Layout-Aware Mixture Preparation of Biochemical Fluids on Application-Specific Digital Microfluidic Biochips

SUDIP ROY, Indian Institute of Technology Roorkee
PARTHA P. CHAKRABARTI and SRIJAN KUMAR, Indian Institute of Technology Kharagpur
KRISHNENDU CHAKRABARTY, Duke University
BHARGAB B. BHATTACHARYA, Indian Statistical Institute

The recent proliferation of digital microfluidic (DMF) biochips has enabled rapid on-chip implementation of many biochemical laboratory assays or protocols. Sample preprocessing, which includes dilution and mixing of reagents, plays an important role in the preparation of assays. The automation of sample preparation on a digital microfluidic platform often mandates the execution of a mixing algorithm, which determines a sequence of droplet mix-split steps (usually represented as a mixing graph). However, the overall cost and performance of on-chip mixture preparation not only depends on the mixing graph but also on the resource allocation and scheduling strategy, for instance, the placement of boundary reservoirs or dispensers, mixer modules, storage units, and physical design of droplet-routing pathways. In this article, we first present a new mixing algorithm based on a number-partitioning technique that determines a layout-aware mixing tree corresponding to a given target ratio of a number of fluids. The mixing graph produced by the proposed method can be implemented on a chip with a fewer number of crossovers among droplet-routing paths as well as with a reduced reservoir-to-mixer transportation distance. Second, we propose a routing-aware resource-allocation scheme that can be used to improve the performance of a given mixing algorithm on a chip layout. The design methodology is evaluated on various test cases to demonstrate its effectiveness in mixture preparation with the help of two representative mixing algorithms. Simulation results show that on average, the proposed scheme can reduce the number of crossovers among droplet-routing paths by 89.7% when used in conjunction with the new mixing algorithm, and by 75.4% when an earlier algorithm [Thies et al. 2008] is used.

Categories and Subject Descriptors: B.7.1 and B.7.2 [**Integrated Circuits**]: Design Aids - Layout, Types and Design Styles - Algorithms Implemented in Hardware; B.8.1 [**Performance and Reliability**]: Reliability, Testing, and Fault-Tolerance; J.3 [**Life and Medical Sciences**]: Biology and Genetics, Health

General Terms: Algorithms, Design

Additional Key Words and Phrases: Biochips, design automation, digital microfluidics, dilution and mixing, sample preparation

---

Preliminary versions of this article appeared in *Proceedings of the International Conference on VLSI Design (VLSID'11),* [Roy et al. 2011b] and *Proceedings of the IEEE International Symposium on VLSI (ISVLSI'13),* [Roy et al. 2013]. S. Roy was supported in part by Indian Institute of Technology Kharagpur and by Microsoft Research India PhD Fellowship (2010–2014). P. P. Chakrabarti was supported in part by the J. C. Bose Fellowship from Department of Science and Technology (DST), Government of India. K. Chakrabarty was supported in part by the US National Science Foundation under grant CNS-1135853. B. B. Bhattacharya was supported by a special grant to Nanotechnology Research Triangle from Indian Statistical Institute, Kolkata. S. Kumar is currently affiliated with the University of Maryland.
Authors' addresses: S. Roy, Department of Computer Science and Engineering, Indian Institute of Technology, Roorkee-247667, India; email: sudiproy.fcs@iitr.ac.in; P. P. Chakrabarti, Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur-721302, India; email: ppchak@cse.iitkgp.ernet.in; S. Kumar, University of Maryland, College Park, MD; email: srijan@cs.umd.edu; K. Chakrabarty, Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708; email: krish@ee.duke.edu; B. B. Bhattacharya, Advanced Computing and Microelectronics Unit, Indian Statistical Institute, 203, B.T. Road, Kolkata-700108, India; email: bhargab@isical.ac.in.

DOI: http://dx.doi.org/10.1145/2714562

ACM Reference Format:
Sudip Roy, Partha P. Chakrabarti, Srijan Kumar, Krishnendu Chakrabarty, and Bhargab B. Bhattacharya. 2015. Layout-Aware Mixture preparation of biochemical fluids on application-specific digital microfluidic biochips. ACM Trans. Des. Autom. Electron. Syst. 20, 3, Article 45 (June 2015), 34 pages.
DOI: http://dx.doi.org/10.1145/2714562

1. INTRODUCTION

The rapid escalation in healthcare cost for cardiovascular diseases, cancer, diabetes, and global HIV/AIDS crisis has fueled a new field of interdisciplinary research centered around "lab-on-a-chip (LoC)" [Sista et al. 2008; Tan et al. 2008]. A typical LoC (or a biochip) implements one or more biochemical laboratory protocols or assays on a single microfluidic chip that is a few square centimeters in size. In general, digital microfluidic (DMF) biochips use electrical actuation to manipulate (i.e., dispense, navigate, mix/split, wash, and detect) discrete droplets of nano- or picoliter volume of biochemical fluids on a two-dimensional electrode array [Chakrabarty and Xu 2010]. Recently, DMF biochips have gained wide acceptance for developing LoC applications because of their flexibility and programmability [Abdelgawad and Wheeler 2009; Chatterjee et al. 2006; Miller and Wheeler 2009].

A real-life biochemical laboratory protocol often requires mixing of several reagent fluids. Sample preparation and analyte identification steps in such bioprotocols involve mixing for mixture preparation, that is, various reactant fluids are to be mixed in a certain volumetric ratio to obtain the desired mixture. For instance, in the polymerase chain reaction (PCR) used for DNA amplification, a master-mixture of seven fluids (reactant buffer, dNTPs, forward primer, reverse primer, DNA template, optimase and water) is required with a volumetric ratio {10%:8%:0.8%:0.8%:1%:1%:78.4%}.[1]

Recently, several design automation methods have been proposed for architectural synthesis (i.e., operation scheduling, resource selection and binding) and physical design (i.e., module placement and droplet routing) of DMF biochips [Chakrabarty and Xu 2010; Su and Chakrabarty 2008; Zhao and Chakrabarty 2009]. On-chip mixing of several reactant fluids with a specified ratio of concentration factors (CFs) is a challenging problem in automating biochemical laboratory protocols on a DMF biochip. Furthermore, many CAD algorithms and schemes for automated mixture-preparation have been reported [Dinh et al. 2014; Hsieh et al. 2012a; Huang et al. 2012; Kumar et al. 2013; Mitra et al. 2012; Roy et al. 2011b, 2010; Thies et al. 2008]. Hsieh et al. [2012b, 2014] presented a design methodology with dynamic error recovery for architectural and layout synthesis of a sample preparation biochip. However, it is observed that the performance of a mixing algorithm depends on resource-allocation, for instance, the placement of mixer modules, storage units, boundary reservoirs or dispensers on the chip floor. Moreover, the complexity of the droplet routing depends on the mix-split steps to be executed for on-chip.

In this article, we present a layout-aware mixing algorithm for efficient and automated mixture-preparation of three or more fluids on a digital microfluidic (DMF) biochip. Unlike some previously proposed methods [Kumar et al. 2013; Liu et al. 2013; Thies et al. 2008], which rely on a bottom-up construction of a mixing graph, this algorithm uses a top-down approach based on fractional decomposition aided with number partitioning. The method also leads to a relative placement of resources (mixers, fluid reservoirs) on-chip that facilitates the implementation of the mixing algorithm. For a desired ratio, the proposed mixing algorithm determines a mixing tree with longer sub-sequences of mixing steps with a small number of distinct fluids (called "tall dilution subtrees"). This property helps in allocating the boundary reservoirs to the reactant fluids in the proximity of the assigned mixers, which in turn, reduces the total

[1]PCR Master Mix Calculator; http://www.mutationdiscovery.com.

droplet-transportation time for moving droplets from the fluid reservoirs to the designated on-chip mixers.

In addition to conventional DMF biochips, the proposed algorithms are also applicable to the more advanced class of Active-Matrix (AM) based microfluidic architecture, where the individual electrode resembles a tiny pixel [Hadwen et al. 2012]. Such a device not only provides a versatile control over droplet volumes and shapes, but also allows the droplets to move along a random trajectory. Since our algorithms are independent of electrode architecture and are based solely on the relative placement of reservoirs and mixers, they translate seamlessly to this new architecture as well.

In a biochip, the mixing modules, storage units (some additional electrodes used to store intermediate droplets), and the fluid reservoirs at the chip boundary are regarded as *resources*. We model the collection of all droplet transportation routes from fluid reservoirs to mixers with a bipartite graph. In this graph, the boundary dispensers and on-chip mixers are represented as two linearly ordered disjoint sets of nodes; an edge, which represents a reservoir-to-mixer droplet routing path, is drawn as a straight-line, when the graph is embedded on a plane. Next, the total number of edge intersections (i.e., crossing among droplet pathways) is minimized by reallocating the fluid reservoirs. The proposed routing-aware technique leads to a suitable allocation of reservoirs, and placement of mixers on a biochip, which reduce the path-crossovers as well as the total length of droplet-transportation routes. The scheme also facilitates the scheduling of droplet-routing paths as a fewer number of stalls (in terms of clock cycles), are needed in order to avoid unintended mixing of two crossing droplets. It can also be used to improve the performance of other mixing algorithms. Simulation results show that on the average (computed by varying the number of available on-chip mixers), the proposed placement scheme can reduce the number of crossovers among droplet-routing paths by 89.7% for the new top-down decomposition algorithm, and by 75.4% for an earlier algorithm [Thies et al. 2008].

The remainder of the article is organized as follows. Basic preliminaries and prior work on automated dilution and mixing algorithms are discussed in Section 2. Section 3 describes the motivation and problem formulation for mixture-preparation with several reactant fluids. An efficient mixing algorithm for layout-aware mixture-preparation is introduced in Section 4. In Section 5, we present a routing-aware resource-allocation scheme that can used to improve the performance of a mixing algorithm. Section 6 presents the integration of the routing-aware resource-allocation scheme with mixing algorithms. In Section 7, we discuss the resource-allocation problem in the context of multiple target ratios. Finally, conclusions are drawn in Section 8.

2. AUTOMATED MIXTURE PREPARATION OF BIOCHEMICAL FLUIDS

2.1. Preliminaries

A DMF biochip operates with discrete droplets on a uniform 2D-array of equi-sized electrodes, hence the volume of a droplet is usually an integral multiple of that of a single droplet. Various $(k : \ell)$ mixing models may be considered, where a k-unit volume of one fluid is mixed with an ℓ-unit volume of another fluid to produce a $(k + \ell)$-unit volume of the resultant mixture in a single mixing operation. Three such mixing models are: (i) $k = \ell = 1$, (ii) $k = \ell \neq 1$, and (iii) $k \neq \ell$, where k, ℓ are positive integers. Note that the first one, that is, the $(1 : 1)$ mixing model is the easiest to implement.

While specifying a sample, the term *concentration factor* (CF) is used to quantify the amount of raw fluid in the prepared sample [Herold and Rasooly 2009]. The *concentration factor* is defined as the ratio of initial volume of the sample to the final volume of the prepared mixture (i.e., the inverse of the dilution factor, $CF = \frac{1}{DF}$). The fluid with which the sample is mixed for dilution is called the diluent or buffer solution, for instance, water or other liquid, which is neutral to the sample. The pure sample fluid

is considered as having 100% concentration, that is, $CF = 1$. Hence, the buffer solution can be viewed as having 0% concentration with respect to sample, that is, $CF = 0$. Dilution is commonly used in biological studies to create a variety of solution concentrations of a fluid with the help of a buffer solution [Herold and Rasooly 2009]. Hence, dilution is a special case of mixing of two input fluids, one of which is a buffer solution. In general, dilution of a sample fluid with $CF = C_1$ can be achieved by mixing it with another sample of same fluid with $CF = C_2$, if $C_2 < C_1$. The CF of the resultant fluid lies between C_1 and C_2 because, if the samples with $CF = C_1$ and C_2 are mixed in a volumetric ratio of $k : \ell$, then the resulting $(k + \ell)$-unit volume fluid has a $CF = C_r = \frac{k.C_1+\ell.C_2}{k+\ell}$. After a balanced splitting of the resultant volume, two $(\frac{k+\ell}{2})$-unit volume resultant droplets are produced.

Mixing is used to prepare a mixture (solution) of three or more different fluids with a desired ratio of their concentrations. In this work, we assume the $(1 : 1)$ mixing model for mixing, that is, every *mix-split cycle* consists of a mix operation between two unit-volume fluid droplets and followed by a balanced split operation of the mixed fluid. Thus, if a fluid droplet with $CF = C_1$ is mixed with its another droplet of the same reagent, with $CF = C_2$ (where $0 \leq C_2 \leq C_1$), then the final CF of the each of the resulting two droplets becomes $\frac{C_1+C_2}{2}$, assuming that the same diluent is used in preparing the two droplets. One mix operation and a subsequent split are together called as one *mix-split operation, cycle, or step*. Since after one $(1 : 1)$ mix-split operation the resulting CF becomes the mean value of those of the two source droplets, for the sake of convenience, the desired target CF (C_t) of a sample is expressed as a d-bit binary fraction (by truncating it off beyond d^{th}-bit), when an accuracy level of d is desired.

In all subsequent discussions, we will use the following notation to indicate various parameters: N denoting the number of input fluids; d for accuracy level; T_{ms}, the total number of $(1 : 1)$ mix-split cycles needed in during dilution/mixing; W for the total number of waste droplets produced; M_{lb} denoting the minimum number of mixers required for the earliest possible completion of dilution/mixing steps.

2.2. Prior Work: Dilution and Mixing Algorithms

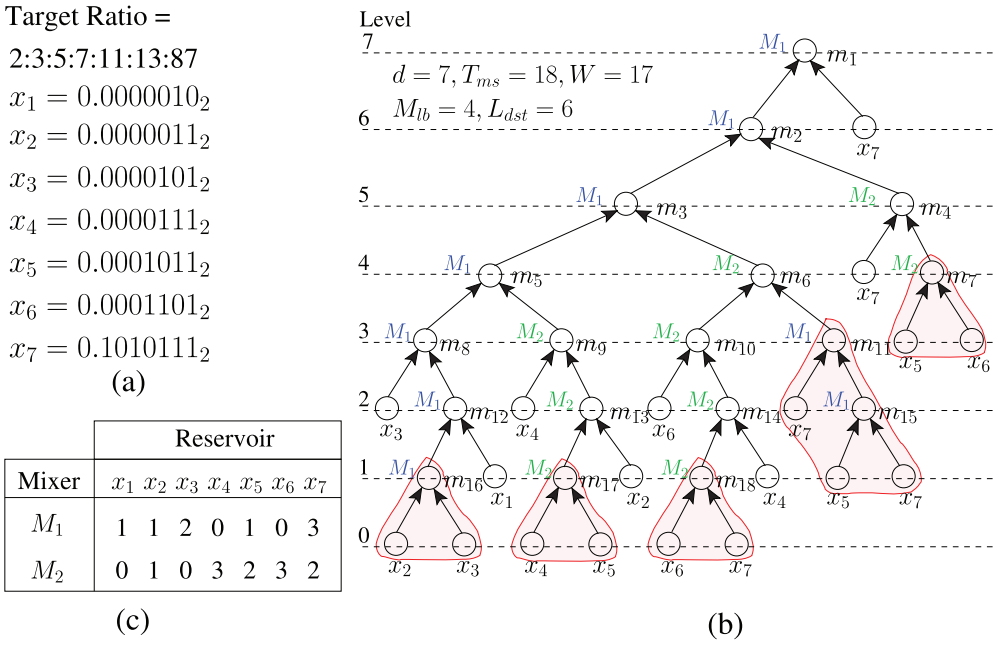
A number of algorithms have appeared in the literature for performing dilution or mixing of fluids on a DMF platform [Ren et al. 2003], *GAG* [Griffith et al. 2006], *twoWayMix* [Thies et al. 2008], *DMRW* [Roy et al. 2010], *IDMA* [Roy et al. 2011a], *MD* [Bhattacharjee et al. 2012], *REMIA* [Huang et al. 2012], *GORMA* [Chiang et al. 2013], *WARA* [Huang et al. 2013], *MinMix* [Thies et al. 2008], *RMA* [Roy et al. 2011b], *RSM* [Hsieh et al. 2012a], *MTCS* [Kumar et al. 2013], *CoDOS* [Liu et al. 2013] and *NFSP* [Dinh et al. 2014]. A majority of them [Bhattacharjee et al. 2012, 2014; Chiang et al. 2013; Dinh et al. 2014; Griffith et al. 2006; Huang et al. 2012; Mitra et al. 2014, 2012; Ren et al. 2003; Roy et al. 2010, 2011a, 2014a, 2014b, 2014c; Thies et al. 2008] deal with with the task of diluting a fluid to a certain CF by mixing it with a buffer. Note that the dilution process needs only two input fluids to prepare a target solution, and up to two mixers are sufficient to complete the task [Roy et al. 2010; Thies et al. 2008]. On the other hand, mixture-preparation needs three or more input fluids to be mixed in a certain ratio, and a number of mixers are allocated on-chip to speed up the task. Since several fluids are often required as input, the performance of a mixing algorithm strongly depends on resource placement and allocation, for instance, the placement of input reservoirs or dispensers, mixer modules, and on-chip storage units. It also depends on the number of mixing modules available for concurrent operation, the type of mixers, scheduling of mix-split steps, and the physical layout of droplet-routing paths. Implementation of a mixing algorithm with a limited number of on-chip mixer modules and storage units is also a challenging task.

