Dilution and Mixing Algorithms for Flow-Based Microfluidic Biochips

Sukanta Bhattacharjee, Sudip Poddar, Sudip Roy, *Member, IEEE*, Juinn-Dar Huang, *Member, IEEE*, and Bhargab B. Bhattacharya, *Fellow, IEEE*

***Abstract*—Albeit sample preparation is well-studied for digital microfluidic biochips, very few prior work addressed this problem in the context of continuous-flow microfluidics from an algorithmic perspective. In the latter class of chips, microvalves and micropumps are used to manipulate on-chip fluid flow through microchannels in order to execute a biochemical protocol. Dilution of a sample fluid is a special case of sample preparation, where only two input reagents (commonly known as *sample* and *buffer*) are mixed in a desired volumetric ratio. In this paper, we propose a satisfiability-based dilution algorithm assuming the generalized mixing models supported by an N-segment, continuous-flow, rotary mixer. Given a target concentration and an error limit, the proposed algorithm first minimizes the number of mixing operations, and subsequently, reduces reagent-usage. Simulation results demonstrate that the proposed method outperforms existing dilution algorithms in terms of mixing steps (assay time) and waste production, and compares favorably with respect to reagent-usage (cost) when 4- and 8-segment rotary mixers are used. Next, we propose two variants of an algorithm for handling the open problem of k-reagent mixture-preparation ($k \geq 3$) with an N-segment continuous-flow rotary mixer, and report experimental results to evaluate their performance. A software tool called *FloSPA* (Flow-Based Sample Preparation) has also been developed that can be readily used for running the proposed algorithms.**

***Index Terms*—Flow-based microfluidic biochip, sample preparation, satisfiability**

I. Introduction

ADVANCES in microfluidic technologies over the last few decades have boosted the miniaturization of biochemical laboratory for medical diagnostics and drug discovery with the advantages of reduced consumption of sample and reagent fluids. A microfluidic biochip or "lab-on-a-chip" device is a highly integrated system with the capability of performing multiple tasks required in a biochemical laboratory. Miniaturized biochips offer many advantages over large-scale laboratory instrumentation, including improvements in sample processing speed and throughput. Several microfluidic lab-on-a-chip platforms have been developed, e.g., droplet-based [1–8], digital [2, 9], and continuous flow-based microfluidics [10].

Manuscript received February 4, 2016; revised May 21, 2016; accepted June 28, 2016. The work of B. B. Bhattacharya is supported by a special grant to Nanotechnology Research Triangle from Indian Statistical Institute, Kolkata. This paper was recommended by Associate Editor C. H. Chang.

S. Bhattacharjee, S. Poddar, and B. B. Bhattacharya are with the Advanced Computing and Microelectronics Unit, Indian Statistical Institute, Kolkata, India 700108. E-mail: {sukanta_r, sudipp_r, bhargab}@isical.ac.in

S. Roy is with the Department of Computer Science and Engineering, Indian Institute of Technology, Roorkee, India. E-mail: sudiproy.fcs@iitr.ac.in

J.-D. Huang is with Department of Electronics Engineering, National Chiao Tung University, Hsinchu 300, Taiwan. E-mail: jdhuang@mail.nctu.edu.tw

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

Digital Object Identifier xx.xxxx/TCAD.2016.xxxxxxx

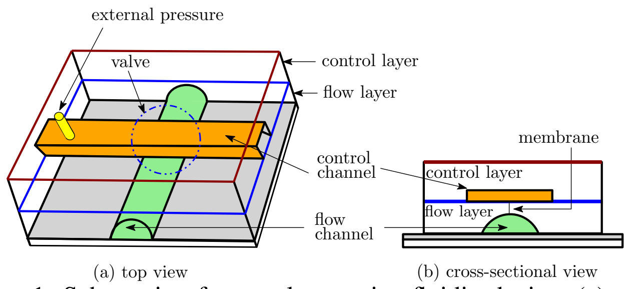


Fig. 1: Schematic of a two-layer microfluidic device: (a) top view, (b) cross-sectional view.

A continuous flow-based microfluidic biochip (*CFMB*) is equipped with several pressure-driven microvalves to manipulate fluid flow in a network of microchannels. A typical microfluidic device consists of two separate layers of elastomer (polydimethylsiloxane) called flow layer and control layer as shown in Fig 1. Soft-lithographic technique is used to fabricate the network of microchannels in both flow and control layers. A flexible membrane/microvalve is formed at the overlapping area between the channels of two layers to control the movement of biochemical fluids in the flow layer. An external pressure source in the control layer is used to deflect membrane/valve deep into the flow channels to block fluidic flow. Channels in flow layers are also connected to an external pressure source. More complex units such as mixers, micropumps, and multiplexers can be built with several hundreds of such units accommodated on a single chip [10].

*CFMB*s are quite popular in the biochemistry community because of their simplicity of fabrication, flexibility in reagent-volume control, and versatility of applications such as automation of assays and point-of-care diagnosis [11]. Many clinical applications, e.g., DNA analysis [12], drug discovery [13], and sample preparation [14] have been successfully implemented on *CFMB*. In most of the biochemical applications, automatic sample preparation plays a significant role, e.g., it accounts for 60% of the work in analytical tests [15]. Microfluidic networks based on *CFMB* can be used to produce logarithmic and linear concentration gradients effectively [16, 17]. Moreover, flow-based microfluidic devices have been developed for implementing serial and parallel mixing, and also for producing dilution gradients, and diffusion-based gradients [17].

In recent years, many sample preparation algorithms [2–

6] have been proposed for preparing a mixture of fluids with a digital microfluidic biochip (*DMFB*), based on the $(1:1)$ mixing model. On a continuous flow-based microfluidic platform, an N-segment rotary mixer that is divided into N equal-length segments (henceforth, denoted as Mixer-N), can be conveniently used to implement more powerful and multiple mixing models [18]. Rotary mixers with micropumps use circular motion to mix two fluids that are pumped into a ring-like or rotary structure [19]. However, all previous work [18, 20] fail to exploit the full functionality of different mixing models that are supported by a rotary mixer as they impose several constraints on allowable mixing steps. It is also observed that flow-based microfluidic mixing operations are quite slow [21], and hence, it is desirable to reduce such operations as far as possible during sample preparation.

In this paper, we first propose a dilution algorithm for *CFMB*s that can be used to prepare a mixture of two reagents (*sample* and *buffer*) using a rotary mixer that minimizes the number of mixing operations, and reduces reagent-usage. Secondly, we generalize our approach to propose a reagent-saving mixture preparation algorithm, where the number of reagents may be greater than two. To the best of our knowledge, this is the first attempt to solve algorithmic mixture preparation with *CFMB*s by using a Mixer-N.

We present a novel formulation of the dilution problem using a set of linear constraints over integers, and then invoke a solver based on *Satisfiability Modulo Theory* (*SMT*) [22] to obtain the optimal solution. The proposed algorithm is guaranteed to produce a sample of desired target concentration factor within a tolerable error limit, while minimizing the number of mixing operations as a *primary objective*, and minimizing reagent-usage as a *secondary objective* criterion. Unlike existing methods [18, 20], we deploy a two-step optimization technique: first, we construct a mixing tree that consists of a minimum number of mixing steps; next, using an *SMT*-based approach, we search among all such trees to find the one that produces the desired target concentration with minimum sample-usage. Simulation results show that for an 8-segment mixer, the proposed dilution algorithm terminates very fast (in around 0.34 second on a 2 GHz, Intel Core i5 processor), and on the average it reduces the number of mixing operations and waste production compared to prior work *TPG* [20] and *VOSPA* [18]. Furthermore, with respect to reagent-usage, it improves upon *TPG* and compares favorably with *VOSPA*. Moreover, the performance of mixture-preparation algorithms are also studied in detail.

The rest of this paper is organized as follows: Basics of sample preparation and mixing models are provided in Section II, and a brief review is presented in Section III. A motivational example is given in Section IV. An overview of the solution procedure is presented in Section V. The proposed method for dilution with a *CFMB* is presented in Section VI. Reagent-saving mixture preparation is described in Section VII. Experimental evaluation is reported in Section VIII. Finally, the paper is concluded in Section IX with pointers to a few future research challenges.

II. Sample preparation and mixing models

Sample preparation is the process of mixing two or more biochemical fluidic reagents in a given volumetric ratio through a sequence of mixing operations. Dilution is a special case of sample preparation, where only two input reagents (commonly known as *sample* and *buffer*) are mixed in a desired volumetric ratio. In a *DMFB*, two equal-volume droplets are mixed using the $(1:1)$ mixing model [5–7]. On the other hand, a *CFMB* supports multiple ratios as mixing models by deploying a Mixer-N [18]. A Mixer-N is divided into N equal-length segments, where each segment can be filled with a fluid.

