval presented in [10] for congestion cost computation. 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, $T^m(d_j^i)$ represents the earliest time when d_j^i reaches (x,y) from its source and $T^M(d_j^i)$ represents the latest time when d_j^i can stay at (x,y) without violating the time limitation. If we set T_l as 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 $T^m(d_j^i)$ and $T^m(d_j^i)$ for a subproblem at time t can be computed by the following two equations:

$$T^m(d_j^i) = t + d^m(x,y,\hat{t}_t^x, \hat{t}_t^y) \quad (31)$$

$$T^M(d_j^i) = T_l - d^m(x,y,\hat{t}_t^x, \hat{t}_t^y), \quad (32)$$

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 all cells' congestion information within the bounding box defined by cell (x,y)

Table 2: 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
*in-vitro*_1	24889	138	176580	388	> 7200 min.	2.55 sec.
*in-vitro*_2	18061	134	126983	398	> 7200 min.	2.53 sec.
Protein_1	49561	134	350645	357	> 7200 min.	15.36 sec.
Protein_2	22238	132	252989	385	> 7200 min.	6.70 sec.

Table 3: Routing result of the two bioassays.

Circuit	[10] + [8]		[10] + [4]		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.)
*in-vitro*_1	40 / 16.72	0.05	47 / 20.18	0.06	24 / 13.09	2.55
*in-vitro*_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

and its sink. Note that we use the real bounding box computed by the maze routing algorithm. 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 equation:

$$g_j^i(x, y, \hat{t}_x^i, \hat{t}_y^i) = \sum_{(x',y') \in b_j^i(x,y)} \frac{|I_j^i(x', y')|}{\sum_{d_{j'}^{i'} \in D} |I_{j'}^{i'}(x, y)|}, \quad (33)$$

where $b_j^i(x, y)$ is the bounding box obtained by the maze routing 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, our router will not tend to move a droplet to cell (x, y).

4.4 Handling 3-pin Nets

For a 3-pin net n_i, we need to merge these two droplets during their transportation. To guarantee that these two droplets are merged, we first route these two droplets toward each other. After 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 merged.

4.5 Implementation

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 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.

5. EXPERIMENTAL RESULTS

Our progressive ILP algorithm was implemented in the C++ language, and GLPK [1] was used as our ILP solver. For the purpose of comparison, we also implemented the clique partitioning based [8] and the graph coloring based [4] algorithms. All of the above algorithms and ILP formulations were executed on a 1.2 GHz SUN Blade-2000 machine with 8GB memory. We evaluated our routing algorithm on the two practical bioassays used in the previous work [10]: the *in-vitro* diagnostics and the colorimetric protein assay.

We performed two experiments to verify the efficiency and effectiveness of our progressive ILP algorithm. In the first experiment, we compared the basic ILP and progressive ILP algorithms. We generated both the basic and progressive ILP formulations for each 2D plane. The experimental results are listed in Table 2. Columns 2 and 3 (Columns 4 and 5) list the maximum number of variables (constraints) of all problem instances. For the basic ILP, the problem instance is one 2D plane, and for the progressive ILP, the problem instance is one time step. The results show that the basic ILP needs at least five days to solve all 2D planes of one benchmark, which is not feasible for this problem; in contrast, our progressive ILP algorithm needs at most 15.36 sec due to the significantly smaller problem size. For example, as shown in Table 2, for the protein_1 benchmark, we can reduce the number of variables (constraints) by 99.72% (99.89%) for the largest problem instance. This result demonstrates the efficiency of our progressive ILP formulation.

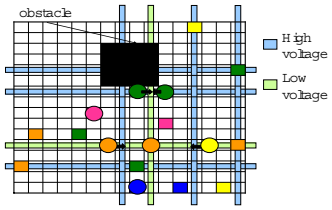

Figure 6: The result of the *in-vitro*_1 benchmark.

Our progressive ILP formulation can obtain a near-optimal solution with much less CPU time compared with the basic ILP formulation. For example, for one 2D plane of the protein_1 benchmark with four droplets, the progressive ILP formulation obtains a solution of 21 cycles in 0.40 sec while the basic ILP formulation obtains a 20-cycle solution in 19632 sec. Since it is not feasible to solve the basic ILP for practical bioassays, we are not able to compare the solution quality of these two algorithms directly. Nevertheless, we observe that the difference in the solution quality for the two algorithms is at most 3 cycles for 41 2D planes completed by the basic ILP formulation, and the average solution difference is only 1.68%. The result shows that the progressive ILP algorithm is close to the optimal solution.

In the second experiment, we verified the quality of our progressive ILP routing scheme. Since previous works require a direct-addressing routing solution, for 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 3 shows the experimental result. We report the maximum and average T_l of all 2D planes and the CPU time to route all 2D planes. The CPU times for [10]+[8] and [10]+[4] are the total CPU times to obtain the cross-referencing routing solution. As shown in this table, our progressive ILP routing algorithm can obtain much smaller droplet transportation times (in cycles) than the previous approaches, under reasonable CPU times. This result demonstrates that our algorithm is very effective for droplet routing on cross-referencing biochips. Figure 6 shows the routing result of one 2D plane of the *in-vitro*_1 benchmark at cycle 8.

6. 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. Experimental results have shown the efficiency and effectiveness of our algorithm.

7. REFERENCES

[1] http://www.gnu.org/software/glpk/.

[2] M. Cho and D. Z. Pan. Boxrouter: a new global router based on box expansion and progressive ilp. In *DAC*, pages 373–378, July 2006.

[3] J. Gong, S.-K. Fan, and C.-J. Kim. Portable digital microfluidics platform with active but disposable lab-on-chip. In *MEMS*, pages 355–358, 2004.

[4] E. J. Griffith, S. Akella, and M. K. Goldberg. Performance characterization of a reconfigurable planar-array digital microfluidic system. *TCAD*, 25(2):435–357, 2006.

[5] F. Su, K. Chakrabarty, and R. B. Fair. Microfluidic-based biochips: Technology issues, implementation platforms, and design-automation challenges. *TCAD*, 25(4):211–223, 2006.

[6] F. Su, W. Hwang, and K. Chakrabarty. Droplet routing in the synthesis of digital microfluidic biochips. In *DATE*, pages 323–328, 2006.

[7] E. Verpoorte and N. F. D. Rooij. Microfluidics meets MEMS. *Proceedings of IEEE*, 1(6):930–953, June 2003.

[8] T. Xu and K. Chakrabarty. A cross-referencing-based droplet manipulation method for high-throughput and pin-constrained digital microfluidic arrays. In *DATE*, pages 552–557, Apr. 2007.

[9] T. Xu and K. Chakrabarty. Droplet-trace-based array partitioning and a pin assignment algorithm for the automated design of digital microfluidic biochips. In *CODES+ISSS*, pages 112–117, Oct. 2007.

[10] P.-H. Yuh, C.-L. Yang, and Y.-W. Chang. BioRoute: A network-flow based routing algorithm for digital microfluidic biochips. In *ICCAD*, Nov. 2007.