Thies et al. [2008] proposed a binary bit-scanning based mixing algorithm, called *MinMix*, to determine a (1 : 1) mixing tree for a desired concentration ratio of N different fluids. In this method, the *CF* values of each of the constituents in the target ratio is expressed as a d-bit binary fraction. Next, these N d-bit binary representations are bit-wise scanned from right-to-left to construct, in a bottom-up fashion, a mixing tree of height d. Another mixing algorithm, namely *RSM* [Hsieh et al. 2012a] determines the mixing tree/graph for one/more target ratio(s) of input fluids using a factorization based technique. Kumar et al. [2013] proposed another mixing algorithm called *MTCS*, which can reduce the total number of mix-split steps and reactant usage by reusing intermediate waste droplets. Recently, Liu et al. [2013] described a mixing algorithm called *CoDOS* to generate the mixing trees with common dilution operation sharing in order to recycle the unused intermediate droplets. However, none of these methods [Hsieh et al. 2012a; Kumar et al. 2013; Liu et al. 2013; Thies et al. 2008] have addressed the problem of considering the placement geometry of reservoirs and mixers that determines the droplet-routing overhead during mixture preparation.

3. MOTIVATION AND PROBLEM FORMULATION

Given a target ratio, the mixing process, that is, the sequence of (1 : 1) mix-split steps is represented by a task graph known as dilution or mixing graph [Roy et al. 2014c; Thies et al. 2008]. In a (1 : 1) dilution or mixing tree, each leaf node corresponds to a unit-volume droplet of an input fluid; an internal (or nonleaf) node denotes the resultant mixture obtained by applying a (1 : 1) mix-split step on the two droplets corresponding to its two children. Only one unit-volume droplet of the resultant mixture, is used in the next mix operation denoted by its parent node. The remaining droplet is either discarded as waste or stored for reuse in a subsequent mix operation, depending upon the underlying mixing algorithm [Griffith et al. 2006; Hsieh et al. 2012a; Huang et al. 2012, 2013; Kumar et al. 2013; Liu et al. 2013; Roy et al. 2010, 2011a; Thies et al. 2008]. Such a discarded droplet is referred to as a waste droplet. The mix-split operation denoted by an internal node can be executed only when each of its two children is either already processed or is a leaf node. To represent this sequential dependence, every edge connecting a node (except the root node) to its parent is directed towards the latter.

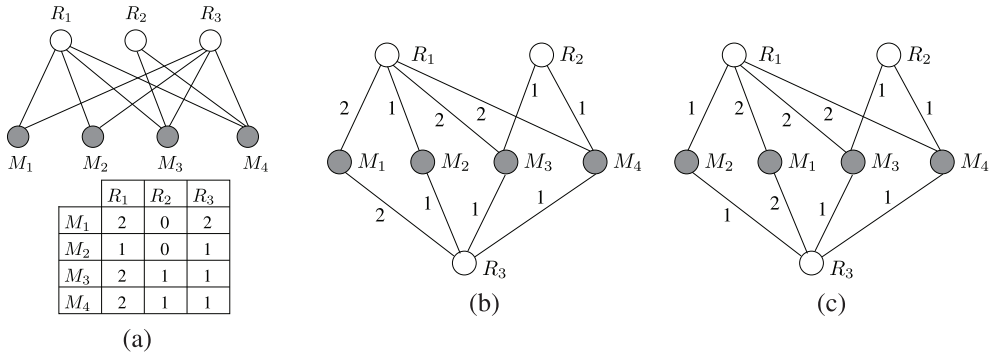
For an example target ratio 2 : 3 : 5 : 7 : 11 : 13 : 87 of seven reactant fluids, with an accuracy level of $d = 7$ in the target *CF*, the binary representations used to construct the mixing tree of Figure 1(b) are shown in Figure 1(a). The total number (1 : 1) mix-split steps (T_{ms}) in this mixing tree is 18. In this figure, each mix-split step is marked as m_i with $i = 1$ to 18, where the final mix-split step required to obtain the target droplets corresponds to the root of the mixing tree. After every mix-split step barring the one denoted by the root node, only one out of two droplets is used at a subsequent step, and hence, the total number of waste droplets (W) produced in the process is 17. Note that the minimum number of mixers required for the earliest completion of the execution of this mixing tree (M_{lb}) [Luo and Akella 2011] is four. However, the desired mixture can also be prepared at a slower speed with a limited number of on-chip mixers. If the number of available mixers is less than M_{lb}, on-chip storage units (additional electrodes) are needed to store intermediate droplets. We assume that only two mixers M_1 and M_2 are available and a corresponding scheduling is shown in Figure 1(b). The table shown in Figure 1(c) depicts the total reservoir-to-mixer transportation workload. For example, the entry "2" in row-1 and column-3 indicates that two droplets of input fluid x_3 are needed by mixer M_1 during the mixing process. This table can be easily constructed from the mixing tree.



Mixer	Reservoir						
	x_1	x_2	x_3	x_4	x_5	x_6	x_7
M_1	1	1	2	0	1	0	3
M_2	0	1	0	3	2	3	2

(c)

Fig. 1. For an example ratio 2 : 3 : 5 : 7 : 11 : 13 : 87, (a) bit-representations of the ratio, (b) mixing tree obtained by *MinMix*, and (c) a table containing reservoir-to-mixer droplet transportation workload.



	R_1	R_2	R_3
M_1	2	0	2
M_2	1	0	1
M_3	2	1	1
M_4	2	1	1

Fig. 2. For a DMF biochip layout with three reservoirs and four mixers, (a) initial embedding of the bipartite graph, (b) reembedding of the bipartite graph, and (b) final embedding of the bipartite graph after reallocation of reservoirs.

3.1. Impact of Reservoir/Mixer Allocation on Droplet Routing

Given a mixing graph and a number of boundary dispensers, the workload pattern can be conveniently represented by a bipartite graph. For simplicity, we assume that all boundary reservoirs and mixers are arranged linearly. The following example illustrates how the impact of reservoir allocation on droplet routing can be perceived from such a graph representation. Consider a mixture preparation task represented as a bipartite graph (see Figure 2(a)), in which three reactant fluids x_1, x_2 and x_3 are loaded into three physical resources (reservoirs) R_1, R_2 and R_3, respectively of a biochip that has deployed four virtual resources (mixers) M_1, M_2, M_3 and M_4. The

linear ordering of the nodes in the embedding of the graph indicates the relative placement of dispensers and mixers. An edge represents a required droplet transportation from a reservoir to a mixer. For example, the droplets from R_1 and R_3 should arrive at mixer M_1; in the bipartite graph, this fact is reflected by putting two edges from M_1 to R_1 and R_3. The transportation workload can be obtained as before from a mixing tree following a scheduling of mixers; this can be indicated by attaching the corresponding integer weight $w(e_i)$ to an edge e_i. Note that in the initial embedding of the bipartite graph (Figure 2(a)), the total number of edge-crossings (X) is 12. Each cross-point may mandate a stall during droplet routing if the two corresponding droplets arrive at the cross-point simultaneously. Thus, the maximum number of stalls that may be needed at a cross-point of two edges e_i and e_j can be estimated as $min(w(e_i), w(e_j)$. Therefore, considering the workload, we can compute the weighted edge-crossing number as $X_w = \sum min(w(e_i), w(e_j))$, for every cross-point (e_i, e_j).

In the initial embedding of the bipartite graph (Figure 2(a)) the total number of weighted edge-crossings (X_w) is 14. A reembedding of the bipartite graph is shown in Figure 2(b), in which the weighted edge-crossing number (X_w) is one; in this case, note that the reservoir placement is changed (or equivalently, input fluids are allocated to them differently). Finally, by changing the order of mixers ($M_2; M_1; M_3; M_4$), another embedding of the graph is shown in Figure 2(c). The total weighted edge-crossing (X_w) remains the same as in the previous case; however, the geometric distance from M_1 to R_1 and R_3 is reduced, and therefore, the total transportation distance reduces as the workload of these two edges are higher than those to M_2. In a general scenario, the number of crossings can be minimized by permuting the reservoirs (mixers) among themselves, and placing them around the boundary (on-chip) appropriately. As the transportation workload of a reservoir-mixer pair increases, the impact of a crossover point becomes more pronounced in terms of routing distance and the frequency of stalls. We define "weighted-distance" between a reservoir R_i and a mixer M_j as the product of geometric distance between them and the weight $w(e_{ij})$ of the edge between R_i and M_j. Thus, a reembedding that reduces the total weighted-geometric-distance between the reservoirs and mixers, is also preferable. Note that in a planar embedding of a quadrilateral where no three points are collinear, the combined length of two nonadjacent sides is always less than that of the two crossing-diagonals (by virtue of the triangle inequality). This fact may be utilized to reorder the resources so that the reservoir-to-mixer weighted-distances are reduced. Hence, in the proposed layout-aware resource placement technique, by changing the location of the resources, we will attempt to minimize the weighted-crossing-number and weighted-geometric-distance so that the number of possible stalls and droplet transportation distance during mixture preparation are reduced.

3.2. Crossing Number in a Bipartite Graph

A common point of two edges in a drawing of bipartite graph that is not an incident vertex is called a crossing. The *crossing number* $cr(G)$ is defined as the minimum number of crossings in any drawing of G [Buchheim et al. 2013]. Zarankiewicz (and independently Urbaník) conjectured that the crossing number of a complete bipartite graph $K_{m,n}$ as $cr(K_{m,n}) = \lfloor \frac{m}{2} \rfloor \lfloor \frac{m-1}{2} \rfloor \lfloor \frac{n}{2} \rfloor \lfloor \frac{n-1}{2} \rfloor$ [Buchheim et al. 2013]. However, the correctness proof of this equation is still unknown [Buchheim et al. 2008, 2013; Richter and Thomassen 1997; Schaefer 2013; Tamassia 2013]. Garey and Johnson proved that the general crossing minimization problem is *NP-complete* [Buchheim et al. 2013; Garey and Johnson 1979]. Hence, in this article, we propose a heuristic technique to reduce the total weights of the edge-crossings among all the droplet transportation routes for a mixture preparation.

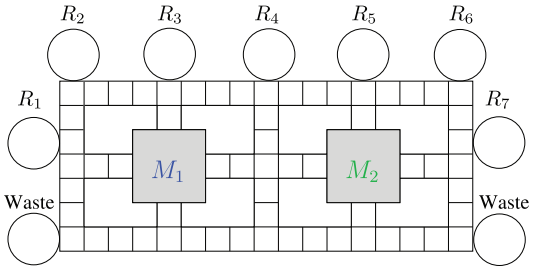


Fig. 3. An example DMF biochip layout for mixture-preparation.

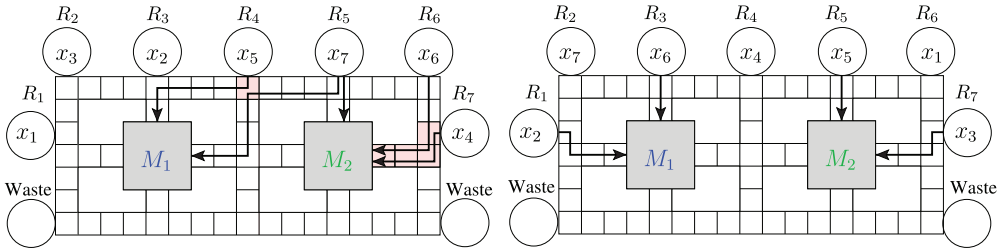


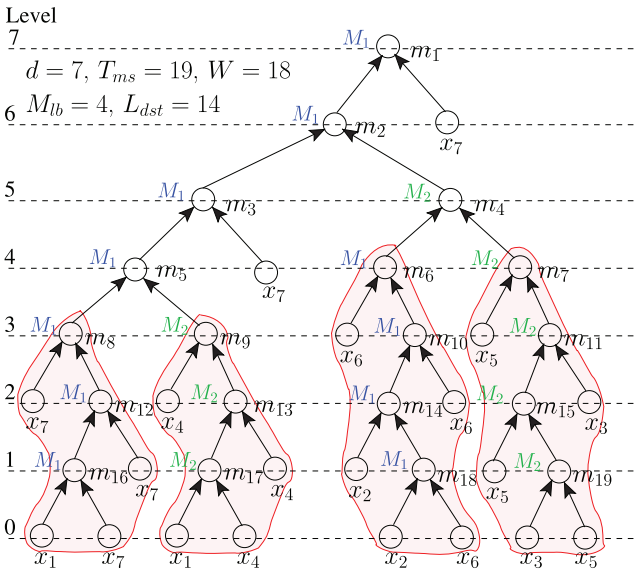
Fig. 4. Droplet routes for the mixing steps of the subtree rooted (a) at m_6 of the mixing tree in Figure 1(b), and (b) of the subtree rooted at m_4 of the mixing tree in Figure 5(a).

3.3. Arrangement of Reservoirs and Mixer Scheduling

The reservoirs (fluid dispensers and waste collectors) of a chip are considered as physical resources and they are usually located around the boundary of the chip. The mixers on a DMF platform can be thought as virtual resources as they can be instantiated on a vacant area of the chip as needed. The mixer geometry can have different values of aspect-ratio, which determine mixing time [Paik et al. 2003a, 2003b]. Figure 3 shows a layout of a biochip with seven reactant dispensers $R_1, R_2, \ldots, R_7$, two waste reservoirs, and two (3×3) mix/split modules M_1 and M_2, which can perform mix and split operations in both horizontal and vertical directions. We assume that each reactant fluid is loaded into a dedicated reservoir. In order to implement a mixing graph onto a physical layout, we need to perform the following two tasks: (1) allocating the reservoirs to reactant fluids (reservoir allocation to the leaf nodes of the mixing tree), and (2) assigning each mix-split operation (an internal node of the mixing tree) to a mixer, referred to as mixer assignment.

Given a number of mixers, the scheduling scheme *OSM* (Optimal-Scheduling-With-*M*-Mixers) [Luo and Akella 2011] can be used to assign the mixers. Assuming two mixers M_1, M_2, their assignments for the ratio $2 : 3 : 5 : 7 : 11 : 13 : 87$, are shown in Figure 1(b). A greedy method is used to obtain a reservoir allocation as shown in Figure 4(a). Corresponding to the mixing subtree rooted at m_6 of Figure 1(b), the droplet routing paths from the fluid reservoirs are indicated with directed lines in Figure 1(a), some of which are overlapping. Hence, movement of these droplets should be handled accordingly in order to avoid any any routing conflict.

We consider another mixing tree for the same target ratio $2 : 3 : 5 : 7 : 11 : 13 : 87$ as shown in Figure 5(a) for which a different reservoir allocation is shown in Figure 4(b). In this case, the droplet transportation paths corresponding to the mixing subtree rooted at node m_4 of Figure 5(a), are shown in Figure 4(b). Here, the eight mixing steps in the subtree rooted at m_4 can be performed with no crossover. It is observed



Mixer	Reservoir						
	x_1	x_2	x_3	x_4	x_5	x_6	x_7
M_1	1	2	0	0	0	3	5
M_2	1	0	2	3	3	0	0

(b)

Fig. 5. An alternative mixing tree for the example ratio 2 : 3 : 5 : 7 : 11 : 13 : 87, and (b) a table containing reservoir-to-mixer droplet transportation workloads.

that for the target ratio 2 : 3 : 5 : 7 : 11 : 13 : 87, if *MinMix*-tree is executed on a chip of Figure 4(a), the weighted-crossing number turns out to be four. However, for the mixing tree of Figure 5(a), this number becomes two. Also, the total reservoir-to-mixer weighted-distance is reduced in the latter case. Thus, even though the latter case requires more mix-split steps, the overall performance of the algorithm improves.

Note that if M_{lb} of a mixing tree is greater than one ($M_{lb} > 1$) and only one on-chip mixer is available, then T_{ms} mix-split cycles are required for completion of the mixing process, where T_{ms} is the total number of nonleaf nodes in the mixing tree. Clearly, mixer assignment and reservoir allocation strongly will depend on the characteristics of the mixing tree, number of reactant fluids, and the number of on-chip mixer modules, which are concurrently available.

For the following discussions, we define the following terms.

Definition 1 (*Dilution / Mixing Subtree*). A tallest subtree with only one nonleaf node at each level of the mixing tree is called a dilution/mixing subtree with a limited number, typically 2 to 4, of distinct leaf nodes (distinct reactant fluids).

Definition 2 (*Dilution Subtree*). A dilution/mixing subtree with only two reactant fluids is called a dilution subtree, and it represents the mixing of two reactant fluids in a certain ratio of their concentration factors.