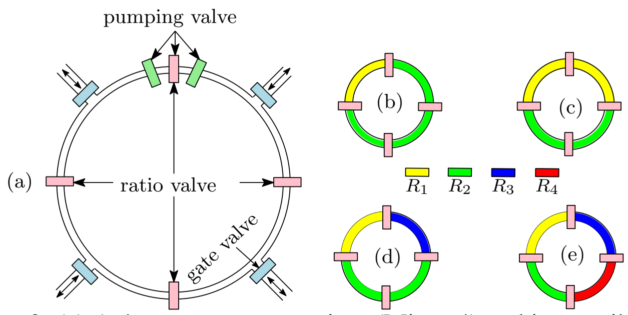


Fig. 2: (a) A 4-segment rotary mixer (Mixer-4) and its possible mixing models (b) $(1:3)$, (c) $(1:1)$, (d) $(1:1:2)$, and (e) $(1:1:1:1)$.

Example 1. Fig. 2 shows a Mixer-4, and various mixing models supported by this device. Different valves are used for loading and unloading a segment (gate valve), for determining the segment boundary (ratio valve), and for mixing of fluids within the segments (pumping valve). A description of such mixers can be found in [23]. Figs. 2(b)-2(e) show each of the four different reagents with a different color. ■

A mixture of k reagents $R_1, R_2, \ldots, R_k$ is denoted as $\mathcal{M} = \{\langle R_1, c_1\rangle, \langle R_2, c_2\rangle, \cdots, \langle R_k, c_k\rangle\}$, where $\sum_{i=1}^{k} c_i = 1$ (validity condition) and $0 \le c_i \le 1$ for $i = 1, 2, \ldots, k$. In other words, $R_1, R_2, \cdots, R_i, \cdots, R_k$ are mixed with a ratio of $\{c_1 : c_2 : \cdots : c_i : \cdots : c_k\}$, where c_i denotes the *concentration factor* (*CF*) of R_i. The condition $\sum_{i=1}^{k} c_i = 1$ [2] ensures the validity of a mixing ratio. Note that the *CF* of a *pure* (100%) reagent R_i is assumed to be 1, and the *CF* of neutral *buffer* (0%) is assumed to be 0. Thus, the value of each c_i is normalized with respect to 1, i.e., the sum of all c_i's. Because of the inherent mixing models used for microfluidic implementation of mixers, each c_i is required to be approximated as a special form $\frac{x_i}{N^d}$, where x_i, N, and d are positive integers, i.e., $x_i, N, d \in \mathbb{N}$. The value of N is determined by the mixing model supported by the microfluidic mixer, and d is determined by the desired *accuracy* of approximation (error-tolerance limit ϵ in *CF*, $0 \le \epsilon < 1$), which is user specified. A lower value of ϵ denotes higher accuracy in *CF*. Assuming the underlying mixing model of Mixer-N and for a given value of error-tolerance limit ϵ, we choose the minimum value of d such that each c_i in mixture $\mathcal{M}$ is approximated as $\frac{x_i}{N^d}$, where $x_i \in \mathbb{N}$ and $\max_i\{|c_i - \frac{x_i}{N^d}|\} < \epsilon$ and $\sum_{i=1}^{k} \frac{x_i}{N^d} = 1$. In other words, each c_i is approximated as a d-digit N-ary fraction. For example, in case of $(1:1)$

mixing model ($N = 2$) and a given ϵ, mixture $\mathcal{M}$ needs to be approximated as $\{\frac{y_1}{2^d} : \frac{y_2}{2^d} : \cdots : \frac{y_k}{2^d}\} \equiv \{y_1 : y_2 : \cdots : y_k\}$ by choosing the smallest d such that $\max_i\{|c_i - \frac{y_i}{2^d}|\} < \epsilon$ and $\sum_{i=1}^{k} y_i = 2^d$. The following example illustrates the ratio approximation procedure and the choice of d.

Example 2. Consider a mixture $\mathcal{M} = \{\langle R_1, c_1 = 0.25\rangle, \langle R_2, c_2 = 0.30\rangle, \langle R_3, c_3 = 0.45\rangle\}$, which is to be generated using $(1 : 1)$ model with an error-tolerance limit $\epsilon = 0.004$. Table I shows the detailed approximation procedure; we increase d iteratively starting from 1 to find a solution. The rounding procedure for $d = 4$ is shown in the footnote of Table I. Note that for $d \leq 5$, the error-tolerance limit is not satisfied, and for $d = 6$, the approximated ratio is $\mathcal{M}' = \{\langle R_1, c'_1 = \frac{16}{2^6}\rangle, \langle R_2, c'_2 = \frac{19}{2^6}\rangle, \langle R_3, c'_3 = \frac{29}{2^6}\rangle\}$ or $\{R_1 : R_2 : R_3 = 16 : 19 : 29\}$ that satisfies the given error-tolerance and the validity condition: $16 + 19 + 29 = 2^6$. ■

TABLE I: Approximating a target ratio based on mixing model and user-defined error-tolerance in *CF*.

d	Approx. ratio	Error in *CF*
4	$4 : 5 : 7$[†]	$\max\{\lvert 0.25 - \frac{4}{2^4}\rvert, \lvert 0.30 - \frac{5}{2^4}\rvert, \lvert 0.45 - \frac{7}{2^4}\rvert\} = 0.01 > \epsilon$
5	$8 : 10 : 14$	$\max\{\lvert 0.25 - \frac{8}{2^5}\rvert, \lvert 0.30 - \frac{10}{2^5}\rvert, \lvert 0.45 - \frac{14}{2^5}\rvert\} = 0.01 > \epsilon$
6	$16 : 19 : 29$	$\max\{\lvert 0.25 - \frac{16}{2^6}\rvert, \lvert 0.30 - \frac{19}{2^6}\rvert, \lvert 0.45 - \frac{29}{2^6}\rvert\} = 0.003 < \epsilon$

[†] $16 \times 0.25 = 4; 16 \times 0.3 = 4.8 \approx 5; 16 \times 0.45 = 7.2 \approx 7$; and $4 + 5 + 7 = 2^4$
For $d = 1, 2, 3$, $\max_{i \in \{1,2,3\}}\{|c_i - \frac{x_i}{2^d}|\} \geq \epsilon$, where c_i is approximated as $\frac{x_i}{2^d}$

A mixing graph [3, 4] is commonly used to represent the dependencies of mixing operations, where a node with zero in-degree (out-degree) represents an input reagent (target ratio). The internal nodes represent intermediate mixing operations. In the domain of *CFMB*, an incoming edge-weight denotes the number of segments that are to be filled with input reagents, or fluids with previously-obtained intermediate *CF*s for the concerned mixing operation.

In the case of dilution problem, the concentration factor (CF) of the mixture denotes the volumetric ratio of *sample* and *buffer* fluids, i.e., the portion of *sample* fluid present in the mixture fluid (hence, $0 \leq CF \leq 1$). Assuming that N segments of a Mixer-N are filled with CFs $c_1, c_2, \cdots, c_N$, of a fluid diluted with the same *buffer*; then the resultant CF of the mixture will be $\frac{\sum_{i=1}^{N} c_i}{N}$ after homogeneous mixing. In the case of dilution, the mixing graph is commonly known as dilution graph. Fig. 3 shows dilution graphs corresponding to the target ratio $\{\langle$*sample*$, \frac{125}{4^4}\rangle, \langle$*buffer*$, \frac{131}{4^4}\rangle\}$ or $\{$*sample* : *buffer* $= 125 : 131\}$ obtained by four different algorithms, when Ring-3 is used for *TPG* [20], and Mixer-4 is used for *NWayMix*, *VOSPA* [18], and *FloSPA-D*.

III. State-of-the-art

The algorithmic aspects of sample preparation with a *CFMB* have recently been studied by many researchers. Liu *et al.* [20] developed the tree pruning and grafting algorithm (*TPG*) by transforming an initial mixing graph that is obtained based on the $(1 : 1)$ mixing model, which is often used in digital microfluidics, e.g., in *twoWayMix* [2] or *REMIA* [4]. The *TPG* algorithm applies tree-pruning and grafting on the initial dilution graph to obtain the dilution graph for a special type of unequally-segmented rotary mixer (Ring-N [20]). In each mixing step, *TPG* considers only a few specific volumetric ratios supported by $(k : \ell)$ mixing model ($k + \ell = 2^m$, $k, \ell, m \in \mathbb{N}$) supported by Ring-$N$. Therefore, *TPG* is unable to exploit all available mixing models for Ring-N. For example, in Ring-3, the possible mixing models are $(1 : 1), (1 : 3)$ and $(1 : 1 : 2)$; however, *TPG* uses only first two of them.

The volume-oriented sample preparation algorithm (*VOSPA* [18]) uses a uniformly-segmented rotary mixer (Mixer-N), and exploits the power of utilizing multiple intermediate *CF*s. *VOSPA* consists of two phases: master process and subsidiary process. The subsidiary process fills a concentration bank that can be used by the master process, which greedily accumulates various *CF*s from the bank for each segment of the mixer. Note that only one segment of Mixer-N is reused between two calls of the master process, and the remaining $(N - 1)$ segments are left unused. Moreover, the subsidiary process can use only two *CF*s in each mixing step for filling the bank that is subsequently used by the master process.

Note that both *TPG* and *VOSPA* produce a mixture of two reagents: *sample* and *buffer*. Algorithmic mixture preparation for more than two reagents using *CFMB* still remains unexplored. Our method also provides a solution to this general problem. Table II presents the scope of the proposed work against the background of prior art.

IV. Motivation and Contribution

One of the major objectives in sample preparation is to minimize the assay time, which is proportional to the number of mixing operations [24]. Thies *et al.* [2] proposed a sample preparation algorithm (*MinMix*) for generating a target ratio based on the $(1 : 1)$ mixing model. A special case of *MinMix* (known as *twoWayMix* [2] algorithm) can be used to perform dilution with at most d mixing steps, where d is the number of bits used in approximating the target concentration with a binary fraction. In the case of Mixer-N, we can generalize *twoWayMix* for *CFMB*s so that it produces a given target CF based on multiple mixing models that are supported by Mixer-N; let us call this procedure as *NWayMix*. The entire range of concentrations generated with Mixer-N can thus be represented as $\{x : y \,|\, x + y = N^d\}$, where d digits in base-N are used to approximate a *CF* as $\frac{x}{N^d}$. **Algorithm 1** describes the outline of *NWayMix*.

Example 3. Consider a mixing-ratio $\{$*sample* : *buffer* $= 125 : 131\}$, or equivalently, *sample-CF* in the target is $\frac{125}{256}$. Assume the availability of Mixer-4 ($N = 4$). Since the sum of ratio-components $125 + 131 = 256 = 4^4$, error-tolerance $\epsilon = 0$ can be achieved when d is chosen as 4. In Fig. 3(c), the bottommost node indicates that one segment of the rotary Mixer-4 is filled with pure *sample*, whereas three remaining segments are filled with neutral *buffer*. Hence, the resultant *CF* of the *sample* in the mixture obtained after mixing is $\frac{1+3\times0}{4} = \frac{1}{4}$. The *CF* of the *buffer* will be $1 - \frac{1}{4} = \frac{3}{4}$. Hence, for this intermediate fluid, $\{$*sample* : *buffer* $= 1 : 3\}$.

At the second mixing step (second node from bottom), three segments are filled with pure sample, and the remaining

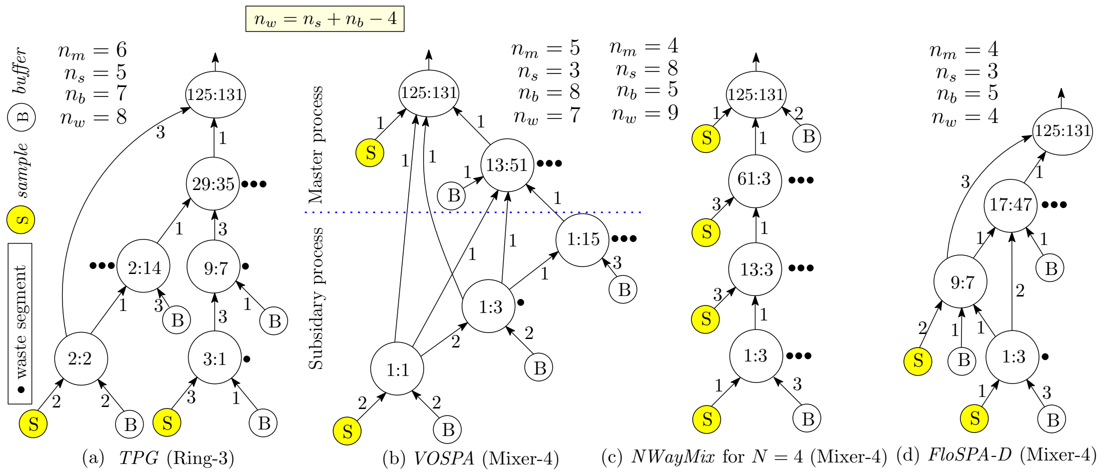

$n_w = n_s + n_b - 4$
● waste segment
S sample
B buffer
$n_m = 6$ $n_s = 5$ $n_b = 7$ $n_w = 8$
$n_m = 5$ $n_s = 3$ $n_b = 8$ $n_w = 7$
$n_m = 4$ $n_s = 8$ $n_b = 5$ $n_w = 9$
$n_m = 4$ $n_s = 3$ $n_b = 5$ $n_w = 4$
Master process
Subsidary process
125:131
29:35
2:14
9:7
2:2
3:1
13:51
1:15
1:3
1:1
61:3
13:3
17:47
(a) TPG (Ring-3)
(b) VOSPA (Mixer-4)
(c) NWayMix for $N = 4$ (Mixer-4)
(d) FloSPA-D (Mixer-4)
n_m : number of mixing operations; n_w : number of waste segments; $n_s(n_b)$: number of segments filled up with sample (buffer).

Fig. 3: Generation of four units of target ratio $(125 : 131)$ using sample preparation algorithms (a) *TPG*, (b) *VOSPA*, (c) *NWayMix*, and (d) the proposed algorithm *FloSPA-D*.

TABLE II: Scope of sample preparation algorithms on *CFMB*.

Attribute	*TPG* [20]	*VOSPA* [18]	Proposed
1. Purpose	Reagent minimization	Reagent minimization	Reagent minimization using least-depth mixing graph
2. Solution strategy	Iteratively modifies the dilution graph created by existing *DMFB* dilution algorithms[1]	Greedy-heuristic based	*SMT*-based optimal modeling
3. Mixing models	Unable to exploit all mixing-ratios of Ring-*N*	Only master process may use all mixing-ratios of Mixer-*N*	Can utilize any mixing-ratio produced by Mixer-*N*, as needed
4. # Mixing steps in dilution	Minimality not ensured	Minimality not ensured	Minimality guaranteed
5. Mixture preparation	Not addressed	Not addressed	Reagent-saving mixture preparation

[1] based on the $(1 : 1)$ mixing model only

Algorithm 1: *NWayMix*$(\{x : y\}, N, d)$

```
Input: {sample : buffer = x : y}: Target ratio, N: Mixer-N, d: accuracy
Output: Mixing steps
/* Represent $x$ as $d$ digits in base-$N$ and $1 \le x \le N^d - 1$ */
1  $x = x_{d-1}x_{d-2} \dots x_1x_0$, where $x_i \in \{0, 1, \dots, N-1\}, 0 \le i \le d-1$;
   /* Discard 0's, if any, from right to retain $d'$ digits, $d' \le d-1$ */
2  $j = 0$;
3  while $x_j = 0$ do
4      $j = j + 1$;
5  Fill $x_j$ segments with sample and the remaining $(N - x_j)$ segments with buffer;
   /* $Mix()$ denotes the mixing operation after filling all segments of Mixer-$N$ */
6  $newFluid = Mix()$;
7  for $(j = j+1; j \le d-1; j = j+1)$ do
8      Fill $x_j$ segments with sample and $(N - x_j - 1)$ segments with buffer;
9      Fill the last segment with $newFluid$;
10     $newFluid = Mix()$;
```

one segment is filled with the intermediate fluid obtained at the previous step. Hence, *sample-CF* will be $\frac{3+\frac{1}{4}}{4} = \frac{4\times3+1}{4^2}$. Similarly, the *sample-CF* produced at the third mixing node is $\frac{4^2\times3+4\times3+1}{4^3}$. Finally, at the target node, it will become $\frac{4^3\times1+4^2\times3+4^1\times3+4^0\times1}{4^4} = \frac{125}{4^4}$. Henceforth, for the sake of convenience, we will call *sample-CF* as target *CF*.

Note that, $125 = (1331)_4$. Algorithm 1 essentially expands 125 in base-4, i.e., $125 = (1331)_4 = 4^3 \times 1 + 4^2 \times 3 + 4^1 \times 3 + 4^0 \times 1$. Hence, conversely, the target *CF* can be achieved by filling the segments with pure *sample* in the same sequence as depicted in the Fig. 3(c). ■

The above argument can be generalized to justify the correctness of Algorithm 1. Given the target *CF*, N, and ϵ, the value of d is first derived. Next, we express the target *CF* as a fraction with at most d-digits in base-N. Since in each mixing step, we retain only one segment of previously produced fluid, the contribution of the pure *sample* added to a particular mixing step is reduced by a factor of N with each mixing step it undergoes while reaching the target. The process terminates after at most d mixing steps producing a *CF* value that exactly corresponds to the desired target ratio.

Note that $\frac{x}{N^d}$ $(x, N, d \in \mathbb{N})$ may be reducible, i.e., there may be a common factor of N between the numerator and denominator. To express the ratio in irreducible form, we simplify it as follows. Let $\frac{x}{N^d} = \frac{x'}{N^{d'}}$ $(x', d' \in \mathbb{N}, d' \le d)$, where x' is not divisible by N $(x'\%N \ne 0)$, i.e., we reduce the numerator and denominator until there does not exist any common factor of N between them. Note that, after reduction d'-bit representation of x' will have no trailing zeros in its base-N representation. Theorem 1, stated below, guarantees the minimality of the number of mixing steps required by *NWayMix*.