A mixing tree may contain multiple dilution/mixing subtrees or dilution subtrees of different depths in the mixing tree. The sum of the depths of all the dilution subtrees in a mixing tree is referred to as the total depth of all dilution subtrees and it denoted by L_{dst}. In the example mixing tree of Figure 1(b) obtained by *MinMix* [Thies et al. 2008], the dilution subtrees are shown within *pink*-shaded regions (here, $L_{dst} = 6$). Similarly, in the mixing tree of Figure 5(a), the dilution subtrees are marked (here, $L_{dst} = 14$). For the latter tree, the workload table for reservoir-to-mixer droplet transportation is shown in Figure 5(b). Compared to the workload table as in Figure 1(c), this has more "zero" entries, and the workload on mixers M_1 and M_2 are more skewed with respect to

input reactants. This happens because the mixing tree of Figure 5(a) has taller dilution subtrees and the value of L_{dst} is larger compared to those in Figure 1(c). Note that each dilution subtree can be scheduled and executed in one mixer for consecutive time steps, and the reservoirs holding the two corresponding reactants can be placed at the near proximity of that mixer. Such a localized placement of reservoirs around a mixer provides two advantages: (a) the reservoir-to-mixer weighted-distance is reduced, and (b) the transportation of droplets from a reservoir to a mixer can be pipelined as long as the subtree is being executed, and hence no crossover of droplets will occur among their pathways.

Another interesting observation can be made from Figure 1(b) and Figure 5(a). Note that the latter consists of three tall dilution subtrees whose input sets of reactants $\{x_1, x_4\}$, $\{x_2, x_6\}$, $\{x_3, x_5\}$ are disjoint. Thus, each pair of reactants can be loaded into two reservoirs in the near vicinity of the corresponding mixer. On the other hand, in the mixing graph of Figure 1(b) produced by *MinMix*, the presence of intersection among the reactant-sets causes crossovers among the droplet routing paths. This motivates us to device a new mixing algorithm that will produce taller and disjoint-input dilution subtrees in the mixing graph, so that its implementation on a biochip will be layout-friendly. To the best of our knowledge, no such layout-driven mixing algorithm has been studied in the literature. In this article, we thus focus on two tasks: (i) propose a layout-friendly mixing algorithm that produces a mixing tree with taller and disjoint-input dilution subtrees; (ii) given any mixing algorithm, determine reservoir allocation and mixer placement on the layout that minimizes droplet crossovers and transportation distance.

Note that the proposed mixing and resource allocation scheme differs from the usual phase of resource binding that is employed during high-level of synthesis (HLS) of a biochip. Given a task graph and a layout with physical resources as input, the objective of HLS is to allocate the physical resources (dispensers, detectors, heaters) and virtual resources (mixers) on a chip, in space and in time, such that the chip area and assay-completion time are minimized subject to the fulfillment of all fluidic constraints [Chakrabarty and Su 2007; Su and Chakrabarty 2004, 2008]. In this work, our objective is different: to design a task graph (a mixing graph in our context), which will be more layout-friendly, in terms of fewer crossovers among droplet pathways, and shorter droplet-transportation distance, during implementation.

3.4. Problem Formulation

The problem of automated mixture-preparation of N reactant fluids can be stated as follows.

Input. A supply of N reactant fluids $x_1, x_2, \ldots, x_N$, each with $CF = 1$ and a desired target ratio of CFs $a_1, a_2, \ldots, a_N$ for mixing them; the maximum error in CF of each component fluid is $\frac{1}{2^d}$, for a given integer d.

Output. A mixing tree of depth at most d denoting the sequence of $(1:1)$ mix-split steps (m_is) for preparing the target ratio.

4. LAYOUT-AWARE MIXING ALGORITHM

In this section, we present a layout-aware mixing algorithm referred to as Ratio-ed Mixing Algorithm (*RMA*). The proposed algorithm is based on fractional decomposition of the algebraic expression corresponding to a desired ratio, which finally leads to a mixing tree corresponding to the target ratio.

4.1. Algebraic Expression for a Target Ratio

For a desired ratio $a_1 : a_2 : \ldots : a_N$, let $L = \sum_{i=1}^{N} a_i$ be the sum of ratio components (nonzero positive integers), the target mixture can be represented by an algebraic

expression $X(N) = \frac{a_1x_1+a_2x_2+\cdots+a_Nx_N}{L}$, where $a_1, a_2, \ldots, a_N$ are the positive integers corresponding to the desired ratio and $x_1, x_2, \ldots, x_N$ are the variable names corresponding to N reactant fluids. For example, a mixture of seven fluids with a desired ratio 2 : 3 : 5 : 7 : 11 : 13 : 87 can be represented by the algebraic expression $X(7) = \frac{2x_1+3x_2+5x_3+7x_4+11x_5+13x_6+87x_7}{128}$, where $L = 128$ and $x_1, x_2, \ldots, x_7$ are seven different reactant fluids. The *CF* c_i of a reactant fluid x_i in the target mixture can be represented by $c_i = \frac{a_i}{L}$ and $\sum_{i=1}^{N} c_i = 1$.

4.2. Fractional Decomposition of the Expression $X(N)$

For a given target ratio, the algebraic expression $X(N) = \frac{a_1x_1+a_2x_2+\cdots+a_Nx_N}{2^d}$ can be decomposed into a fractional representation, as stated in the following theorem.

THEOREM 4.1. *If $X(N) = \frac{a_1x_1+a_1x_2+\cdots+a_Nx_N}{2^d}$, where $d \geq N$, with all a_i's are positive integers and $\sum_{i=1}^{N} a_i = 2^d$, then $X(N)$ can always be expressed as a fractional decomposition of n terms, $n \geq d$: $X(N) = \sum_n[\frac{1}{2}(x_{i1} + \frac{1}{2}(x_{i2} + \frac{1}{2}(\cdots + \frac{1}{2}(x_{i(r-1)} + x_{ir}))))]$, where $x_{ij} \in \{x_1, x_2, \ldots, x_N\}$ with repetitions, $r \geq N$ and $r < d$. Let q denote the number of "+" signs in the fractional decomposition; $1 \leq r < q$.*

PROOF. Let a_i be the largest integer in $\{a_1, a_2, \ldots, a_N\}$. If $a_i \geq 2^{d-1}$, we can write

$$X(N) = \frac{a_1x_1 + a_2x_2 + \cdots + a_Nx_N}{2^d} \tag{1}$$

$$= \frac{1}{2}\left(x_i + \frac{1}{2^{d-1}}\left(a_1x_1 + \cdots + a_{i-1}x_{i-1} + (a_i - 2^{d-1})x_i + \cdots + a_Nx_N\right)\right). \tag{2}$$

The expression within the inner parentheses can further be decomposed applying the same process.

If $a_i < 2^{d-1}$, then there exists a j such that $a_1 + a_2 + \cdots + a_{j-1} < 2^{d-1} < a_1 + a_2 + \cdots + a_j$. Hence, $X(N)$ can be written as

$$X(N) = \frac{a_1x_1 + a_2x_2 + \cdots + a_Nx_N}{2^d} \tag{3}$$

$$= \frac{1}{2}\left(\frac{1}{2^{d-1}}\left(a_1x_1 + \cdots + a_{j-1}x_{j-1} + a'_jx_j\right) + \frac{1}{2^{d-1}}\left((a_j - a'_j)x_j + a_{j+1}x_{j+1} + \cdots + a_Nx_N\right)\right), \tag{4}$$

where $a'_j = 2^{d-1} - \sum_{k=1}^{j-1} a_k$.

Recursively applying this scheme on the linear algebraic expressions, we obtain

$$X(N) = \sum\left[\frac{1}{2}\left(x_{i1} + \frac{1}{2}\left(x_{i2} + \frac{1}{2}\left(\cdots + \frac{1}{2}(x_{i(r-1)} + x_{ir})\right)\right)\right)\right] \tag{5}$$

where $x_{ij} \in \{x_1, x_2, \ldots, x_N\}$ with repetitions. □

For example, the target ratio 2 : 3 : 5 : 7 : 11 : 13 : 87 can be represented by the algebraic expression $X(7) = \frac{2x_1+3x_2+5x_3+7x_4+11x_5+13x_6+87x_7}{128}$. A fractional decomposition of

$X(7)$ can be obtained as follows:

$$X(7) = \frac{1}{2}\Bigg[\frac{1}{2}\Bigg(\frac{1}{2}\Bigg(\frac{1}{2}\Bigg(\frac{1}{2}\Bigg(\frac{1}{2}(x_1+x_7)+x_7\Bigg)+x_7\Bigg) + \frac{1}{2}\Bigg(\frac{1}{2}\Bigg(\frac{1}{2}(x_1+x_4)+x_4\Bigg)+x_4\Bigg)\Bigg)+x_7\Bigg) + \frac{1}{2}\Bigg(\frac{1}{2}\Bigg(\frac{1}{2}\Bigg(\frac{1}{2}(x_2+x_6)+x_2\Bigg)+x_6\Bigg)+x_6\Bigg) + \frac{1}{2}\Bigg(\frac{1}{2}\Bigg(\frac{1}{2}\Bigg(\frac{1}{2}(x_3+x_5)+x_5\Bigg)+x_3\Bigg)+x_5\Bigg)\Bigg)\Bigg)+x_7\Bigg]. \quad (6)$$

Observation 4.1 (*Correspondence Between Fractional Decomposition and Mixing Tree*). Each fractional decomposition of the algebraic expression of a target ratio can be represented by a mixing tree.

Note that (1 : 1) mixing of two different reactant fluids of *CF* x_1 and x_2, produces a mixture denoted by $\frac{x_1+x_2}{2}$. In the next mixing step, one droplet of this mixture is used to mix with a third reactant fluid, say x_3, in a ratio 1 : 1 to produce a mixture denoted by $\frac{1}{2}(\frac{x_1+x_2}{2}+x_3)$. Thus, the fractional decomposition obtained from the target ratio can be mapped into a mixing tree, following the sequences of one (1 : 1) mix and a consecutive split operation. The depth of the mixing tree is d and the total number (1 : 1) mix-split steps (T_{ms}) is the total number of "+" signs in the fractional decomposition, that is, q, where $(q+1)$ is the number of leaf nodes in the mixing tree denoting the input reactants. Thus, there is a one-to-one correspondence between a fractional decomposition and the mixing tree as constructed above, corresponding to a target ratio. For example, the fractional decomposition of the algebraic expression $X(7)$ given by Equation (6) can be directly translated to the mixing tree of Figure 5(a) for the desired ratio 2 : 3 : 5 : 7 : 11 : 13 : 87.

4.3. Proposed Algorithm: *RMA*

Assume that the desired ratio $a_1 : a_2 : \ldots : a_N$ is an approximated ratio for which the ratio-sum, $L = \sum_{i=1}^{N} a_i = 2^d$, for an accuracy level d of *CF*s in the mixture. In this section, we present the mixing algorithm *RMA* that uses an integer-partitioning scheme to obtain a fractional decomposition of the algebraic expression $X(N)$ corresponding to the desired ratio. It distributes the component reactant fluids in a disjoint manner, as far as possible, from the early stage in a top-down fashion to generate the mixing tree. The pseudocode of the proposed algorithm *RMA* written as Algorithm 1; this calls a procedure *Partitioning* described by Algorithm 2, with the initial partition $\langle P, L\rangle$ corresponding to the algebraic expression $X(N)$. The procedure *Partitioning* uses another procedure *Expression_Partition* given by Algorithm 3 to obtain two new partitions $\langle P_1, L_1\rangle$ and $\langle P_2, L_2\rangle$ from the input partition $\langle P, L\rangle$. Each time two new partitions are returned to *Partitioning* from *Expression_Partition*, two (left and right) child nodes are created to the parent node in the mixing tree T. The procedure *Partitioning* recursively calls itself to construct the mixing tree. If a new partition $\langle P, L\rangle$ contains only one variable x_i with $L = 1$, the procedure *Partitioning* stops partitioning further.

4.4. Analysis of *RMA*

4.4.1. Characteristics of the Mixing Tree. The procedure *RMA* works in a top-down fashion unlike *MinMix* [Thies et al. 2008], that is, the mixing tree is obtained by decomposing the algebraic expression that represents the target ratio, which serves as the root of the tree. The tree is recursively constructed from the root node to the leaves. Note

ALGORITHM 1: *RMA*($\langle a_1 : a_2 : \ldots : a_N \rangle, d$)

1: Obtain the algebraic expression $X(N) = \dfrac{a_1x_1 + a_2x_2 + \cdots + a_Nx_N}{L}$, where $L = \sum_{i=1}^{N} a_i = 2^d$.
2: Represent an integer partition as $\langle P, L \rangle$, where $P = a_1x_1 + a_2x_2 + \cdots + a_Nx_N = \sum_{i=1}^{N} a_ix_i$.
3: Create the root node of the mixing tree $\mathcal{T}$ with partition $\langle P, L \rangle$.
4: $\mathcal{T}$ = *Partitioning*($\langle P, L \rangle$).
5: **return** $\mathcal{T}$.

ALGORITHM 2: *Partitioning*($\langle P, L \rangle$)

1: **if** $L = 1$ **then**
2: **return** NULL.
3: **else**
4: Run *Expression_Partition*($\langle P, L \rangle$) and obtain two new partitions, $\langle P_1, L_1 \rangle$ and $\langle P_2, L_2 \rangle$ corresponding to two (left and right) child nodes of the parent node with $\langle P, L \rangle$ in the mixing tree.
5: Create two (left and right) child nodes with these two partitions, respectively, of the parent node with $\langle P, L \rangle$ in the mixing tree $\mathcal{T}$.
6: *Partitioning*($\langle P_1, L_1 \rangle$).
7: *Partitioning*($\langle P_2, L_2 \rangle$).
8: **end if**

ALGORITHM 3: *Expression_Partition*($\langle P, L \rangle$)

1: Let a_i be the largest integer coefficient in the partition $\langle P, L \rangle$ (corresponding to the variable x_i).
2: Set $L_1 = L_2 = \frac{L}{2}$.
3: **if** $a_i \geq L_1$ **then**
4: Create two partitions as $\langle P_1, L_1 \rangle$ and $\langle P_2, L_2 \rangle$, where $P_1 = x_i$, $L_1 = 1$ and $P_2 = P - a_ix_i$.
5: **else**
6: Create two partitions as $\langle P_1, L_1 \rangle$ and $\langle P_2, L_2 \rangle$, where $P_1 = a_ix_i$ and $P_2 = P - a_ix_i$.
7: Compute the required extra integer coefficient in first partition to make $L_1 = \frac{L}{2}$, $E = \frac{L}{2} - a_i$.
8: **if** $E = a_j$ for any coefficient a_j in $\langle P_2, L_2 \rangle$ (corresponding to the variable x_j) **then**
9: /* If the required extra integer coefficient in the first partition is found exactly equal to an integer coefficient in the second partition */
10: $P_1 = P_1 + a_jx_j$ and $P_2 = P_2 - a_jx_j$.
11: **else if** E is exactly equal to the sum of n_1 coefficients in $\langle P_2, L_2 \rangle$ **and** $n_1 < \lfloor \frac{1}{2} \times$(the number of nonzero coefficients in $\langle P_2, L_2 \rangle \rfloor$ **then**
12: Move those n_1 terms from $\langle P_2, L_2 \rangle$ to $\langle P_1, L_1 \rangle$.
13: **else**
14: Let a_k = be the second largest integer coefficient in $\langle P, L \rangle$ (corresponding to the variable x_k) and $a_k > E$.
15: $P_1 = P_1 + E.x_k$ and $P_2 = P_2 - E.x_k$.
16: **end if**
17: **end if**
18: **return** Two new partitions $\langle P_1, L_1 \rangle$ and $\langle P_2, L_2 \rangle$.

that the mixing tree is a full binary tree, that is, every node has exactly two children (representing a 1 : 1 mix/split step), or no children (representing an input reactant, that is, a variable) [Aho et al. 1974; Roy et al. 2014b]. The root starts at level d. The partitioning of the tree into a left and a right subtree at a level ℓ, is obtained in such a way that the sum of the component ratios (or the coefficients of variables) in each child node becomes $2^{\ell-1}$. We call this condition *half-sum*. For example, *RMA* partitions the algebraic expression $X(7)$ corresponding to the target ratio 2 : 3 : 5 : 7 : 11 : 13 : 87, as in Figure 6, and the mixing tree of Figure 5(a) can be constructed directly following the partition.

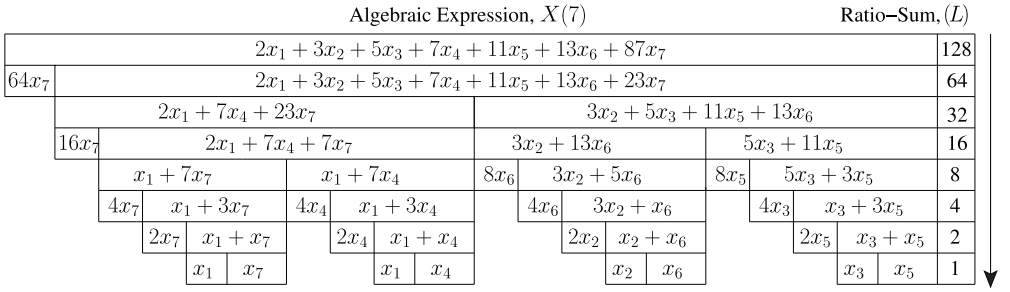


Fig. 6. For the ratio 2 : 3 : 5 : 7 : 11 : 13 : 87, partitioning of algebraic expressions to construct the mixing tree of Figure 5(a).