Theorem 1. Algorithm *NWayMix* generates a target ratio $\{sample : buffer = x : y\}$, where $x + y = N^d$, with minimum number of mixing operations d' $(d' \le d)$, when an N-segment rotary mixer (Mixer-N) is used.

Proof: By construction, it follows that the mixing graph is a tree, where the total number of internal nodes (representing mixing operations) is equal to the depth of the tree. Thus, the mixing tree resembles a chain (i.e., a skewed tree). Note that the target $CF = \frac{x}{N^d} = \frac{x'}{N^{d'}}$, where x' and $N^{d'}$ do not have any common factor of N. Hence, the depth of the tree is exactly d'. ■

Although *NWayMix* generates a target CF using the minimum number of mixing operations, fluids residing in $(N-1)$ segments are wasted in each mixing step when Mixer-N is used, as in the master process of *VOSPA* [18]. Fig. 3 shows the performance parameters of different sample preparation techniques for the target ratio $\{125 : 131\}$. It is observed that *NWayMix* produces the target CF using the fewest mixing operations but it consumes more units of reagents. On the other hand, *TPG* and *VOSPA* attempt to minimize reagent-usage but may need more mixing steps, i.e., they may increase sample preparation time. A natural question in the context is: Can we reach a target CF with minimum mixing operations, and then minimize reagent-usage as a secondary objective? Fig. 3(d) shows the dilution graph generated by the proposed algorithm *FloSPA-D*, which not only minimizes the number of mixing operations but also reduces reagent-usage and waste-production by cleverly sharing intermediate *CF*s. The following section briefly describes the overview of the solution procedure.

V. Overview of the Proposed Method

We use a modeling that solves the dilution problem based on decision procedures over linear arithmetic, i.e., by utilizing the deductive power of solvers for *SMT* [25]. The *SMT* over linear arithmetic (*SMT($\mathcal{LA}$)*) problem is defined as follows.

Definition 1. Let Ψ be a first-order propositional formula, where the atoms are linear equations. The *SMT($\mathcal{LA}$)* problem is to determine an assignment to the variables of Ψ, if there exists a satisfiable assignment, otherwise to prove that no such assignment exists [26]. If a satisfiable assignment exists, we call the formula Ψ *satisfiable*, otherwise Ψ is *unsatisfiable*. ■

Example 4. Let $\Psi = (2x + 3y \leq 5) \wedge (3x + 5y \geq 6) \wedge (x \geq 0) \vee (y \leq 0)$. Here, Ψ is *satisfiable*, where $\{x = 2, y = 0\}$ is a satisfiable assignment. ■

Solving engines based on *SMT* algorithms have now reached enough sophistication, and hence they are being used for handling many complex optimization problems efficiently [27–29]. Although *SMT*-solvers are used to solve a decision problem, an optimization problem can be formulated as a sequence of decision problems. In our context, we first model the problem of diluting a *sample* using Mixer-N as a set of linear constraints over integers. Next, we invoke the *SMT($\mathcal{LA}$)* solver [22] to check whether there exists a satisfiable assignment of underlying variables that can generate the target CF using only one unit of *sample*. If the answer is *yes*, we are done. Otherwise, we keep on increasing the number of *sample*-units and re-check for satisfiability until we are successful. Fig. 4 shows the overview of the proposed dilution process. In the following sections, we now formulate the optimization problem in detail.

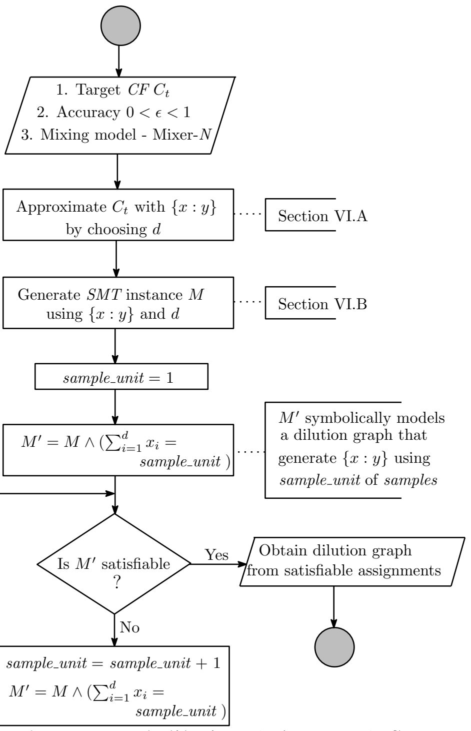

1. Target CF C_t
2. Accuracy $0 < \epsilon < 1$
3. Mixing model - Mixer-N
Approximate C_t with $\{x : y\}$ by choosing d
Section VI.A
Generate SMT instance M using $\{x : y\}$ and d
Section VI.B
$sample\_unit = 1$
$M' = M \wedge (\sum_{i=1}^{d} x_i = sample\_unit)$
M' symbolically models a dilution graph that generate $\{x : y\}$ using $sample\_unit$ of $samples$
Is M' satisfiable ?
Yes
Obtain dilution graph from satisfiable assignments
No
$sample\_unit = sample\_unit + 1$
$M' = M \wedge (\sum_{i=1}^{d} x_i = sample\_unit)$

Fig. 4: Proposed dilution (*FloSPA-D*) flow.

VI. Dilution: Proposed Method

We present a methodology that exploits the fact that *NWayMix* generates a target concentration with the minimum number of mixing steps when Mixer-*N* is used. Initially, we start with the mixing tree generated by *NWayMix*. Next we use a SAT-based technique to transform the tree into a mixing graph that corresponds to least reagent-usage. Henceforth, we will denote the proposed method of Flow-based Sample Preparation Algorithm for Dilution as *FloSPA-D*. The dilution problem on a *CFMB* is now formally stated as follows:

- **Inputs:**
- A supply of *sample* ($CF = 1$) and *buffer* ($CF = 0$);
- A target concentration factor $C_t \in (0, 1)$;
- An N-segment rotary mixer (Mixer-N);
- A user-defined error-tolerance limit $\epsilon, 0 \leq \epsilon < 1$;
- **Output:**
- Mixing graph for generating C_t within error limit ϵ starting with *sample* and *buffer* as input nodes;
- **Objective:**
- Minimize the number of mixing operations as primary objective, and then minimize reagent-usage as secondary objective.

A. Approximating the given target concentration factor

Given a target CF $C_t \in (0, 1)$, error-tolerance $\epsilon, 0 \leq \epsilon < 1$ and the underlying mixing model supported by Mixer-N, we approximate the target CF as $C'_t = \frac{x}{N^d}$ as stated earlier in Section II. Note that by Theorem 1, d also determines the depth of the skewed dilution tree, where the target node lies at depth 1.

Example 5. Let the desired *CF* $C_t = 0.39$, and a Mixer-4 is given, i.e., $N = 4$. Assume $\epsilon = 0.007$. Consider an

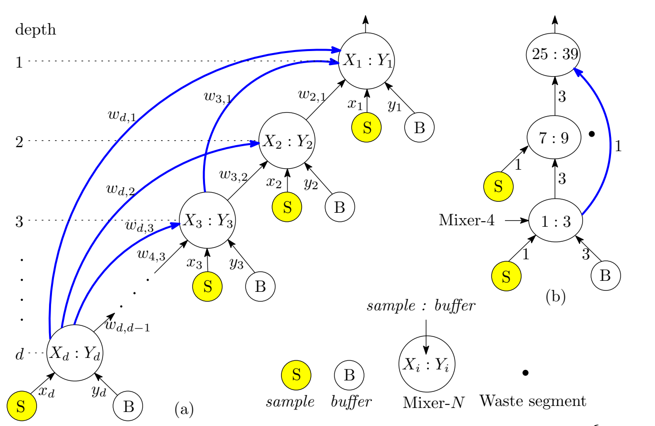


Fig. 5: (a) Skeleton of a dilution graph for a target ratio $\{X_1 : Y_1\}$, where $X_1+Y_1 = N^d$, using mixing models supported by Mixer-N, and (b) the optimal solution for target ratio $\{25 : 39\}$ using Mixer-4.

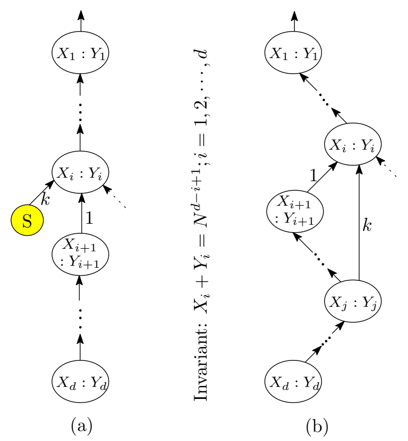


Fig. 6: Skeleton of a dilution graph for proving the statements in (a) Lemma 1, and (b) Lemma 2

approximation of C_t as $\{sample : buffer = 25 : 39\}$. Note that $|\frac{25}{4^3} - 0.39| = 0.0006 < \epsilon(= 0.007)$, for $d = 3$, which is the smallest value that satisfies the given error-tolerance. Hence, the depth of the skewed dilution tree is 3. ■

B. Modeling of dilution

The procedure *NWayMix* gives the minimum number of mixing steps for a given target ratio when Mixer-N is used. The resultant mixing tree is a chain (skewed) and each intermediate mixing step produces $(N-1)$ waste segments. In our modeling, we will try to maximize the reuse of intermediate fluids. The underlying search problem is modeled as a set of linear equations with logical connectives. For the sake of simplicity, we have initially modeled it as non-linear constraints over integers, and then we systematically transform them into a set of linear constraints.

Modeling formalism

Fig. 5(a) shows the structure of the dilution graph for target ratio $\{X_1 : Y_1\}$, where $X_1 + Y_1 = N^d$. The dilution graph generated by *NWayMix* takes d mixing steps for generating $\{X_1 : Y_1\}$ using Mixer-N, i.e., the depth of the dilution graph is d. In order to identify how intermediate-fluids can be reused, we need to transform the mixing tree generated by *NWayMix* by adding additional edges that represent such reuse of *CF*s. These edges represent possible sharing of intermediate CFs between non-adjacent mixing operations. In Fig. 5(a), highlighted edges show the possibility of using intermediate CFs between non-consecutive mixing operations. Note that the knowledge of d is sufficient to generate the mixing graph corresponding to *FloSPA-D* while adding new edges on the skewed mixing tree. The value of d can be determined based on the error-tolerance (ϵ) and N. Hence, *NWayMix* need not be run explicitly in order to create a *FloSPA-D* instance. Before going into the details, we present the following properties of the dilution graph with reference to Fig. 6.

Lemma 1. If k segments of a Mixer-*N* are filled with *sample* and are used in the mixing operation at depth i, where $1 \leq i \leq d$, they contribute the amount kN^{d-i} to the *sample* content X_i at this depth. The same is true for *buffer* also.

Proof: By Theorem 1, the dilution graph is a tree of depth d. Note that, $X_i+Y_i = N^{d-i+1}$ for $1 \leq i \leq d$. Let k segments of the mixer at depth i be filled with the *sample*. Without loss of generality, we assume that one segment of the Mixer-N at depth i is filled with the intermediate-fluid obtained at preceding mixing operation at depth $i+1$, and the remaining $(N-k-1)$ segments are filled with *buffer*, or with intermediate *CF*s from previous operations at depth $i+2, i+3, \ldots, d$. Fig. 6(a) shows the skeleton of the dilution graph in this setting. Hence, the portion (X_i) of the *sample* in the resultant mixing operation executed at depth i, can be calculated as shown below, where $(\cdots)$ denotes the contributions of *sample* to X_i received form other incoming edges (dotted lines in Fig. 6(a)):

$$X_i = \frac{\frac{X_{i+1}}{N^{d-i}} + k + (\cdots)}{N} = \frac{(\cdots) + kN^{d-i} + (\cdots)}{N^{d-i+1}}$$

Hence, an amount of kN^{d-i} is contributed to X_i. ■

Example 6. Consider a mixing node at depth 2 inFig. 3(c) with concentration-ratio $\{sample : buffer = 61 : 3\}$. Note that target ratio is at depth 1. Three segments are filled with *sample* i.e., $k = 3$, and the remaining segment is filled with fluids from mixing node at depth 3. Hence, the portion of *sample* in the mixing node at depth 2 is $\frac{3+\frac{13}{4^2}}{4} = \frac{3\times4^2+13}{4^2\times4} = \frac{61}{4^3}$. Note that three units of *sample* and one unit of intermediate fluid reused from the mixing node at depth 3, contribute $3 \times 16 = 48$ and 13, respectively, to produce 61 at depth 2. ■

Lemma 2. If the fluid in k segments of Mixer-*N* at depth j is shared at depth i, where $1 \leq i < j \leq d$, then it contributes an amount $N^{j-i-1}X_j$ and $N^{j-i-1}Y_j$ to X_i and Y_i, respectively.

Proof: By Theorem 1, the dilution graph is a tree of depth d. Note that, $X_i+Y_i = N^{d-i+1}$ for $1 \leq i \leq d$. Let k segments of the rotary mixer at depth i be filled with the intermediate fluid produced after the mixing operation at depth j, where $j > i$. Without loss of generality, we assume that a segment of the rotary mixer at depth i is filled with the intermediate fluid at depth $i+1$, and the remaining $(N-k-1)$ segments are filled with *sample*, *buffer*, or with intermediate *CF*s produced at depth $i+2, i+3, \cdots, j-1, j+1, \cdots, d$. Fig. 6(b) shows the skeleton of the dilution graph in this setting. The portion of the sample in the resultant mixing operation at depth i, i.e., X_i can be calculated as follows:

$$\begin{aligned} X_i &= \frac{\frac{X_{i+1}}{N^{d-i}} + k\frac{X_j}{N^{d-j+1}} + (\cdots)}{N} \\ &= \frac{X_{i+1} + kN^{(d-i)-(d-j-1)}X_j + (\cdots)}{N^{d-i+1}} \\ &= \frac{(\cdots) + kN^{j-i-1}X_j + (\cdots)}{N^{d-i+1}} \end{aligned}$$

Thus, the mixing operation at depth j contributes the amount $kN^{j-i-1}X_j$ towards X_i at depth i when k segments are filled with intermediate fluids produced at depth j. ■

Example 7. Let us consider a mixing node at depth 2 in Fig. 3(d) where $\{$*sample* : *buffer* = 17 : 47$\}$. Each of the two segments is filled with *buffer* and intermediate fluids obtained from the mixing operation at depth 3. The remaining two segments are filled with fluids from the mixing node at depth 4. Hence, the portion of *sample* in the mixing node at depth 2 is $\frac{\frac{9}{4^2}+2\times\frac{1}{4}}{4} = \frac{\frac{9+2\times 4}{4^2}}{4} = \frac{17}{4^3}$, i.e., when each unit of fluid from the mixing node at depth 4 is used in the mixing node at depth 2, it contributes the amount $4^{(4-2-1)} = 4$ to the *sample* content in the resultant concentration. ■

SMT-based modeling

In the proposed modeling, we define several variables. Fig. 5(a) shows different types of variables as edge labels. The non-negative integer variables are defined as follows.

Node variables: For each mixing node at depth i in the dilution graph, we define two variables X_i and Y_i $(1 \le i \le d)$ that denote the ratio between *sample* and *buffer* in the intermediate fluid produced by the mixing operation at depth i.

Reagent variables: The input reagents (*sample* and *buffer*) can be used in any mixing node at depth i, where $1 \le i \le d$. For denoting the number of segments filled with *sample* and *buffer* in a Mixer-N at depth i, we associate two variables x_i and y_i respectively.

Segment-sharing variables: The integer variable $w_{i,j}$ represents the number of segments that are used to fill Mixer-N at depth j from the intermediate fluid produced by Mixer-N at depth i, where $1 \le j < i \le d$.

Example 8. For the target ratio $\{25 : 39\}$, the depth (d) of the dilution graph is three. Therefore, the exploration of first three depths (1-3) of the dilution graph (shown in Fig. 5(a)) suffices for our search. For $1 \le i \le 3$, the *node variables* are X_i, Y_i, and the *reagent variables* are x_i, y_i. Additionally, $w_{3,2}, w_{3,1}$ and $w_{2,1}$ denote *segment-sharing variables*. ■

The correctness of mixing ratios generated at each node is guaranteed by the following equations that model the consistency of a mixing node at depth j for $\forall j, 1 \le j \le d$.

Note that the ratio of *sample* and *buffer* obtained at the mixing node at depth j is $\{X_j : Y_j\}$. From Fig. 5(a) it can be seen that the mixing node at depth j can fill the segments of Mixer-N with *sample*, *buffer* or with any unused fluid-segments produced at depth $i > j$. From Lemma 1, it follows that *sample* or *buffer* contributes N^{d-j} to Mixer-N at depth j. Moreover, Lemma 2 implies that if $1 \le j < i \le d$, then fluids produced at depth i will contribute $N^{i-j-1}X_i$ and $N^{i-j-1}Y_i$ to X_j and Y_j, respectively, at node j. Therefore, the followings equations should be satisfied for the desired ratio $X_j : Y_j$ at depth j.

$$N^{d-j}x_j + \sum_{i=j+1}^{d} N^{i-j-1}X_i w_{i,j} = X_j \tag{1}$$

$$N^{d-j}y_j + \sum_{i=j+1}^{d} N^{i-j-1}Y_i w_{i,j} = Y_j \tag{2}$$

Note that $X_j + Y_j = N^{d-j+1}$ for $j = 1, 2, \ldots, d$, where X_1 (Y_1) denotes the portion of *sample* (*buffer*) in the target.