Equation (5) reveals that the half-sum partitioning condition can always be fulfilled while keeping at most one common variable (input reactant) between the two subtrees. For example, in Figure 6, at the top level of partitioning, only the input x_7 is kept common between the two subtrees (second row of Figure 6), and at the next level, the inputs to the two subtrees, that is, (x_1, x_4, x_7) and (x_2, x_3, x_5, x_6), are disjoint (third row of Figure 6). Since the component ratios are partitioned at every level based on this principle, the intersection among the input sets of two subtress includes at most one reactant. As a result, the dilution subtrees tend to become taller and mutually "almost-disjoint" (having at most one common input reactant between every pair of subtrees), a property which is desirable for enabling a layout-friendly placement of reservoirs and mixers. The total depth of all dilution subtrees also increases compared to that obtained otherwise. Thus, *RMA*, by virtue of its top-down design, automatically provides a layout-aware solution of the mixing problem.

4.4.2. Time-Complexity of RMA. The time-complexity of *RMA* depends on the time-complexity of the algorithm used to find the exact sum in Step 11 of *Expression_Partition*. This problem can be stated as a subset-sum problem, that can be solved by using a pseudo-polynomial time dynamic programming approach [Kleinberg and Tardos 2005]. The time complexity of the pseudo-polynomial time dynamic programming approach depends on the size of input numbers, that is, $\mathcal{O}(N.E)$, where E is the desired sum to find in a set of N integers. At the lowest level (level 1) of the mixing tree, at most $\frac{N}{2}$ times the procedure *Partitioning* can be called and for a mixing tree of depth d these calls can recursively occur d times. Hence, at most $d.\frac{N}{2}$ times the procedure *Expression_Partition* may be called. At every level ℓ, *RMA* first searches whether the half-sum condition is satisfied with two disjoint set of inputs, and if found, proceeds to the next level with this decomposition. Otherwise, using a greedy approach, it keeps on including the input coefficients in one partition until their sum exceeds $2^{\ell-1}$, and selects one of them to split its coefficient so that the half-sum is achieved in both the resulting partitions. Thus, the time-complexity of the proposed mixing algorithm *RMA* is $\mathcal{O}(N.E.d.\frac{N}{2})$, that is, $\mathcal{O}(d.N^2)$, since $E \ll 2^d$.

4.4.3. Some Examples Highlighting RMA. We simulated both the mixing algorithms (*MinMix* [Thies et al. 2008] and *RMA*) for several target ratios with different number of input reactants that are used in real-life bioprotocols as described in BioCoder tool[2]. For example, a target ratio 40 : 10 : 1 : 1 : 48 is used in "Splinkerette PCR method". The proposed algorithm *RMA* determines a mixing tree with taller dilution subtrees (i.e., the total depth of dilution subtrees, $L_{dst} = 7$) as shown in Figure 7(b). In contrast,

[2]BioCoder: A Programming Language for Biology Protocols, Microsoft Research India, http://research.microsoft.com/en-us/um/india/projects/biocoder/, December, 2009.

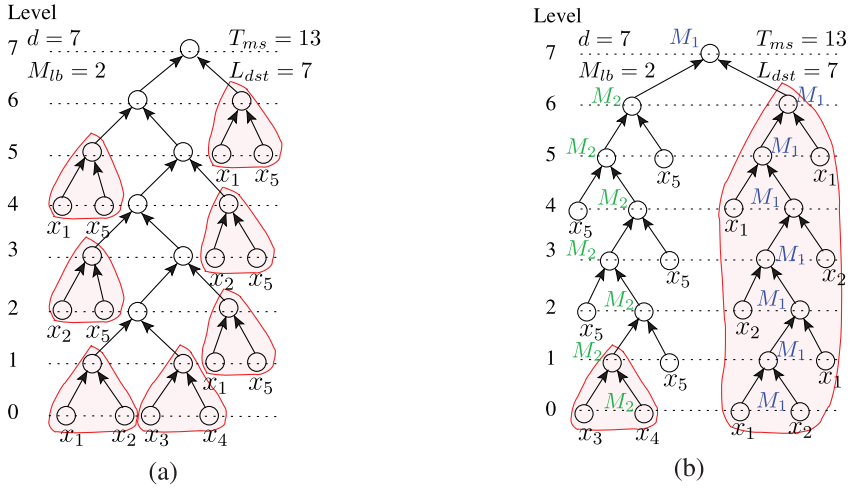


Fig. 7. For an example ratio 40 : 10 : 1 : 1 : 48 (≡ 51 : 13 : 1 : 1 : 62, in binary scale) used in "Splinkerette PCR method", mixing tree obtained by (a) *MinMix*, and (b) *RMA*.

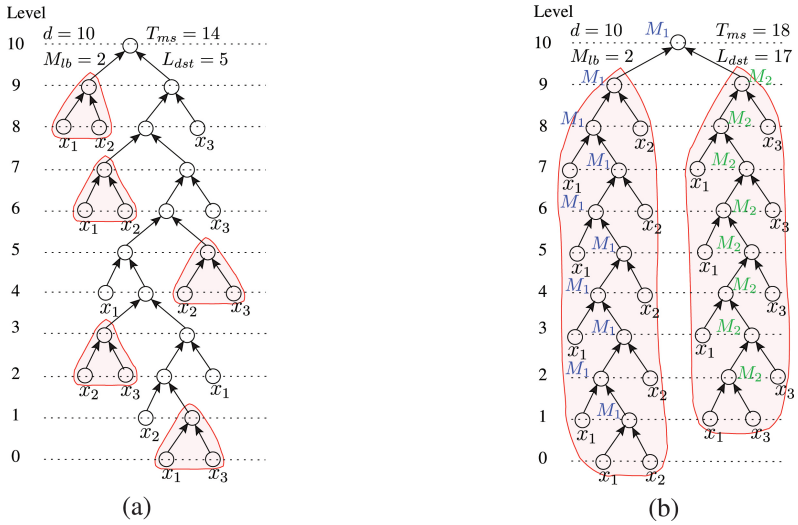


Fig. 8. For a target ratio 341 : 341 : 342, mixing trees obtained by (a) *MinMix*, and (b) *RMA*.

MinMix constructs a mixing tree as shown in Figure 7(a), which has seven dilution subtrees (here, $L_{dst} = 7$). Due to the presence of many common variables between the two subtrees of *MinMix*-tree, a physical implementation of this tree on a layout will necessitate a number of crossovers among droplet-routing paths (see Figure 3). On the other hand, *RMA* offers a more layout-friendly implementation. Note that in the *RMA*-tree, the subtree that is rooted at M_2 of level 6, and leaves at level 1, needs only reactant x_7 and an intermediate droplet from mixer M_2 at every node. Hence, this subtree (whose depth is five), also resembles a dilution subtree and can be implemented without any droplet crossover in its implementation as well.

For the target ratio 1 : 1 : 1 approximated as 341 : 341 : 342 in the binary scale, Figure 8(a) and 8(b) depict the mixing trees obtained by *MinMix* [Thies et al. 2008] and *RMA* respectively. The *RMA*-tree consists of taller dilution subtrees, and it can

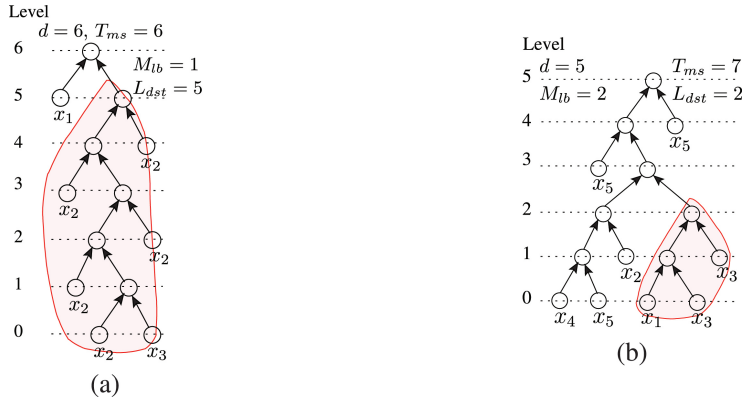


Fig. 9. The same mixing tree obtained by both the methods, *MinMix* and *RMA*, for two example target ratios (a) 25 : 24 : 1, and (b) 1 : 2 : 3 : 1 : 23.

Table I. Comparative Results of *MinMix* and *RMA* for Some Randomly Chosen Target Ratios†

Example	Target Ratio	Accuracy Level (d)	*MinMix*				*RMA*			
			T_{ms}	W	M_{lb}	L_{dst}	T_{ms}	W	M_{lb}	L_{dst}
1.	40:10:1:1:48 (≡ 51:13:1:1:62)	7	13	12	2	7	13	12	2	7
2.	13:12:5:2	5	7	6	2	3	7	6	2	3
3.	15:7:4:4:1:1	5	10	9	3	5	10	9	3	8
4.	2:3:5:7:11:13:87	7	18	17	4	8	19	18	4	14
5.	341:341:342	10	14	13	2	5	18	17	2	17
6.	12:7:7:3:3	5	11	10	3	5	12	11	3	8
7.	18:5:3:3:3	5	9	8	3	4	10	9	3	7
8.	7:14:11	5	8	7	2	5	8	7	2	7
9.	9:8:8:8:7:7:5:5:4:1:1:1	6	18	17	4	9	20	19	5	13
10.	34:7:5:5:5:4:3:1	6	14	13	4	6	16	15	4	10

†In order to produce two droplets of a target ratio with accuracy level d, T_{ms}: total number of (1 : 1) mix-split steps, W: total number of waste droplets generated, M_{lb}: minimum number of mixers required for earliest completion of executing a mixing tree, and L_{dst}: total depth of all the dilution subtrees in a mixing tree.

be easily implemented on chip with two mixers without any crossover among droplet paths. However, the number of mix-split steps (T_{ms}) increases compared to that needed by *MinMix* [Thies et al. 2008].

A target ratio, in which the component integers are *highly-skewed* (for example 1 : 1 : 1 : 1000 or 25 : 24 : 1), both the methods (*MinMix* [Thies et al. 2008] and *RMA*) are likely to provide the same mixing tree. For example, the mixing trees for two target ratios 25 : 24 : 1 (used in "One step miniprep method", see footnote 2) and 1 : 2 : 3 : 1 : 23 (used in "Splinkerette PCR method", see footnote 2) as shown in Figures 9(a) and 9(b), respectively.

4.5. Simulation Results: *MinMix* vs. *RMA*

Table I presents the comparative results of *MinMix* and *RMA* for some randomly chosen target ratios with different accuracy levels (d). It is observed that the total number of (1 : 1) mix-split steps (T_{ms}), the total number of waste droplets produced (W) and the minimum number of mixers required for earliest completion of executing a mixing tree (M_{lb}) are almost same for both the mixing trees obtained by *MinMix* and *RMA*. Whereas, the total depth of all dilution subtrees in a mixing tree (L_{dst}) increases in *RMA* compared to *MinMix*.

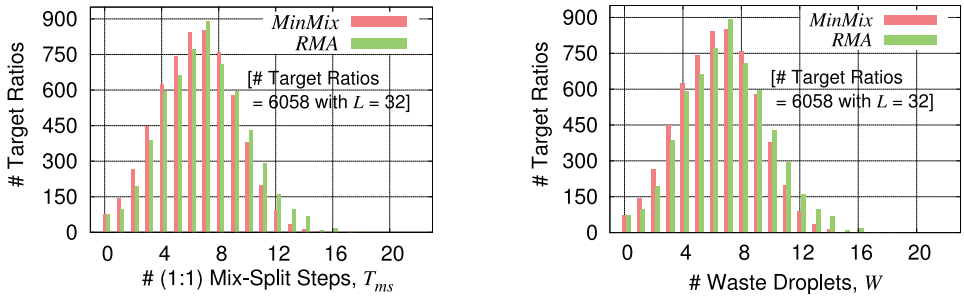


Fig. 10. Histograms for the distributions of (a) the total number of mix-split steps, T_{ms} and (b) the total number of waste droplets, W, in the mixing trees obtained by both *MinMix* and *RMA*.

We have also carried out simulation experiments with a large number of target ratios for a comparative evaluation of *RMA* and *MinMix* [Thies et al. 2008]. A detailed discussion on how the samples are generated by using an integer-partitioning technique, is presented in the Appendix. For 6058 synthetic target ratios of N different fluids ($3 \leq N \leq 12$) with the ratio-sum $L = 32$ (i.e., the accuracy level $d = 5$), both the mixing algorithms (*MinMix* [Thies et al. 2008] and *RMA*) are studied. We compute different performance parameters such as the total number of mix-split steps (T_{ms}), the total number of waste droplets (W) and the total depth of the dilution subtrees in the mixing tree (L_{dst}).

Figures 10(a) and 10(b) show that the distributions of T_{ms} and W are very similar for both the mixing algorithms.

In *RMA*, the envelopes of the distributions are slightly shifted towards the right both for T_{ms} and W. Hence, on the average, *RMA* produces only a slight increase in T_{ms} and W compared to *MinMix*. However, in Figure 11, the distribution of L_{dst} shows a significant right-shift for *RMA* compared to that of *MinMix*. This is indicative of the fact that on the average, there will be more frequent occurrences of dilution subtrees with large values of total depth, when *RMA* is deployed. Thus, *RMA* will offer greater benefit over *MinMix* from the viewpoint of reservoir placement and droplet transportation. In other words, a reduction in weighted-crossing number and transportation distance is more likely to be observed during the implementation of *RMA*. Based on the analysis discussed in Section 4.4.1 and the above empirical evidence, we now summarize a characteristic property of the top-down mixing algorithm *RMA* as follows.

Observation 4.2. The proposed mixing algorithm *RMA* produces a mixing tree with taller and almost-disjoint dilution subtrees, and with a higher value of L_{dst}, that is, the total depth of dilution subtrees, compared to those obtained by *MinMix*.

5. ROUTING-AWARE RESOURCE ALLOCATION FOR MIXTURE PREPARATION

As discussed in previous sections, the procedure *RMA* proposed in this work, provides a layout-friendly mixing tree. Given such a tree, the boundary reservoirs can be suitably allocated to the reactant fluids and the mixers can be placed such that droplet crossovers and transportation distances are reduced. In this section, we present an allocation and placement algorithm that can be used to determine the relative positions of on-chip mixers, and fluid reservoirs around the chip boundary, based on a graph-based heuristic approach. The method only only applies to the mixing trees produced by *RMA*, but also to those obtained by other algorithms such as *MinMix* [Thies et al.

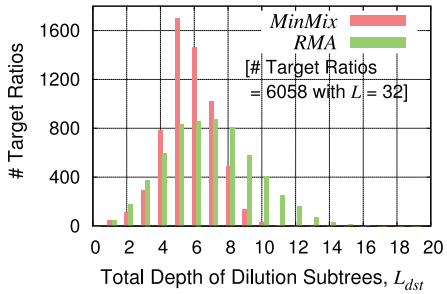


Fig. 11. Histogram for the distribution of the total length of dilution subtrees, L_{dst}, in the mixing trees obtained by both *MinMix* and *RMA*.

2008] or *RSM* [Hsieh et al. 2012a]. As a result, droplet transportation time is reduced during the actual implementation of sample preprocessing.

5.1. Motivation

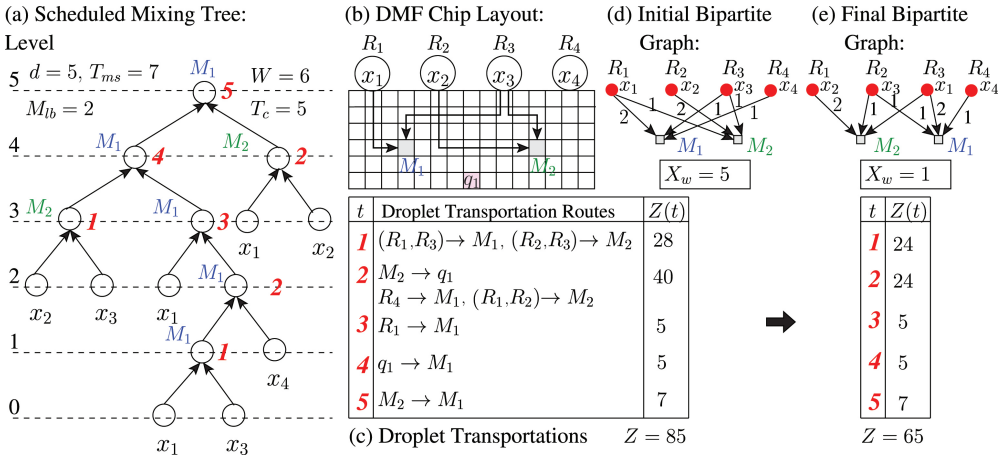
Given a single target ratio, a mixing tree can be constructed by running *RMA*, *MinMix* [Thies et al. 2008], or *RSM* [Hsieh et al. 2012a]. The last one is also suitable for producing multiple target ratios, and for such cases, it outputs a mixing graph based on sharing of intermediate droplets. The placement algorithm proposed in the section will be applicable to both mixing trees and graphs. Before we execute our placement algorithm, we adopt the *OSM* (Optimal-Scheduling-With-*M*-Mixers) procedure [Luo and Akella 2011] to assign M mixers to the nodes of a mixing graph. For a given value of M, this algorithm produces a schedule that optimizes the total time needed in mix/split operations. However, it does not consider the issues of droplet dispensing and transportation time. For simplicity, we assume that a droplet can move from an electrode to its adjacent electrode in one clock cycle, and all mixers require the same time to complete a $(1:1)$ mix-split operation. Depending on the nature of the mixer modules, a mix-split step may require several clock cycles to complete the task [Roy et al. 2014a].