Example 9. Consider the target ratio $\{25 : 39\}$. In the corresponding mixing graph, at depth 1, the required constraints are $16x_1 + 4X_3w_{3,1} + X_2w_{2,1} = 25, 16y_1 + 4Y_3w_{3,1} + Y_2w_{2,1} = 39$. Similarly at depth 2, we have $4x_2 + X_2w_{3,2} = X_2, 4y_2 + Y_2w_{3,2} = Y_2$, and at depth 3, $X_3 = x_3, Y_3 = y_3$. ■

Moreover, we need to ensure that all N segments for Mixer-N be filled with intermediate *CF*s or input-reagents; also Mixer-N can feed at most N segments of other mixers. Therefore, the consistency conditions at depth j are given by:

$$x_j + y_j + \sum_{i=j+1}^{d} w_{i,j} = N, \quad \sum_{i=1}^{j-1} w_{j,i} \le N \tag{3}$$

Finally, all weights must satisfy $0 \le w_{i,j} \le N-1$, for $1 \le j < i \le d$. Similarly, $0 \le x_i, y_i \le N-1$ for $1 \le i \le d$.

Example 10. The consistency conditions for depth 1 are $w_{3,1} + w_{2,1} + x_1 + y_1 = 4$; edge-weights must satisfy $0 \le w_{3,1}, w_{2,1}, x_1, y_1 \le 3$. For depth 2, $w_{3,2} + x_2 + y_2 = 4, w_{2,1} \le 4$ and $0 \le w_{3,2}, x_2, y_2 \le 3$. Finally, for depth 3, the desired constraints are $x_3 + y_3 = 4, w_{3,1} + w_{3,2} \le 4$, and $0 \le x_3, y_3 \le 3$. ■

Elimination of non-linearity

Note that the constraints in Eqn. (1) and Eqn. (2) are nonlinear in nature. Unfortunately, the polynomial constraint solving problem over the integers is undecidable [25]. For the purpose of transforming nonlinear equations into *SMT($\mathcal{LA}$)*, we exploit the inherent properties of the underlying mixing graph to construct a modeling proportional to its size. Note that $w_{i,j}$ is non-negative and bounded by the segment size of Mixer-N (in practice $N \le 8$), i.e., $0 \le w_{i,j} \le N-1$. Hence, the non-linearity in the Eqn. (1) and Eqn. (2) can be dealt as follows. For each $X_iw_{i,j}$ in Eqn. (1), we replace it with non-negative integral variable $X'_{i,j}$ and add the following constraints (Eqn. (2) can also be handled similarly):

$$(w_{i,j} = k) \implies (X'_{i,j} = kX_i), \text{ for } k = 0, 1, \cdots, N-1 \tag{4}$$

Example 11. Consider the non-linear equation $16x_1 + 4X_3w_{3,1} + X_2w_{2,1} = 25$ in Example 9. The given equation is transformed into equivalent linear equations as follows: $16x_1 + 4X'_{3,1} + X'_{2,1} = 25$ and $((w_{3,1} = k) \implies (X'_{3,1} = kX_3)) \wedge ((w_{2,1} = k) \implies (X'_{2,1} = kX_2)),\ k = 0, 1, 2, 3.$ ■

C. Dilution algorithm

For a given target CF C_t and an error-tolerance ϵ, we get an approximated target ratio $\{x : y\}$ along with the depth of dilution graph d (line 1 of Algorithm 2), which is same as the number of mixing operations in *NWayMix*. Hence, the optimality in the number of mixing steps is guaranteed (by Theorem 1). However, to minimize reagent consumption, we have adopted an iterative solution approach as outlined in Algorithm 2 (lines 2-9). Line 2 generates the *SMT* instance (as discussed in Section VI-B) for the desired dilution graph of depth d and let it be M. Next, we check the existence of a dilution graph that takes one unit of *sample*. Thus, we add $\sum_{i=1}^{d} x_i = 1$ to M, and obtain M' (line 3-4), and check the satisfiability of M' (line 5). If M' is *satisfiable*, we obtain a solution. Otherwise, we need to check for the existence of a dilution graph that needs one additional unit of *sample*, and re-check for satisfiability (line 6-9). This process of adding more *sample* units is continued until M' is observed to be *satisfiable*. Hence, it guarantees "optimal *sample* consumption" under the constraint of having minimum number of mixing steps. Fig. 5(b) shows the final solution generated by *FloSPA-D* for the target ratio $\{25 : 39\}$. The following theorem formalizes the optimality of the flow-based dilution algorithm *FloSPA-D*.

Algorithm 2: *FloSPA-D*(C_t, ϵ, N)

```
   Input: $C_t$: target $CF$, $\epsilon$: accuracy, $N$: Mixer-$N$
   Output: Dilution Graph
1  Approximate $C_t$ by choosing the smallest $d \in \mathbb{N}$ such that $|\frac{x}{x+y} - C_t| < \epsilon$;
   Target ratio is $\{x : y\}$, where $x + y = N^d$ and $1 \le x \le N^d - 1$;
   /* Detailed modeling is discussed in Section VI-B */
2  $M$ = SMT instance generated from a dilution graph of depth $d$ and $\{x : y\}$;
   /* one sample_unit is equal to the volume of 'reagent'
      in one segment of rotary mixer */
3  sample_unit = 1;
4  $M' = M \wedge (\sum_{i=1}^{d} x_i = sample\_unit)$;
5  $checkSAT(M')$;
6  while $M'$ is unsatisfiable do
7      $sample\_unit = sample\_unit + 1$;
8      $M' = M \wedge (\sum_{i=1}^{d} x_i = sample\_unit)$;
9      $checkSAT(M')$;
10 Obtain dilution graph from satisfiable assignments of $M'$;
11 return dilution graph;
```

Theorem 2. *FloSPA-D generates a target CF C_t, $0 < C_t < 1$ within the given error limit $\epsilon, 0 \le \epsilon < 1$ that minimizes reagent-usage under the constraint of using minimum number of mixing steps deployed by Mixer-N.*

Proof: Theorem 1 guarantees the optimality of the number of mixing operations needed in the dilution tree returned by *NWayMix*. For maximal reuse of intermediate *CF*s produced by *NWayMix*, *FloSPA-D* encodes the dilution tree with a set of linear equations. The correctness of the encoding scheme is guaranteed by Lemma 1 and Lemma 2 Note that decision procedures of the *SMT*-over-linear-arithmetic is sound and complete [26]. We start with one unit of *sample* and invoke the *SMT*-solver. If the answer is *yes*, we obtain a feasible solution. Otherwise, the number of sample-units is increased in steps, and checking for satisfiability is continued until we are successful. Hence, the consumption of minimum amount of reagent is ensured in the final solution. ■

VII. Mixture Preparation: Proposed Method

In this section, we discuss how an *SMT*-based approach can be extended to implement the more general problem of mixture-preparation. We first modify the *MinMix* algorithm (valid only for $(1 : 1)$ mixing model [2]) so that it can generate a mixing tree under the generalized mixing models supported by Mixer-N; let us denote it as *genMixing*. Our objective is to produce a target ratio $\mathcal{M} = \{\langle R_1, c_1\rangle, \langle R_2, c_2\rangle, \ldots, \langle R_k, c_k\rangle\}$ following a sequence of mixing operations supported by Mixer-N.

A. Approximating the given target mixture ratio

Given a target mixture $\mathcal{M}$ of k reagents with an error-tolerance $\epsilon, 0 \le \epsilon < 1$, and a Mixer-$N$, we approximate the target mixture $\mathcal{M}$ with $\mathcal{R} = \{x_1 : x_2 : \ldots : x_k\}$ by choosing the smallest $d \in \mathbb{N}$ such that $\max_i\{|c_i - \frac{x_i}{N^d}|\} < \epsilon$ and $\sum_{i=1}^{k} x_i = N^d$, where d is the depth of the mixing tree representing the sequence of mixing operations that lead to $\mathcal{M}$.

Example 12. Consider a mixture $\mathcal{M} = \{\langle R_1, 0.30\rangle, \langle R_2, 0.23\rangle, \langle R_3, 0.24\rangle, \langle R_4, 0.23\rangle\}$, and assume that Mixer-4 ($N = 4$) is used. Assume that the error-limit is $\epsilon = 0.007$. $\mathcal{M}$ can be approximated as $\{R_1 : R_2 : R_3 : R_4 = 19 : 15 : 15 : 15\}$, since $\max\{|0.30 - \frac{19}{4^3}|, |0.23 - \frac{15}{4^3}|, |0.24 - \frac{15}{4^3}|\} = 0.005 < \epsilon$, and $19 + 15 + 15 + 15 = 4^3$, i.e., a choice of $d = 3$ suffices here. Hence, the depth (d) of the mixing tree is 3. ■

B. Generalized mixing algorithm

In this subsection, we present the algorithm (*genMixing*) that will produce a target ratio using Mixer-N.