For example, consider a target ratio $13:12:5:2$ of four reactant fluids x_1, x_2, x_3, and x_4. Figure 12(a) shows the *MinMix*-tree scheduled with two mixers M_1 and M_2 (since $M_{lb} = 2$). To each internal node, we attach a time-stamp (t) that reflects the temporal sequence in the task graph. These stamps are labeled as 1 to 5 in 'red' font. A unit time-stamp includes the time needed by a mixer to execute a mix/split step, and the time needed to transport the input droplets from the corresponding reservoirs or mixers. Depending on the schedule, some on-chip storage may be necessary to store the intermediate droplets to be used in subsequent mix operations.

For an illustration, we assume a layout (Figure 12(b)) with two on-chip mixers M_1 and M_2, and four boundary reservoirs R_1, R_2, R_3, R_4 which are loaded with reactant fluids x_1, x_2, x_3 and x_4, respectively.

In the following discussion, by resource-allocation we mean loading of input fluids to the boundary reservoirs and assignment of nonleaf nodes of a mixing tree/graph to the on-chip mixers. The allocation of a reservoir R_j to an input fluid x_i is denoted by the mapping $x_i \rightarrow R_j$, and the mixer assignments are obtained from the scheduled mixing tree. Figure 12(b) shows the droplet-routing paths for time-stamp $t = 1$. Figure 12(c) lists the overall schedule and routing paths.

Let $Z(t)$ denote the number of cells (electrodes) that are activated for transporting all droplets until time-stamp t. Thus, the total weighted-distance considering all reservoir-to-mixer droplet transportation can be computed as $Z = \sum_t Z(t)$. Also, the number



t	Droplet Transportation Routes	$Z(t)$
1	$(R_1,R_3) \rightarrow M_1$, $(R_2,R_3) \rightarrow M_2$	28
2	$M_2 \rightarrow q_1$ $R_4 \rightarrow M_1$, $(R_1,R_2) \rightarrow M_2$	40
3	$R_1 \rightarrow M_1$	5
4	$q_1 \rightarrow M_1$	5
5	$M_2 \rightarrow M_1$	7

t	$Z(t)$
1	24
2	24
3	5
4	5
5	7

Fig. 12. For a ratio 13 : 12 : 5 : 2, scheduled mixing tree obtained by *MinMix* and layout, droplet transportation routes and bipartite graphs.

of stalls (Δ) required while transporting the droplets to a mixer can be estimated (pessimistically) by the corresponding weighted-crossing number (X_w) in the underlying reservoir-to-mixer bipartite graph, as described in Section 3.

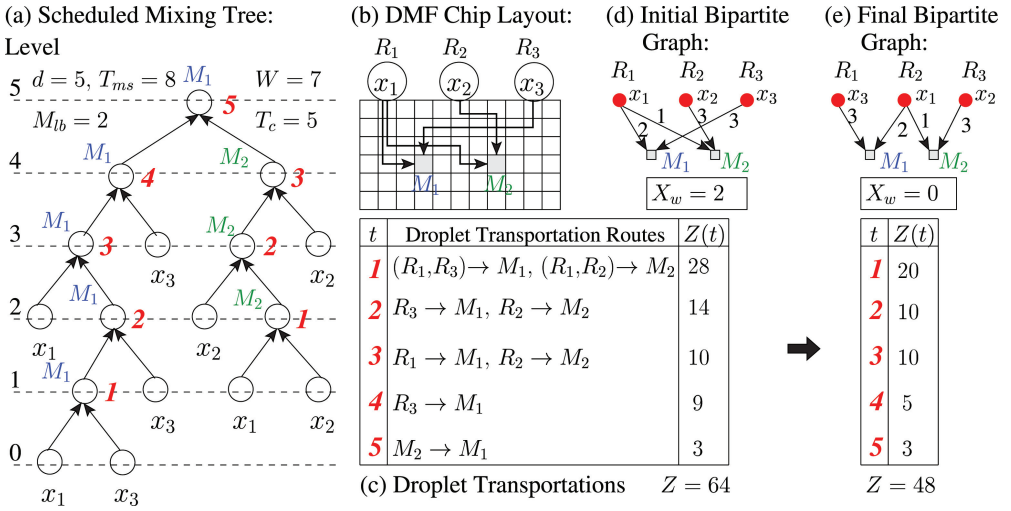
Figure 12(d) depicts the initial resource-allocation with the workloads as the edge weights in the initial bipartite graph for the layout of Figure 12(b), for which $X_w = 5$. A change in resource allocation, obtained by interchanging M_1 and M_2, is shown in Figure 12(e) as the final bipartite graph with the workloads as the edge weights, where $X_w = 1$. Also, the total transportation distance Z reduces compared to that in Figure 12(e).

As discussed in Section 3, reduction in weighted crossing number tends to reduce both the number of stalls and droplet transportation distance. As a result, the total time needed for droplet routing will also be reduced. This fact motivates us to solve the problem of minimizing weighted crossing number in a bipartite graph and explore its application to on-chip mixture-preparation.

For further illustration, we consider another target ratio 7 : 14 : 11 of three reactant fluids x_1, x_2 and x_3. The scheduled *RMA*-tree, and *RSM*-tree [Hsieh et al. 2012a] are shown in Figure 13(a) and Figure 14(a), respectively. A layout with two on-chip mixers and three boundary reservoirs is shown in Figure 13(b) and Figure 14(b). Droplet-transportation routes and time-stamps for these two cases are listed in Figures 13(c) and 14(c), respectively. The corresponding initial and final bipartite graphs along with the workloads as the edge weights are depicted in Figures 13(d) and 13(e), and in Figures 14(d) and 14(e). In both cases, an improvement in the value of weighted crossing number X_w, and total transportation distance Z, can be observed.

5.2. Problem Formulation

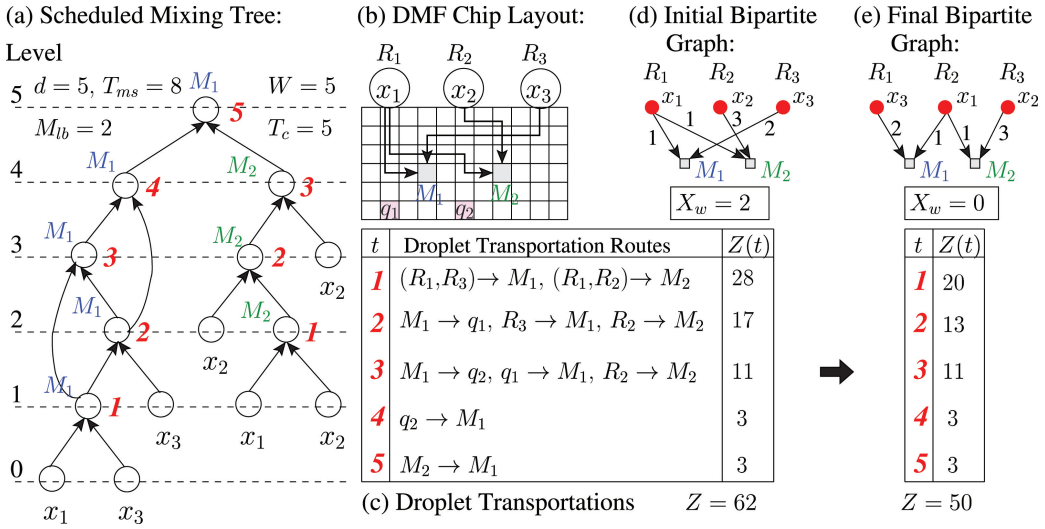
We envisage *resource-allocation* as a mapping function ϕ that provides a linear ordering of the mixers and the input fluids (loaded into the reservoirs), that is, their relative positions in the chip layout from left-to-right. Let the initial and final ordering of mixers and fluid reservoirs be denoted by Π_r^i (Π_m^i) and Π_r^f (Π_m^f), respectively. For a scheduled mixing tree, if the mixers (M_i's) and the fluid-to-reservoir loading ($x_i \rightarrow R_j$) are relatively ordered in such a way that the weighted crossing number X_w in the bipartite graph representing the reservoir-to-mixer workload, is minimized, then the resource-allocation is said to be *optimal* (ϕ_{opt}).



t	Droplet Transportation Routes	$Z(t)$
1	$(R_1,R_3) \rightarrow M_1$, $(R_1,R_2) \rightarrow M_2$	28
2	$R_3 \rightarrow M_1$, $R_2 \rightarrow M_2$	14
3	$R_1 \rightarrow M_1$, $R_2 \rightarrow M_2$	10
4	$R_3 \rightarrow M_1$	9
5	$M_2 \rightarrow M_1$	3

t	$Z(t)$
1	20
2	10
3	10
4	5
5	3

Fig. 13. For a ratio 7 : 14 : 11, scheduled mixing tree obtained by *RMA* and layout, droplet transportation routes and bipartite graphs.



t	Droplet Transportation Routes	$Z(t)$
1	$(R_1,R_3) \rightarrow M_1$, $(R_1,R_2) \rightarrow M_2$	28
2	$M_1 \rightarrow q_1$, $R_3 \rightarrow M_1$, $R_2 \rightarrow M_2$	17
3	$M_1 \rightarrow q_2$, $q_1 \rightarrow M_1$, $R_2 \rightarrow M_2$	11
4	$q_2 \rightarrow M_1$	3
5	$M_2 \rightarrow M_1$	3

t	$Z(t)$
1	20
2	13
3	11
4	3
5	3

Fig. 14. For a ratio 7 : 14 : 11, scheduled mixing tree obtained by *RSM* and layout, droplet transportation routes and bipartite graphs.

The problem of finding the optimal resource-allocation can be stated as follows.

Inputs: (a) N reactant fluids $x_1, x_2, \ldots, x_N$ each with $CF = 100\%$, (b) a target ratio $a_1 : a_2 : \ldots : a_N$ of N reactant fluids such that $\sum_i^N a_i = 2^d$, where (c) d is the depth of the mixing tree $\mathcal{T}$, and (d) M on-chip mixers. We assume that the number of reservoirs that are placed around the chip-boundary is greater than or equal to N.

The depth of the mixing tree (d) is predetermined according to the required accuracy of the desired *CF*s of the constituent fluids in the target mixture. For a mixing tree, its depth d indicates that the maximum error in *CF* of any component fluid in the target mixture is $\frac{1}{2^{d+1}}$.

ALGORITHM 4: *RAMP*$(\langle a_1 : a_2 : \ldots : a_N \rangle, d, M)$

1: Compute $L = \sum_i^N a_i$.
2: Determine mixing tree $\mathcal{T}(P \cup Q, E)$ by a mixing algorithm, where $\forall q \in Q,\ f(q) : x_i \rightarrow R_i$.
3: Compute $T_{ms} = |P|$.
4: Set $T_{lb} = d$ and compute M_{lb} as provided in [Luo and Akella 2011].
5: **if** M is not known **then**
6: Set $M = M_{lb}$.
7: **end if**
8: Schedule $\mathcal{T}$ by *OSM* [Luo and Akella 2011] with M mixers to obtain $\mathcal{T}_{sch}(P \cup Q, E)$, so that each node p, $p \in P$, has a pair $\{time(p), mixer(p)\}$ for time stamp and mixer assignments.
9: Compute $T_c = \max(time(p)), \forall p \in P$.
10: Set initial orders as $\Pi_r^i : R_1, R_2, \ldots, R_N$ and $\Pi_m^i : M_1, M_2, \ldots, M_M$.
11: Obtain initial embedding of G by *GetBipartiteGraph*$(\mathcal{T}_{sch})$ and form the table of reservoir-to-mixer workloads.
12: Compute the initial crossing number X_i in G.
13: Apply Barycenter heuristic on G to obtain an intermediate embedding of G (ϕ_{interm}) with the intermediate orders Π_r and Π_m and the reduced crossing number $X_{initial}$.
14: Compute the weighted crossing number X_w in ϕ_{interm} using the workload table.
15: If $X_w > X_{initial}$, run a procedure for SA-based weighted crossing minimization with the initial embedding ϕ_{interm} and two bounds $X_{initial}$ and X_w.
16: Obtain final resource-allocation from the final embedding of G, i.e., $\phi_{opt} : (\Pi_r^f, \Pi_m^f)$ and the optimal weighted crossing number X_{opt}.

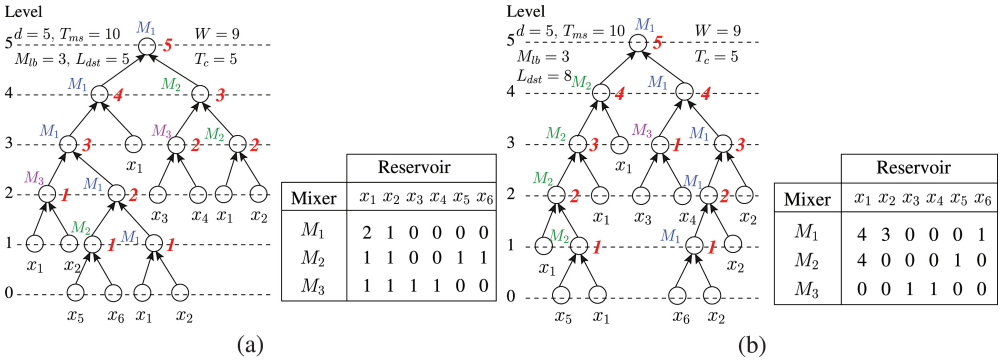
Outputs: (a) A scheduled mixing tree $\mathcal{T}_{sch}$, and (b) an optimal resource-allocation ϕ_{opt} for the mixture-preparation biochip.

A scheduled mixing tree $\mathcal{T}_{sch}(P \cup Q, E)$ has a set of nonleaf nodes (P), a set of leaf nodes (Q) and a set of directed edges (E) between two nodes. Each node q, $q \in Q$, indicates a mapping of fluid x_i loaded into the reservoir R_j (i.e., the mapping $x_i \rightarrow R_j$) denoted by the function $x_i = fluid(q)$. Whereas, each node p, $p \in P$, has an assigned time stamp t denoted by the function $t = time(p)$ along with an allocated mixer M_j denoted by the function $M_j = mixer(p)$. The time of completion (T_c) for a scheduled mixing tree is computed as $T_c = \max(time(p)), \forall p \in P$. However, as the problem of minimizing the number of edge-crossings in a bipartite graph is *NP-complete* [Garey and Johnson 1979], we propose the following heuristic algorithm to solve this resource-allocation problem.

5.3. Proposed Algorithm: *RAMP*

In order to obtain a solution of resource-allocation problem, for an application-specific mixture-preparation biochip, we propose an algorithm called *Routing-Aware* resource-allocation for *Mixture Preparation* (*RAMP*). The number of on-chip mixers (M) may be either predetermined or restricted to the minimum number of mixers for minimum-time completion of mixing tree (M_{lb}). The pseudo-code for the proposed algorithm is written as Algorithm 4. It first uses the Barycenter heuristic [Sugiyama et al. 1981] to reduce the crossing number in a bipartite graph. Then it uses a technique based on simulated annealing (SA) [Kirkpatrick et al. 1983] to minimize the weighted crossing number in the bipartite graph. The steps of *RAMP* are discussed in the following sections.

5.3.1. Determining the Mixing Tree for a Target Ratio. Given a target ratio, any mixing algorithm such as *MinMix* [Thies et al. 2008], *RSM* [Hsieh et al. 2012a] or *RMA*, can be used to determine the mixing tree. In a mixing tree $\mathcal{T}$, the (leaf and nonleaf) nodes are at different levels 0 to d. The root of $\mathcal{T}$ is at level d and the level of any other node is one less than the level of its parent node. Let T_{ms} be the total number of nonleaf nodes



Mixer	Reservoir					
	x_1	x_2	x_3	x_4	x_5	x_6
M_1	2	1	0	0	0	0
M_2	1	1	0	0	1	1
M_3	1	1	1	1	0	0

Mixer	Reservoir					
	x_1	x_2	x_3	x_4	x_5	x_6
M_1	4	3	0	0	0	1
M_2	4	0	0	0	1	0
M_3	0	0	1	1	0	0

Fig. 15. For target ratio 15:7:4:4:1:1 (a) *MinMix*-tree and (b) *RMA*-tree, both optimally scheduled by *OSM* with $M = M_{lb}$.

(mix-split cycles) in $\mathcal{T}$. For an example target ratio $15 : 7 : 4 : 4 : 1 : 1$, the *MinMix*-tree and *RMA*-tree are shown in Figures 15(a) and 15(b), respectively.

5.3.2. Scheduling a Mixing Tree. We adopt the *OSM* scheme [Luo and Akella 2011] for scheduling the mixing tree (T) with a given number (M) of mixers. This minimizes the total time (T_c) needed for mixing operations, and produces a scheduled-tree $\mathcal{T}_{sch}$ that indicates a sequence of mix-split steps with time-stamps and mixer allocation to each nonleaf node of $\mathcal{T}_{sch}$. If only one mixer is available (i.e., for $M = 1$), one needs $T_c = T_{ms}$. As aforementioned, M_{lb} is the minimum number of mixers required to complete mixing in minimum time (denoted by T_{lb} time stamps). Hence, T_{lb} is the depth of the mixing tree, that is, $T_c = T_{lb} = d$.

Let M^* and M' be the values of M_{lb} for *MinMix*-tree [Thies et al. 2008] and *RMA*-tree of a target ratio, respectively, where M' may be greater than M^*. For the example target ratio $15 : 7 : 4 : 4 : 1 : 1$, we found $M_{lb} = 3$ for both *MinMix*-tree and *RMA*-tree with $T_{lb} = 5$ as shown in Figures 15(a) and 15(b), respectively. In the scheduled *MinMix*-tree and *RMA*-tree with three mixers (Figure 15), each nonleaf node is assigned with a mixer M_i ($i = 1, 2$ *or* 3) and a time stamp t ($t = 1, 2, 3, 4$ *or* 5). The corresponding tables of reservoir-to-mixer workload for both the schedules are also shown in Figure 15.

5.3.3. Initial Resource Allocation. An application-specific mixture-preparation biochip consists of on-chip mixers and fluid-reservoirs, which are placed at fixed locations around the boundary of the chip. The initial resource-allocation is modeled and determined by *GetBipartiteGraph* (Algorithm 5) from the scheduled mixing tree $\mathcal{T}_{sch}$ in accordance to a bipartite graph $G(X, Y)$. In $G(X, Y)$, the set of vertices X is partitioned into two disjoint sets of vertices that represent the set of mixers (X_m), and the set of reservoirs (X_r). The set of directed edges Y denotes the droplet-transportation routes from the fluid-reservoirs to mixers, and for each directed edge $(v_1, v_2) \in Y$, $v_1 \in X_r$ and $v_2 \in X_m$. For each directed edge an integer weight is assigned which indicates the corresponding reservoir-to-mixer workload. The table of reservoir-to-mixer workloads is computed while constructing $G(X, Y)$. For the initial embedding of $G(X, Y)$, the initial crossing number (X_i) is computed and the weighted crossing number (X_w) is computed using the workload table. The set of vertices X_r (X_m) of $G(X, Y)$ has an associated order Π_r^i (Π_m^i) of reservoirs for the input fluids (mixers). The initial orders of reservoirs and mixers are set as $\Pi_r^i : R_1, R_2, \ldots, R_N$ and $\Pi_m^i : M_1, M_2, \ldots, M_M$. The initial allocations for the scheduled *MinMix*-tree and *RMA*-tree are depicted in Figures 16(a) and 16(c), respectively.

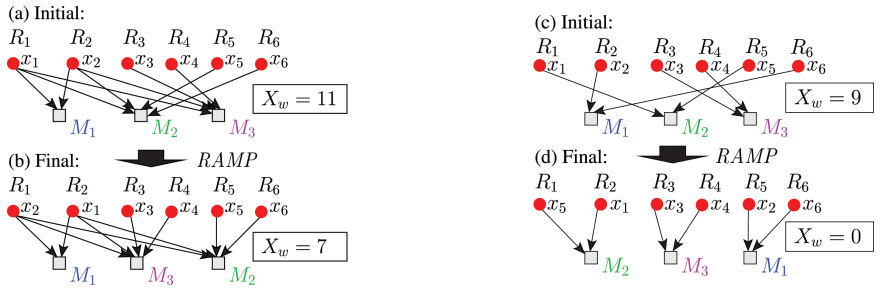


Fig. 16. Target ratio 15:7:4:4:1:1. (a) Initial and (b) final bipartite graphs for the scheduled *MinMix*-tree. (c) Initial and (d) final bipartite graphs for the scheduled *RMA*-tree.

ALGORITHM 5: *GetBipartiteGraph*$\left(\mathcal{T}_{sch}(P \cup Q, E)\right)$

1: Initialize $G = (X, Y)$ as $X = X_r \cup X_m$ and $Y = \phi$, where $X_r = \{R_1, R_2, \ldots, R_N\}$ and $X_m = \{M_1, M_2, \ldots, M_M\}$.
2: Initialize all the edge weights in the workload table as zero.
3: **for** each $p \in P$ **do**
4: **if** left-child$(p) = q, q \in Q$ **then**
5: $x_i = fluid(q), Y = Y \cup (\overrightarrow{R_i, mixer(p)})$. /* add a directed edge */
6: Update edge weight in the workload table as $w(\overrightarrow{R_i, mixer(p)}) = w(\overrightarrow{R_i, mixer(p)}) + 1$.
7: **end if**
8: **if** right-child$(p) = q, q \in Q$ **then**
9: $x_i = fluid(q), Y = Y \cup (\overrightarrow{R_i, mixer(p)})$. /* add a directed edge */
10: Update edge weight in the workload table as $w(\overrightarrow{R_i, mixer(p)}) = w(\overrightarrow{R_i, mixer(p)}) + 1$.
11: **end if**
12: **end for**
13: **return** G.

5.3.4. Routing-Aware Resource Allocation. As discussed earlier, we first adopt the Barycenter heuristic [Çakroğlu et al. 2007; Sugiyama et al. 1981] to minimize the crossing number X in the final embedding of the bipartite graph G. As a result, the linear ordering of the reservoirs and mixers are changed. Since the physical locations of the reservoirs and mixers are known, we permute the reservoirs (or the mixers) to further improve the total droplet-transportation distance Z without increasing X_w. For the scheduled *MinMix*-tree and *RMA*-tree of Figure 15, the bipartite graphs corresponding to the final solution are shown in Figures 16(b) and 16(d), respectively. It is observed that the number of crossings in the corresponding bipartite graph is reduced from 11 to 7 for the *MinMix*-tree and from 9 to 0 for the *RMA*-tree.

Some snapshots of Barycenter heuristic that transform the initial bipartite graph to a final solution are from the initial resource-allocation shown in Figure 17. In each iteration, it computes the row-Barycenter, BC(row), and column-Barycenter, BC(col), values for all the rows and columns of the matrix corresponding to the bipartite graph. Then, according to the sorted BC(row) or BC(col) values the rows or columns in the matrix are permuted to obtain a new matrix. Every time the crossings (X) is determined for some resource allocation and the procedure is terminated as a stable value of X is observed over consecutive iterations. As a result, an intermediate embedding of G (ϕ_{interm}) is obtained with the intermediate orders Π_r and Π_m and the reduced crossing number $X_{initial}$.

Next, the weighted crossing number X_w in ϕ_{interm} is computed using the workload table. Note that X_w may be greater than or equal to $X_{initial}$. Finally, a simulated-annealing (SA) based technique [Kirkpatrick et al. 1983] is run on an initial embedding

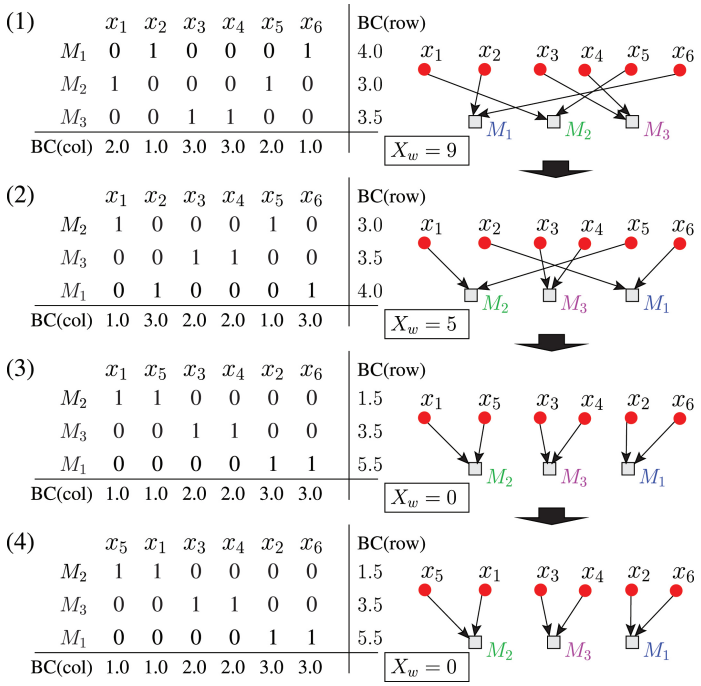


Fig. 17. Step-by-step application of Barycenter heuristic to the bipartite graph corresponding to the resource allocation of Figure 16(c).

of G to further minimize the weighted crossing number. The final embedding of G is either determined by the optimal embedding in SA-based technique (if $X_w > X_{initial}$) or by the intermediate embedding obtained after applying Barycenter heuristic in *RAMP* (if $X_w = X_{initial}$). The proposed algorithm *RAMP* denotes the final embedding of G as $\phi_{opt} : (\Pi_r^f, \Pi_m^f)$ and the optimal weighted crossing number is obtained as X_{opt}. We obtain the final resource-allocation from $\phi_{opt} : (\Pi_r^f, \Pi_m^f)$.

5.4. Discussions on *RAMP*

When *RAMP* is applied on the *RMA*-tree corresponding to the target ratio 15 : 7 : 4 : 4 : 1 : 1, the value of X_w is reduced to zero (final resource-allocation) from seven (initial resource-allocation) as shown in Figures 16(c) and 16(d). Figures 18(a) and 18(b) depict the droplet transportation routes at different time-stamps for the initial and the final placement solutions, respectively.

5.4.1. Benefits of RAMP. Given a mixing algorithm, *RAMP* can be used to improve its performance during implementation. As discussed earlier, *RAMP* provides a suitable resource-allocation that reduces droplet crossovers and transportation distances. Thus, it reduces electrode actuations and the area of the layout. A reduction in crossing number also reduces the number of stalls, which may be otherwise needed to satisfy fluidic constraints between two droplets. This in turn, reduces the time of completion of mixture-preparation. Further, when a multi-target mixing algorithm such as *RSM* [Hsieh et al. 2012a] is used to prepare two or more target ratios concurrently on a chip, a droplet-crossover may be a potential source of contamination. Therefore, the additional overhead of interleaving wash droplets [Zhao and Chakrabarty 2010]

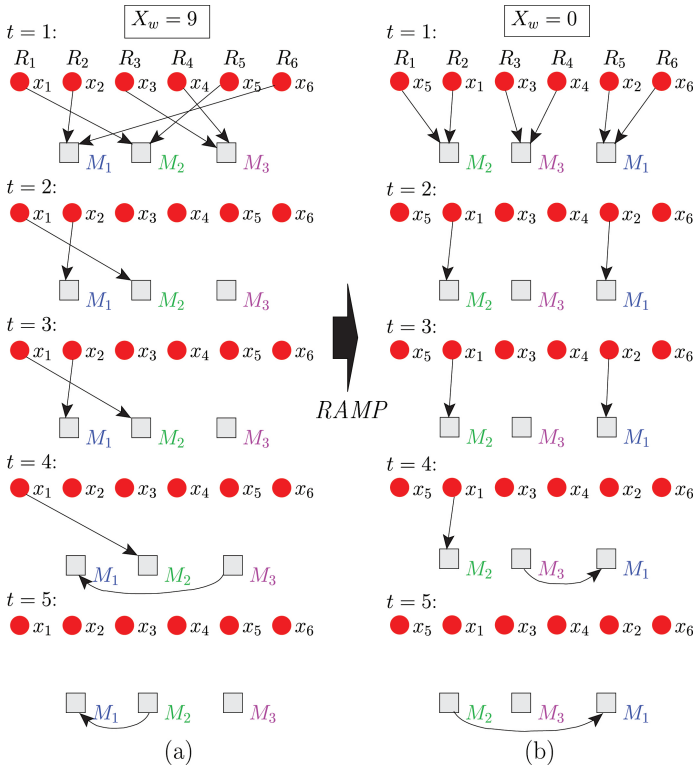


Fig. 18. Cycle-wise droplet transportation routes for the scheduled *RMA*-tree (shown in Figure 15b) in case of (a) initial and (b) final resource-allocation.

will badly impact the overall completion time. Thus, *RAMP*, which aims at minimizing crossovers, will also be useful for the design of such chips.

Figure 19 shows a DMF biochip layout with 13 boundary dispensers, two waste reservoirs, and four on-chip mixers. This layout can be used for mixture-preparation of all ten target ratios listed in Table I. For a given ratio, we consider the mixing trees obtained by *MinMix* and *RMA*. The corresponding droplet-pathways and workloads are determined. In each case, the placement solution obtained by *RAMP* is compared with an initial baseline placement. The results are shown in Table II.

5.5. Simulation Results of *RAMP*

We have carried out simulation experiments with a large number of target ratios to evaluate *RAMP*. The data-set is generated by an integer-partitioning technique as described in the Appendix. For some example ratios, comparative results on X_w obtained by *MinMix* [Thies et al. 2008] and *RMA* are presented in Table II. For some example ratios, comparative results of Z in *RAMP* with *MinMix* [Thies et al. 2008] and *RMA* are presented in Table III.

We simulate *RAMP* for 6058 synthetic target ratios with $L = 32$ (i.e., $d = 5$) of N different fluids, where $3 \leq N \leq 12$. For all these 6058 target ratios, *RAMP* is applied to both *MinMix*-trees and *RMA*-trees with $M = M^*$ number of on-chip mixers (where M^* is M_{lb} of the *MinMix*-tree). Figures 20(a) and 20(c) show that, for both the mixing algorithms, an application of *RAMP* causes a shift in the distribution of X_w towards the origin. Such shifts indicate that the weighted crossings number among droplet

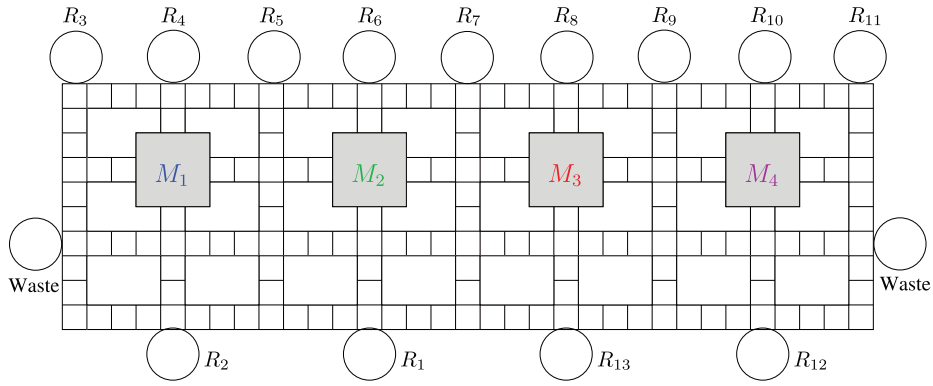


Fig. 19. A typical DMF biochip layout for mixture preparation with 13 boundary reservoirs fluids and four on-chip mixers.

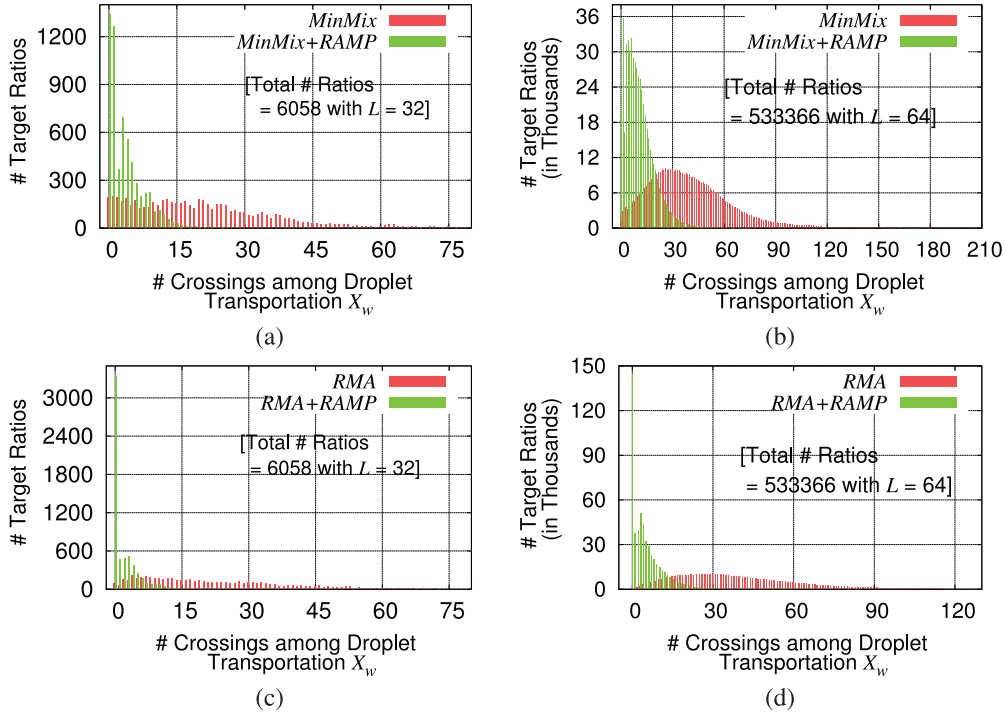


Fig. 20. Histograms for improvements in the number of crossings by *RAMP* over *MinMix* for (a) 6058 target ratios with $L = 32$ and (b) 533366 target ratios with $L = 64$, and over *RMA* for (c) 6058 target ratios with $L = 32$ and (d) 533366 target ratios with $L = 64$, when the mixing trees are scheduled with $M = M^*$ mixers.

pathways is reduced by *RAMP*. Hence, for a comparatively large number of target ratios, we obtained the final resource-allocation with a reduced X_w value compared to initial resource-allocation. Similarly, for 533366 synthetic target ratios with $L = 64$ (i.e., $d = 6$) of N different fluids, where $3 \leq N \leq 12$, *RAMP* can reduce X_w in final resource-allocation as shown in Figures 20(b) and 20(d).