Algorithm 3: *genMixing*$(\mathcal{M}, \epsilon, N)$

```
   Input: $\mathcal{M}$: target mixture with $k$ input reagents, $\epsilon$: accuracy, $N$: Mixer-$N$
   Output: Mixing tree
1  Approximate $\mathcal{M}$ with $\mathcal{R} = x_1 : x_2 : \ldots : x_k$ by choosing the smallest $d$ so
   that $\max_{i\in\{1,2,\ldots,k\}}\{|c_i - \frac{x_i}{N^d}|\} < \epsilon$ and $\sum_{i=1}^{k} x_i = N^d$;
2  Create an array of $d + 1$ empty bins $\mathcal{B} = [Bin_0, Bin_1, \ldots, Bin_d]$;
3  for $(i = 1; i \le k; i = i + 1)$ do
       /* Represent $R_i$ as $d$-digit base-$N$ number */
4      Let $R_i = (a^i_{d-1} a^i_{d-2} \ldots a^i_1 a^i_0)_N$;
5      for $(j = 0; j < d; j = j + 1)$ do
           /* push $a^i_j$ copies of $R_i$ at $Bin_j$ */
6          for $(l = 1; l \le a^i_j; l = l + 1)$ do
7              $\mathcal{B}[j].push(R_i)$;
8  return genMixingHelper($\mathcal{B}$, $d$);
```

Algorithm 4: *genMixingHelper*$(\mathcal{B}, d)$

```
   Input: $\mathcal{B}$: An array of $d + 1$ bins, $d$: depth of mixing tree
   Output: Mixing tree
1  if $\mathcal{B}[d] = \phi$ then
2      for $(i = 1; i \le N; i = i + 1)$ do
3          $T_i$ = genMixingHelper($\mathcal{B}$, $d - 1$);
4      Create an internal node $T$ with $\langle T_1, T_2, \ldots, T_N\rangle$ as its children;
5      return $T$;
6  else
7      $R = \mathcal{B}[d].pop()$;
8      Create a leaf node $\mathcal{L}$ with a value of $R$;
9      return $\mathcal{L}$;
```

Example 13. Fig. 7 shows the resultant mixing tree for the target ratio considered in Example 12. Initially, *genMixing*

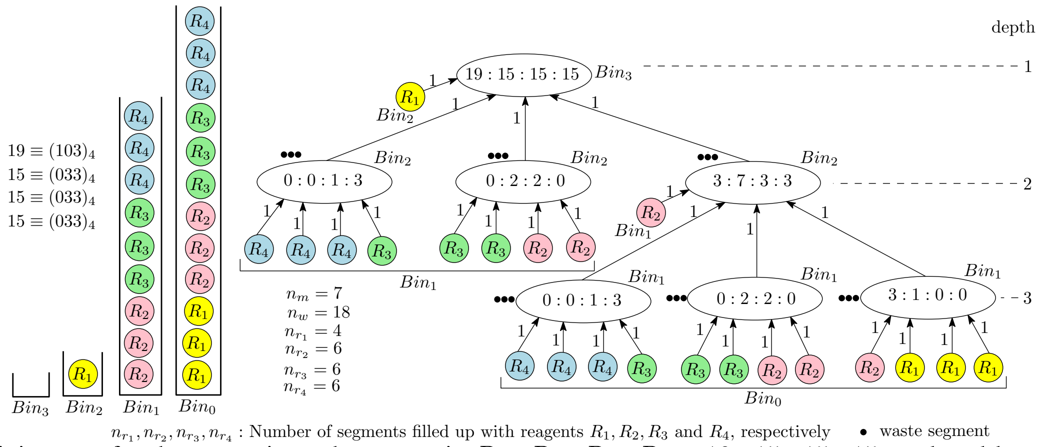


Fig. 7: Mixing tree for the approximated target ratio $R_1 : R_2 : R_3 : R_4 = 19 : 15 : 15 : 15$ produced by *genMixing*.

creates $d+1$ empty bins/stacks $Bin_0, Bin_1, \dots, Bin_d$. Next, it represents each input reagent R_i with d digits using the base-N number system; let $R_i = (a^i_{d-1}a^i_{d-2}\dots a^i_1 a^i_0)_N$. In the next step, *genMixing* scans each digit from right-to-left and populate Bin_j with a^i_j units of R_i for $0 \le j \le d-1$. Note that $R_1 = 19$ is represented as 103 in base-4. Hence, Bin_2 and Bin_0 are populated with 1 and 3 units (1 unit = 1 segment of Mixer-*4*) of reagent R_1, respectively. Finally, *genMixing* invokes another function *genMixingHelper* (Algorithm 4) recursively, to generate the desired mixing tree.

The function *genMixingHelper* (Algorithm 4) invokes the use of bins starting from Bin_d recursively. When a bin is empty, an internal node is created into the mixing tree. Also, N recursive procedures are invoked through Bin_{d-1} for filling N segments of the mixer corresponding to the internal node. Otherwise, it pops one reagent-unit from the non-empty bin and returns it as a leaf/reagent node in the mixing tree. Here, *genMixing* calls *genMixingHelper* with $d = 3$. As Bin_3 is empty, four calls are generated recursively for Bin_2. The first call of *genMixingHelper* for Bin_2 finds it non-empty. Hence, a leaf node with a reagent R_1 is created and attached to the internal node created for Bin_3 (root node in Fig. 7). For all three remaining calls, Bin_2 is empty. The procedure *genMixingHelper* returns the desired mixing tree as shown in Fig. 7, where each node is labeled with Bin_i. ■

Note that *genMixing* produces a mixing tree of depth d, given $\mathcal{M}, \epsilon$, and N. The *CF* of each reagent R_i is expressed as $\frac{x_i}{N^d}$ where $x_i \in \mathbb{N}$. Consider the bottom-up realization of the mixing tree (Fig. 7). In each level, the *CF* of the input reagents is reduced by a factor of N. Thus, in order to achieve N^d in the denominator of *CF*, at least d sequential mixing steps are needed. Algorithm 3 exactly produces a mixing tree of depth d, and hence, it is of minimum depth (height).

C. SMT-based modeling of reagent-saving mixing

In *genMixing*, fluids residing in each of $(N-1)$ segments of the internal mixing nodes are wasted, and hence, the overall reagent consumption may become high. Hence, in contrast to *FloSPA-D*, maximizing the sharing of waste-segments may not be sufficient for reducing reagent-usage in this case. We need to reduce the number of mixing operations by sharing intermediate nodes as far as possible. The proposed modeling that incorporates these issues is described as follows.

Detailed modeling

We create a skeleton tree by adding all missing reagent leaf nodes to all internal/mixing nodes of the mixing tree returned by *genMixing*. The skeleton tree for Fig. 7 is shown in Fig. 8(a). Next, we encode it symbolically using a set of non-negative integer variables and linear equations. The encoding is somewhat similar to the modeling of dilution; however, some additional variables are required for handing multiple reagents. For the ease of illustration, we describe the detailed modeling for the target ratio $\{R_1 : R_2 : R_3 : R_4 = 19 : 15 : 15 : 15\}$. In the proposed modeling, we define several non-negative integer variables as follows.

Node variables: For every internal/mixing node of the skeleton tree, k node variables $R^l_1(i), R^l_2(i), \dots, R^l_k(i)$ are assigned for denoting the portion of input reagents $R_1, R_2, \dots, R_k$, respectively, where l is the depth of that node and i is the index of the node at depth l. Note that at each depth, the index of a mixing node is assigned from left-to-right fashion starting with 1. For example, the node variables corresponding to the rightmost node of depth 2 in Fig. 8(a) are $R^2_1(3), R^2_2(3), R^2_3(3), R^2_4(3)$.

Reagent variables: We define reagent variables $r^l_j(i)$, where $j = 1, 2, \dots, k$, to denote the number of segments filled with reagent R_j, where l is the depth of the mixing node where reagent R_j is used, and i is the index of the mixing node. For example, the reagent variables corresponding to the rightmost node on depth 2 in Fig. 8(a) are $r^2_1(3), r^2_2(3), r^2_3(3), r^2_4(3)$.

Segment-sharing variables: An edge (u, v) between two mixing nodes denotes the sharing of *CF* from node u to node v in the mixing tree. An edge variable w_{l_1,i_1,l_2,i_2} denotes the number of segments shared between a node with index i_1 at depth l_1, and a node with index i_2 at depth l_2. For example, edge weight $w_{3,3,2,3}$ in Fig. 8(a) represents the number of segments shared from the node indexed as 3 at depth 3 (the rightmost node at depth 3) to the node with index 3 at depth 2 (the rightmost node at depth 2).