We have run *RAMP* on the mixing algorithm *MTCS* [Kumar et al. 2013] for 6058 synthetic target ratios with $L = 32$ (i.e., $d = 5$), and for 533366 synthetic target

Table II. Comparative Results of *RAMP* with *MinMix* and *RMA* for Some Randomly Chosen Target Ratios[†]

Ex.	Target Ratio	X_w				(N, M)	Best Scheduled Tree ($\mathcal{T}_{sch}$), Order of Reservoirs (Π_m^f), Order of Mixers (Π_m^f)
		MinMix	*MinMix* + *RAMP*	*RMA* *RAMP*	*RMA* +		
1.	40:10:1:1:48 (≡ 51:13:1:1:62)	1	1	0	0	(5, 2)	*RMA*-tree, (R_2, R_1, R_3, R_4, R_5), (M_1, M_2)
2.	13:12:5:2	5	1	3	0	(4, 3)	*RMA*-tree, (R_2, R_3, R_4, R_1), (M_2, M_1)
3.	15:7:4:4:1:1	11	7	9	0	(6, 3)	*RMA*-tree, ($R_5, R_1, R_3, R_4, R_2, R_6$), ($M_2, M_3, M_1$)
4.	2:3:5:7:11:13:87	50	16	27	2	(7, 4)	*RMA*-tree, ($R_2, R_6, R_3, R_5, R_7, R_1, R_4$), ($M_2, M_1, M_4, M_3$)
5.	341:341:342	1	1	4	0	(3, 2)	*RMA*-tree, (R_2, R_1, R_3), (M_1, M_2)
6.	12:7:7:3:3	11	1	7	4	(5, 3)	*MinMix*-tree, (R_3, R_1, R_2, R_4, R_5), (M_2, M_1, M_3)
7.	18:5:3:3:3	15	4	8	2	(5, 3)	*RMA*-tree, (R_4, R_1, R_3, R_2, R_5), (M_1, M_3, M_2)
8.	7:14:11	2	1	4	0	(3, 2)	*RMA*-tree, (R_2, R_1, R_3), (M_2, M_1)
9.	9:8:8:8:7:7:5:5:4:1:1:1	42	22	35	1	(12, 4)	*RMA*-tree, ($R_3, R_4, R_6, R_{10}, R_1, R_5, R_7, R_9, R_{11}, R_2, R_{12}, R_8$), ($M_4, M_1, M_2, M_3$)
10.	34:7:5:5:5:4:3:1	16	12	21	0	(8, 4)	*RMA*-tree, ($R_3, R_1, R_8, R_2, R_6, R_5, R_7, R_4$), ($M_1, M_4, M_3, M_2$)

[†]In order to produce two droplets of a target ratio, X_w: total weighted crossings in the corresponding (single-layer) bipartite graph of scheduled mixing tree, N: number of reservoirs, M: number of mixers available on-chip for scheduling the mixing tree.

ratios with $L = 64$ (i.e., $d = 6$). Figures 21(a) and 21(b) show that *RAMP* causes a shift of X_w towards the origin for $L = 32$ and $L = 64$, respectively. A comparison of these distribution patterns for *MinMix*, *RMA* and *MTCS*, suggests that *RMA* offers the best reduction in droplet-crossing number among them, when implemented on a chip.

We simulate *MinMix* and *RMA* over 6058 target ratios with $L = 32$ and after scheduling the mixing trees with varying number of mixers (M), we estimate the (μ, σ)-pair for the distributions of the initial weighted crossings (X_w). Next, *RAMP* is applied with *MinMix* and *RMA* for the same target set and estimate the (μ, σ)-pair for the distributions of the final weighted crossings (X_w). Then, we compute the average percentage improvements in the number of final edge-crossings by *RAMP* over *MinMix* and *RMA* for varying number of mixers (M). The number of on-chip mixers (M) is chosen as M_{lb} of the *MinMix*-tree or *RMA*-tree. Table IV shows the results on (μ, σ)-pairs and average percentage improvements of X_w obtained by *RAMP*. On the average, *RAMP* provides

Table III. Comparative Results of Total Weighted-Transportation-Distance (Z) in *RAMP* with *MinMix* and *RMA* on Some Example Ratios[†]

Ex.	Target Ratio	*MinMix*	*MinMix + RAMP*	*RMA*	*RMA + RAMP*
1.	40:10:1:1:48 (≡ 51:13:1:1:62)	108	84	78	64
2.	13:12:5:2	68	52	60	36
3.	15:7:4:4:1:1	118	90	110	42
4.	2:3:5:7:11:13:87	302	164	268	104
5.	341:341:342	58	58	90	74
6.	12:7:7:3:3	108	52	82	62
7.	18:5:3:3:3	104	64	110	50
8.	7:14:11	46	46	62	30

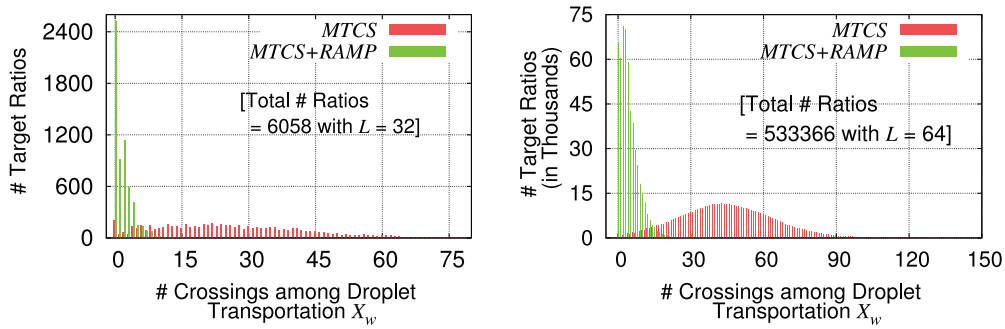


Fig. 21. Histograms for improvements in the number of crossings by *RAMP* over *MTCS* for (a) 6058 target ratios with $L = 32$ and (b) 533366 target ratios with $L = 64$, when the mixing trees are scheduled with $M = M^*$ mixers.

Table IV.
(μ, σ)-pair of distributions and average % improvements by *RAMP* in # weighted crossings (X_w) over 6058 target ratios for $L = 32$.

# Mixers	*MinMix*	*MinMix+RAMP*		*RMA*	*RMA+RAMP*	
M	(μ, σ)	(μ, σ)	*% impr.*	(μ, σ)	(μ, σ)	*% impr.*
2	(10.4,6.5)	(1.5,2.4)	77.3%	(14.2,8.5)	(2.2,4.2)	85.4%
3	(18.6,12.7)	(3.4,4.1)	75.1%	(20.6,14.1)	(3.6,5.7)	86.3%
4	(20.0,13.9)	(3.7,4.2)	74.5%	(22.0,16.6)	(2.5,4.7)	90.6%
5	(20.7,15.6)	(3.6,3.9)	75.2%	(22.9,17.7)	(2.2,4.1)	91.2%
6	(20.8,16.1)	(3.6,4.0)	75.2%	(23.0,17.9)	(2.0,3.3)	91.7%
7	(20.9,16.2)	(3.6,4.0)	75.2%	(22.9,17.8)	(1.9,3.1)	91.8%
M_{lb}	(20.9,16.2)	(3.6,4.0)	75.2%	(22.9,17.7)	(1.9,3.0)	91.8%
M^*	(20.9,16.2)	(3.6,4.0)	75.2%	(23.2,17.6)	(2.7,4.4)	88.9%
Average % improvement			**75.4%**			**89.7%**

75.4% (89.7%) improvement in edge crossing-number compared to a baseline solution, when applied on *MinMix* (*RMA*).

Table V shows the comparative performance of *MinMix*, *RMA*, and *MTCS* when *RAMP* is used for allocating resources. The table presents the mean (μ) and standard deviation (σ) of the distributions of X_w for all three mixing algorithms over 6058 target ratios with $L = 32$. The average percentage improvements in X_w by *RAMP* for three mixing algorithms are also reported in the table. Note that for *RAMP* provides maximum improvement when *RMA* is used.

Table V.
Mean (μ) and standard deviation (σ) of the distributions and average % improvement by *RAMP* in weighted crossings (X_w) for mixing trees obtained by *MinMix*, *RMA* and *MTCS* over 6058 target ratios with $L = 32$.

# Crossings (X_w)	*MinMix*		*RMA*		*MTCS*	
	−RAMP	*+RAMP*	*−RAMP*	*+RAMP*	*−RAMP*	*+RAMP*
Mean (μ)	20.86	3.61	22.85	1.87	25.39	1.62
Std. Dev. (σ)	16.17	3.96	17.69	3.00	15.52	1.99
Avg. % Improvement	—	**75.18%**	—	**91.77%**	—	**89.67%**

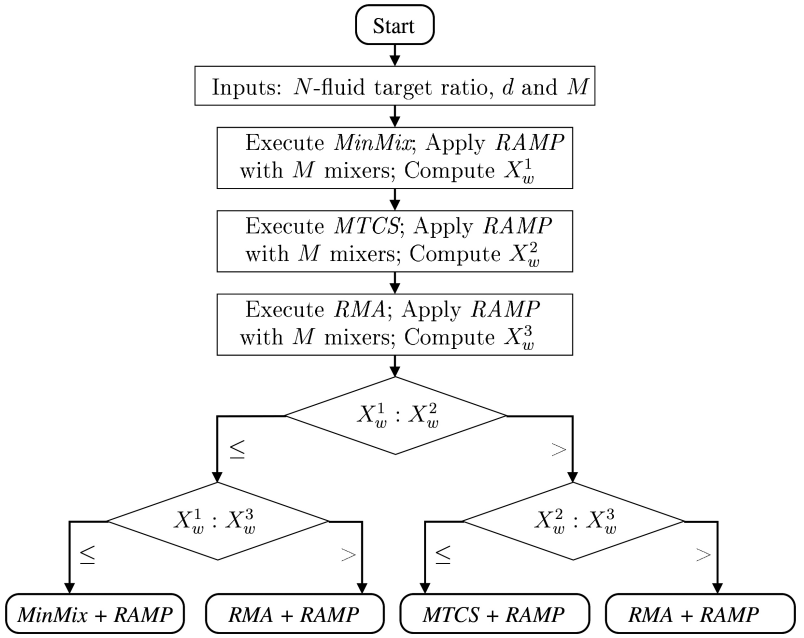


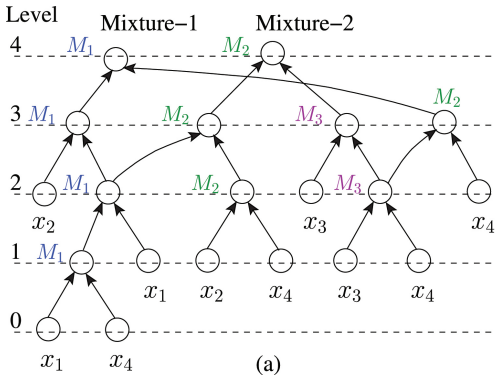
Fig. 22. Flowchart of using *RAMP* with *MinMix*, *MTCS* and *RMA* to determine scheduled mixing tree and on-chip resource-allocation for mixture preparation.

6. INTEGRATION OF *RAMP* WITH *MINMIX*, *MTCS* AND *RMA*

We have observed that though *RMA* provides the best solution for a large number of ratios from the viewpoint of resource allocation, *MinMix* [Thies et al. 2008] and *MTCS* [Kumar et al. 2013] provides a better solution in some cases as well. Also, *MinMix* always provides the minimum number of mix-split steps. Hence, for the purpose of optimization, given a target ratio and a number of on-chip mixers (M), we can choose either *MinMix*, *MTCS* or *RMA* that provides the best solution. The flowchart shown in Figure 22 describes a tool that serves this purpose.

7. APPLICATION OF *RAMP* TO MULTI-TARGET MIXTURE PREPARATION

We have already demonstrated the impact of the proposed *RAMP* algorithm on droplet crossover and transportation distance considering mixing trees. In this section, we show that *RAMP* is equally applicable to mixing graphs as well, for example to those obtained by *RSM* algorithm [Hsieh et al. 2012a], which is used to produce multiple target ratios of three or more reactants on a biochip. When *RAMP* is run on a multiple-target mixing graph, an improvement in performance can be observed as the weighted-crossing-number reduces, which, in turn, reduces the transportation-distance and



$d = 4, T_{ms} = 10, W = 6$

$M_{lb} = 3, M = 3, T_c = 4$

	Reservoir			
Mixer	x_1	x_2	x_3	x_4
M_1	2	1	0	1
M_2	0	1	0	2
M_3	0	0	2	1

(b)

Fig. 23. For two target ratios 3 : 4 : 2 : 7 (Mixture-1) and 3 : 2 : 6 : 5 (Mixture-2), (a) scheduled mixing graph obtained by *RSM* and (b) reservoir-to-mixer transportation workload table.

cross-contamination. Note that in the case of a single target ratio, all droplets that are used in the mixing process, will finally lead to the production of the same target. Hence, any contamination among crossing droplets will not affect the outcome. However, when multiple target ratios are produced simultaneously based on a composite mixing graph as in *RSM* algorithm, the issue of cross-contamination among crossing routing paths indeed becomes important, and the presence of such crossing paths may cause additional overhead in the completion time because of the need for interleaving wash droplets [Zhao and Chakrabarty 2010]. Thus, a reduction in crossing number also tends to reduce of possible cross-contamination instances. We illustrate this problem and its solution using the following example.

Consider an example of generating two target ratios 3 : 4 : 2 : 7 and 3 : 2 : 6 : 5 with four reactants x_1, x_2, x_3, x_4 as studied in Hsieh et al. [2012a]. The mixing graph obtained by *RSM* algorithm is shown in Figure 23(a). We load the reactants x_1, x_2, x_3, x_4 to four dispensers R_1, R_2, R_3, R_4 of a chip respectively, which are linearly ordered on the boundary of the chip. Assume that three mixers M_1, M_2, M_3 are available, which are assigned to the mixing nodes of the graph as in Figure 23(a). The corresponding reservoir-to-mixer transportation workload is shown in Figure 23(b). The initial bipartite graph that represents the scenario is shown in Figure 24(a). Note that in this graph, the weighted crossing number (X_w) is four. An application of *RAMP* modifies the graph as shown in Figure 24(b), where $X_w = 1$. Also, because of the repositioning of reservoirs, the weighted-transportation-distance reduces. This observation is also apparent from the workload table in Figure 23(b). It can be easily verified that some crossing pairs of edges as shown in Figure 24(a), refer to two droplet paths, which are involved in the production of two different target ratios. Thus, such a crossover mandates a need for cross-contamination washing, as a droplet may pick up the residue left by an earlier one, thereby causing an error in a target ratio. On the other hand, in the final bipartite graph of Figure 24(b) as obtained by *RAMP*, the weighted-crossing number reduces from four to one, which, in turn, reduces the wash-load.

8. CONCLUSIONS

In this article, we have proposed an efficient mixing algorithm for automated mixture preparation of three or more reactant fluids, which is based on producing a mixing tree that includes taller and almost-disjoint dilution subtrees. This algorithm leads to a layout-friendly implementation that reduces the number of droplet-crossovers

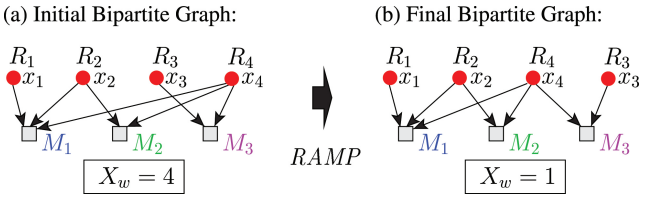


Fig. 24. For two target ratios 3 : 4 : 2 : 7 and 3 : 2 : 6 : 5, (a) initial and (b) final bipartite graphs for the scheduled *RSM*-graph.

and transportation distance among the reservoirs and mixers. Next, we have proposed a routing-aware resource-allocation scheme (*RAMP*) for determining a suitable placement of resources. This can be used to improve the performance of any mixing algorithm. Simulation results show that *RAMP* provides 75.4% improvement in the number of droplet-path crossovers compared to a baseline solution when *MinMix* is used, and the improvement rises to 89.7% when *RMA* is used. It also improves the performance a multiple-target mixing algorithm such as *RSM* [Hsieh et al. 2012a].

We have used a bipartite graph model in our problem formulation, and we have approximated the arrangement of reservoirs or mixers as a linear ordering. In practice, the reservoirs may be placed in a circular arrangement around the chip boundary, and the mixers and other on-chip resources may be located or instantiated in random positions. Also, sample-preparation may be just a preprocessing step; the chip may be used for implementing other complex assays. In order to model these cases, a different kind of graph representation and formulation will be needed. The optimization of resource allocation and placement in such an environment is left as an open problem to study.

APPENDIX: Generating Synthetic Test Cases of Large Number of Target Ratios

For evaluating the proposed scheme related to mixture preparation, we have carried out simulation experiments on a test data set of a large number of target ratios. In number theory and combinatorics, integer partitioning is a way of writing an integer as a sum of positive integers, regardless of their order [Zoghbi and Stojmenović 1998]. By convention, the partitions are usually ordered from the largest to the smallest, for instances, 4 can be partitioned in five distinct ways: 4, 3+1, 2+2, 2+1+1, 1+1+1+1. We use different distinct partitions of the ratio-sum L (where L is the sum of ratio integers) as the target ratios of N fluids after keeping only N-component partitions for N fluids, where $N > 2$.

In the case of dilution (i.e., $N = 2$), a mixing tree can have only one branch and only one mixer can be assigned to execute mixing steps. Moreover, the dilution algorithms can be evaluated with the large data set of $(2^d - 1)$ target concentration factors (*CF*s) based on the accuracy level (d) of the desired target *CF*. For example, for an accuracy level of 10, the test data set contains 1023 *CF*s as follows: $\frac{1}{1024}, \frac{2}{1024}, \ldots, \frac{1023}{1024}$.

In order to evaluate mixing algorithms, while considering the target ratios of more than two fluids, we keep those partitions in which the integers are set-wise co-prime, so that two different ratios do not eventually turn into the same ratio. Hence, the target ratios {2:1:1} and {1:1:1:1} may be considered for $L = 4$. In real-life bioprotocols, it is observed that as many as 12 different fluids may need to be mixed to prepare a target mixture that may be used in a DMF biochip [Ananthanarayanan and Thies 2010]. Hence, the ratio-sum L is partitioned into three to twelve components.

In this thesis, we deal with the (1 : 1) mixing model for automated sample preparation. Therefore, the value of the ratio-sum (L) increases from $L = 4$ to $L = 64$, as the

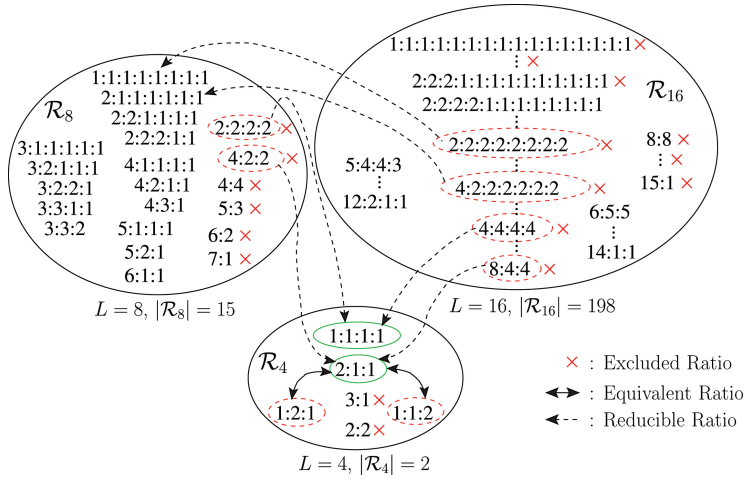


Fig. 25. A snapshot of generating test data set of large number of target ratios (using a number partitioning based technique).

accuracy level (d) of the desired *CF* increases from 2 to 6 (since $L = 2^d$). It is found that the data set of synthetic ratios contains two distinct partitions, if $L = 4$ (i.e., $d = 2$). Hence, the size of the corresponding data set, $|\mathcal{R}_4|$ is 2. If $L = 8$, 16, 32 or 64, then $|\mathcal{R}_8| = 15$, $|\mathcal{R}_{16}| = 198$, $|\mathcal{R}_{32}| = 6058$ or $|\mathcal{R}_{64}| = 533366$, respectively.

In the data set of synthetic ratios, two ratios are said to be *equivalent*, if they represent two different permutations of the same integers. For example, the ratio {2:1:1} is equivalent to the ratio {1:1:2} and {1:2:1}. A ratio is *reducible* to another ratio, if one of them can be obtained from the other by factoring out an integer. One ratio is *excluded* from the test data set, if it is reducible to another, or it is not within the domain of our assumption $3 \leq N \leq 12$). For example, the ratio {8:4:4} is reducible to the ratio {2:1:1} and hence the ratio {8:4:4} is excluded from the data set $|\mathcal{R}_{16}|$. Similarly, the ratio {2:2:2:2} is reducible to the ratio {1:1:1:1} and hence the ratio {2:2:2:2} is excluded from the data set $|\mathcal{R}_8|$. A ratio {7:1} is excluded from the data set $|\mathcal{R}_8|$, as the number of reactant fluids is 2 (not within the domain of $3 \leq N \leq 12$). Similarly, another ratio {1:1:1:1:1:1:1:1:1:1:1:1:1:1:1:1} is excluded from the data set $|\mathcal{R}_{16}|$, as the number of reactant fluids is 16 (not within the domain of $3 \leq N \leq 12$). A portion of the data set for a large number of synthetic ratios is depicted in Figure 25, which shows how some ratios are excluded because of the equivalence of two ratios and the reducibility of one ratio to another.

ACKNOWLEDGMENTS

The authors wish to thank the Associate Editor, Dr. Hai Li, and the anonymous reviewers for their critical comments and valuable suggestions.

REFERENCES

Mohamed Abdelgawad and Aaron R. Wheeler. 2009. The digital revolution: A new paradigm for microfluidics. *Advanced Materials* 21, 8, 920–925.

Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. 1974. *The Design and Analysis of Computer Algorithms*. Addison-Wesley.

Vaishnavi Ananthanarayanan and William Thies. 2010. Biocoder: A programming language for standardizing and automating biology protocols. *J. Biol. Eng.* 4, 1–13.

Sukanta Bhattacharjee, Ansuman Banerjee, and Bhargab B. Bhattacharya. 2012. Multiple dilution sample preparation using digital microfluidic biochips. In *Proceedings of the International Symposium on Electronic System Design*. 188–192.

Sukanta Bhattacharjee, Ansuman Banerjee, and Bhargab B. Bhattacharya. 2014. Sample preparation with multiple dilutions on digital microfluidic biochips. *IET Comput. Digital Tech.* 8, 1 49–58.

Christoph Buchheim, Markus Chimani, Dietmar Ebner, Carsten Gutwenger, Michael Jünger, GunnarW. Klau, Petra Mutzel, and René Weiskircher. 2008. A branch-and-cut approach to the crossing number problem. *Discrete Optim.* 5, 2, 373–388.

Christoph Buchheim, Markus Chimani, Carsten Gutwenger, Michael Jünger, and Petra Mutzel. 2013. Chapter 2: Crossings and planarization. In *Handbook of Graph Drawing and Visualization*, 43–85.

Olca A. Çakroğlu, Cesim Erten, Omer Karataş, and Melih Sozdinler. 2007. Crossing Minimization in Weighted Bipartite Graphs. In *Proceedings of the International Conference on Experimental Algorithms*, Lecture Notes in Computer Science, vol. 4525. 122–135.

Krishnendu Chakrabarty and Fei Su. 2007. *Digital Microfluidic Biochips: Synthesis, Testing and Reconfiguration Techniques*. CRC Press.

Krishnendu Chakrabarty and Tao Xu. 2010. *Digital Microfluidic Biochips: Design and Optimization*. CRC Press.

Debalina Chatterjee, Boonta Hetayothin, Aaron R. Wheeler, Daniel J. King, and Robin L. Garrell. 2006. Droplet-based microfluidics with nonaqueous solvents and solutions. *Lab Chip* 6, 2, 199–206.

Ting-Wei Chiang, Chia-Hung Liu, and Juinn-Dar Huang. 2013. Graph-based optimal reactant minimization for sample preparation on digital microfluidic biochips. In *Proceedings of the International Symposium on VLSI Design, Automation, and Test*. 1–4.

Trung Anh Dinh, Shigeru Yamashita, and Tsung-Yi Ho. 2014. A network-flow-based optimal sample preparation algorithm for digital microfluidic biochips. In *Proceedings of the Asia and South Pacific Design Automation Conference*. 225–230.

Michael R. Garey and David S. Johnson. 1979. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company.

Eric J. Griffith, Srinivas Akella, and Mark K. Goldberg. 2006. Performance characterization of a reconfigurable planar-array digital microfluidic system. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 25, 2, 345–357.

B. Hadwen, G. R. Broder, D. Morganti, A. Jacobs, C. Brown, J. R. Hector, Y. Kubota, and Hywel Morgan. 2012. Programmable large area digital microfluidic array with integrated droplet sensing for bioassays. *Lab Chip* 12, 18, 3305–3313.

Keith E. Herold and Avraham Rasooly. 2009. *Lab-on-a-Chip Technology* (Vol. 1)*: Fabrication and Microfluidics*. Caister Academic Press.

Yi-Ling Hsieh, Tsung-Yi Ho, and Krishnendu Chakrabarty. 2012a. A reagent-saving mixing algorithm for preparing multiple-target biochemical samples using digital microfluidics. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 31, 11, 1656–1669.

Yi-Ling Hsieh, Tsung-Yi Ho, and Krishnendu Chakrabarty. 2012b. Design methodology for sample preparation on digital microfluidic biochips. In *Proceedings of the IEEE International Conference on Computer Design*. 189–194.

Yi-Ling Hsieh, Tsung-Yi Ho, and Krishnendu Chakrabarty. 2014. Biochip synthesis and dynamic error recovery for sample preparation using digital microfluidics. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 33, 2, 183–196.

Juinn-Dar Huang, Chia-Hung Liu, and Ting-Wei Chiang. 2012. Reactant minimization during sample preparation on digital microfluidic biochips using skewed mixing trees. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. 377–384.

Juinn-Dar Huang, Chia-Hung Liu, and Huei-Shan Lin. 2013. Reactant and waste minimization in multitarget sample preparation on digital microfluidic biochips. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 32, 10, 1484–1494.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. 1983. Optimization by simulated annealing. *Science* 220, 671–680.

Jon Kleinberg and Eva Tardos. 2005. *Algorithm Design*. Addison Wesley.

Srijan Kumar, Sudip Roy, Partha P. Chakrabarti, Bhargab B. Bhattacharya, and Krishnendu Chakrabarty. 2013. Efficient mixture preparation on digital microfluidic biochips. In *Proceedings of the IEEE International Symposium on Design and Diagnostics of Electronic Circuits Systems*. 205–210.

Chia-Hung Liu, Hao-Han Chang, Tung-Che Liang, and Juinn-Dar Huang. 2013. Sample preparation for many-reactant bioassay on dmfbs using common dilution operation sharing. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. 615–621.

Lingzhi Luo and Srinivas Akella. 2011. Optimal scheduling of biochemical analyses on digital microfluidic systems. *IEEE Trans. Autom. Sci. Eng.* 8, 1, 216–227.

Elizabeth M. Miller and Aaron R. Wheeler. 2009. Digital bioanalysis. *Springer J. Analytical Bioanalytical Chem.* 393, 2, 419–426.

Debasis Mitra, Sudip Roy, Sukanta Bhattacharjee, Krishnendu Chakrabarty, and Bhargab B. Bhattacharya. 2014. On-chip sample preparation for multiple targets using digital microfluidics. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 33, 8, 1131–1144.

Debasis Mitra, Sudip Roy, Krishnendu Chakrabarty, and Bhargab B. Bhattacharya. 2012. On-chip sample preparation with multiple dilutions using digital microfluidics. In *Proceedings of the IEEE Annual Symposium on VLSI*. 314–319.

Philip Y. Paik, Vamsee K. Pamula, and Richard B. Fair. 2003a. Rapid droplet mixers for digital microfluidic systems. *Lab Chip* 3, 4, 253–259.

Philip Y. Paik, Vamsee K. Pamula, Michael G. Pollack, and Richard B. Fair. 2003b. Electrowetting-based droplet mixers for microfluidic systems. *Lab Chip* 3, 1 28–33.

Hong Ren, Vijay Srinivasan, and Richard B. Fair. 2003. Design and testing of an interpolating mixing architecture for electrowetting-based droplet-on-chip chemical dilution. In *Proceedings of the International Solid-State Sensors, Actuators and Microsystems Conference*. 619–622.

R. Bruce Richter and Carsten Thomassen. 1997. Relations between crossing numbers of complete and complete bipartite graphs. *Am. Math. Monthly* 104, 2, 131–137.

Sudip Roy, Bhargab B. Bhattacharya, Partha P. Chakrabarti, and Krishnendu Chakrabarty. 2011b. Layout-aware solution preparation for biochemical analysis on a digital microfluidic biochip. In *Proceedings of the International Conference on VLSI Design*. 171–176.

Sudip Roy, Bhargab B. Bhattacharya, and Krishnendu Chakrabarty. 2010. Optimization of dilution and mixing of biochemical samples using digital microfluidic biochips. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 29, 11, 1696–1708.

Sudip Roy, Bhargab B. Bhattacharya, and Krishnendu Chakrabarty. 2011a. Waste-aware dilution and mixing of biochemical samples with digital microfluidic biochips. In *Proceedings of the Design, Automation Test in Europe Conference Exhibition*. 1059–1064.

Sudip Roy, Bhargab B. Bhattacharya, Sarmishtha Ghoshal, and Krishnendu Chakrabarty. 2014a. An optimal two-mixer dilution engine with digital microfluidics for low-power applications. *ASP J. Low Power Electronics* 10, 3, 1–12.

Sudip Roy, Bhargab B. Bhattacharya, Sarmishtha Ghoshal, and Krishnendu Chakrabarty. 2014b. High-throughput dilution engine for sample preparation on digital microfluidic biochips. *IET Comput. Digital Tech.* 8, 4, 163–171.

Sudip Roy, Bhargab B. Bhattacharya, Sarmishtha Ghoshal, and Krishnendu Chakrabarty. 2014c. Theory and analysis of generalized mixing and dilution of biochemical fluids using digital microfluidic biochips. *ACM J. Emerg. Technol. Comput. Syst.* 11, 1, 2.1–2.33.

Sudip Roy, Partha P. Chakrabarti, Srijan Kumar, Bhargab B. Bhattacharya, and Krishnendu Chakrabarty. 2013. Routing-aware resource allocation for mixture preparation in digital microfluidic biochips. In *Proceedings of the IEEE Annual Symposium on VLSI*. 1–6.

Marcus Schaefer. 2013. The graph crossing number and its variants: A survey. *Electronic J. Combinatorics: Dynamic Surveys*, 21.

Ramakrishna Sista, Zhishan Hua, Prasanna Thwar, Arjun Sudarsan, Vijay Srinivasan, Allen E. Eckhardt, Michael G. Pollack, and Vamsee K. Pamula. 2008. Development of a digital microfluidic platform for point of care testing. *Lab Chip* 8, 12, 2091–2104.

Fei Su and Krishnendu Chakrabarty. 2004. Architectural-level synthesis of digital microfluidics-based biochips. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. 223–228.

Fei Su and Krishnendu Chakrabarty. 2008. High-level synthesis of digital microfluidic biochips. *ACM J. Emerg. Technol. Comput. Syst.* 3, 4, 1–32.

Kozo Sugiyama, Shojiro Tagawa, and Mitsuhiko Toda. 1981. Methods for visual understanding of hierarchical system structures. *IEEE Trans. Syst. Man Cybern.* 11, 2, 109–125.

Roberto Tamassia. 2013. *Handbook of Graph Drawing and Visualization*. Chapman & Hall/CRC.

Hsih Yin Tan, Weng Keong Loke, Yong Teng Tan, and Nam-Trung Nguyen. 2008. A lab-on-a-chip for detection of nerve agent sarin in blood. *Lab Chip* 8, 6, 885–891.

William Thies, John Paul Urbanski, Todd Thorsen, and Saman Amarasinghe. 2008. Abstraction layers for scalable microfluidic biocomputing. *Natural Computing* 7, 2, 255–275.

Yang Zhao and Krishnendu Chakrabarty. 2009. Cross-contamination avoidance for droplet routing in digital microfluidic biochips. In *Proceedings of the Design, Automation Test in Europe Conference Exhibition*. 1290–1295.

Yang Zhao and Krishnendu Chakrabarty. 2010. Synchronization of washing operations with droplet routing for cross-contamination avoidance in digital microfluidic biochips. In *Proceedings of the IEEE/ACM Design Automation Conference*. 635–640.

Antoine Zoghbi and Ivan Stojmenović. 1998. Fast algorithms for generating integer partitions. *Int. J. Comput. Math.* 70, 2 319–332.

Received July 2014; revised November 2014; accepted January 2015