For simplicity, we assume that intermediate *CF*s are shared between two adjacent depths only. This restriction can be relaxed by suitably modifying the constraints for each mixing

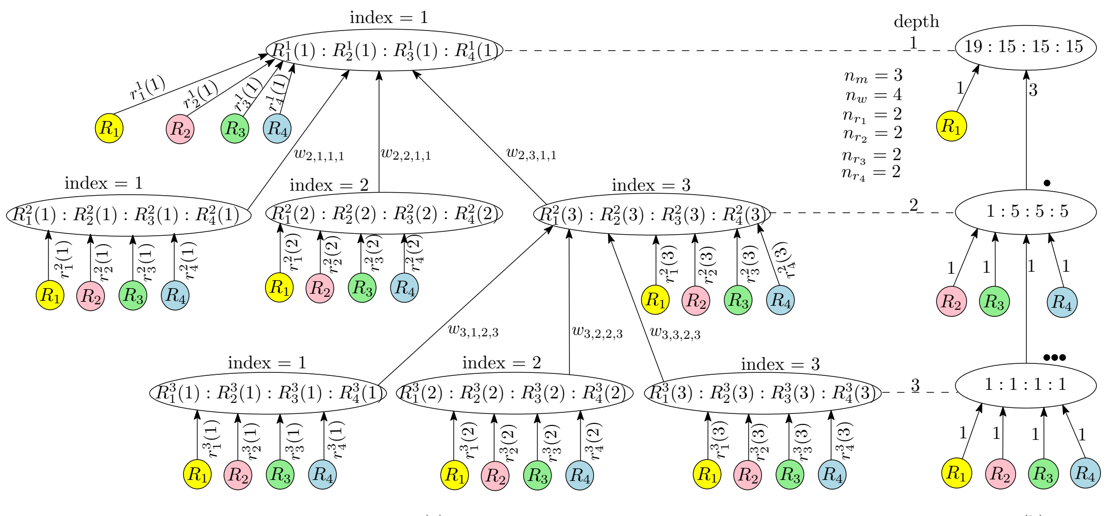
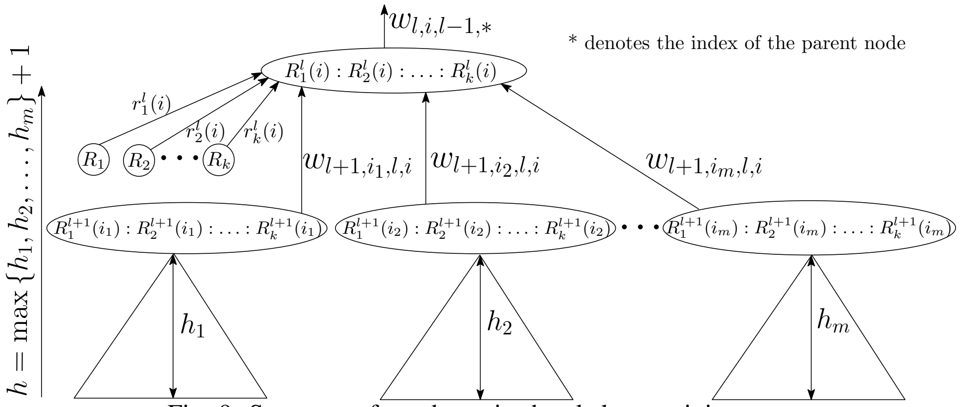


Fig. 8: (a) Skeleton tree obtained from the mixing tree (Fig. 7) (b) Reagent-saving mixing tree obtained by *FloSPA-M*.



$w_{l,i,l-1,*}$
* denotes the index of the parent node
$R_1^l(i) : R_2^l(i) : \ldots : R_k^l(i)$
$w_{l+1,i_1,l,i}$
$w_{l+1,i_2,l,i}$
$w_{l+1,i_m,l,i}$
$h = \max\{h_1, h_2, \ldots, h_m\} + 1$
h_1
h_2
h_m

Fig. 9: Structure of a subtree in the skeleton mixing tree.

node. In order to ensure the correctness of the mixing ratio generated at each mixing node, let us consider a mixing node (l, i) (denoting the mixing node at depth l with index i) that has k reagent nodes as leaves, and m internal mixing nodes (at depth $l+1$) as its children. Fig. 9 shows the portion of the subtree rooted at (l, i). Let the height of m subtrees of the mixing node (l, i) be $h_1, h_2, \ldots, h_m$. Hence, the height of the entire subtree shown in Fig. 9 is $h = \max\{h_1, h_2, \ldots, h_m\}+1$. The desired ratio $R_1^l(i) : R_2^l(i) : \ldots : R_k^l(i)$ can be calculated with the following non-linear equations, which can be transformed into a set of equivalent linear equations.

$$\begin{aligned}
N^{h-2} r_1^l(i) + \sum_{j\in\{i_1,i_2,\ldots,i_m\}} N^{h-h_j-1} w_{l+1,j,l,i} R_1^{l+1}(j) &= R_1^l(i)\\
N^{h-2} r_2^l(i) + \sum_{j\in\{i_1,i_2,\ldots,i_m\}} N^{h-h_j-1} w_{l+1,j,l,i} R_2^{l+1}(j) &= R_2^l(i)\\
\ddots \qquad\qquad\qquad\qquad &\quad \vdots\\
N^{h-2} r_k^l(i) + \sum_{j\in\{i_1,i_2,\ldots,i_m\}} N^{h-h_j-1} w_{l+1,j,l,i} R_k^{l+1}(j) &= R_k^l(i)
\end{aligned} \tag{5}$$

Moreover, we need to ensure that either all N input segments for a Mixer-N are filled with intermediate fluids/input reagents, or the concerned mixing operations are not necessary for generating the desired target ratio. In the second case, the number of mixing operations is reduced. A Mixer-N can provide fluids to at most N segments of other mixers. The required consistency conditions are encoded as follows:

$$\begin{aligned}
&\left(\sum_{j=1}^{k} r_j^l(i) + \sum_{j\in\{i_1,i_2,\ldots,i_m\}} w_{l+1,j,l,i} = N\right) \bigvee\\
&\left(\sum_{j=1}^{k} r_j^l(i) + \sum_{j\in\{i_1,i_2,\ldots,i_m\}} w_{l+1,j,l,i} = 0\right),\\
&0 \le w_{l,i,l-1,*} \le N
\end{aligned} \tag{6}$$

All weights must satisfy $0 \le w_{l+1,j,l,i} \le N-1$, for $j \in \{i_1, i_2, \ldots, i_m\}$. Analogously, $0 \le r_j^l(i) \le N-1$ for $j = 1, 2, \ldots, k$.

The following example illustrates the consistency constraints for the root node of Fig. 8(a).

Example 14. The root node of the tree (Fig. 8(a)) has three incoming edges from other internal nodes. The depth of its subtrees are 2, 2, and 3 in the left-to-right order. Moreover, the depth of the complete tree, i.e., h is 4. The consistency conditions for the root node in Fig. 8(a) are given as follows.

$$4^2 r_1^1(1) + 4w_{2,1,1,1}R_1^2(1) + 4w_{2,2,1,1}R_1^2(2) + w_{2,3,1,1}R_1^2(3) = R_1^1(1)$$
$$4^2 r_2^1(1) + 4w_{2,1,1,1}R_2^2(1) + 4w_{2,2,1,1}R_2^2(2) + w_{2,3,1,1}R_2^2(3) = R_2^1(1)$$
$$4^2 r_3^1(1) + 4w_{2,1,1,1}R_3^2(1) + 4w_{2,2,1,1}R_3^2(2) + w_{2,3,1,1}R_3^2(3) = R_3^1(1)$$
$$4^2 r_4^1(1) + 4w_{2,1,1,1}R_4^2(1) + 4w_{2,2,1,1}R_4^2(2) + w_{2,3,1,1}R_4^2(3) = R_4^1(1)$$
$$\left(r_1^1(1) + r_2^1(1) + r_3^1(1) + r_4^1(1) + w_{2,1,1,1} + w_{2,2,1,1} + w_{2,3,1,1} = 4\right) \bigvee$$
$$\left(r_1^1(1) + r_2^1(1) + r_3^1(1) + r_4^1(1) + w_{2,1,1,1} + w_{2,2,1,1} + w_{2,3,1,1} = 0\right)$$

Additionally, $0 \leq r_1^1(1), r_2^1(1), r_3^1(1), r_4^1(1) \leq 3$ and $0 \leq w_{2,1,1,1}, w_{2,2,1,1}, w_{2,3,1,1} \leq 3$. ■

D. Reagent-saving mixing algorithm

In summary, for a given $(\mathcal{M}, \epsilon, N)$, a mixing tree is first generated using *genMixing*. In the next step, the skeleton tree is created. An *SMT* instance for the skeleton tree is then generated as discussed in Section VII-C. All nonlinearities in the constraints are removed using certain transformations as described earlier. Let the set of constraints be M. Next, we check the existence of a mixing tree that takes N units of reagents. Thus, we add $\sum_{x \in \text{set of all reagent variables}} x = N$ to M, and obtain M'. If M' is *satisfiable*, a solution is found. Otherwise, we need to check for the existence of a mixing tree assuming the allowance for one additional unit of reagent, and re-check for satisfiability. The selection of additional reagent is automatically decided by the *SAT*-solver based on the underlying constraints. This process is continued until M' is found to be *satisfiable*. Algorithm 5 summarizes the steps needed in the Flow-based Sample Preparation Algorithm for Mixtures (*FloSPA-M*).

Algorithm 5: *FloSPA-M*$(\mathcal{M}, \epsilon, N)$

```
Input: $\mathcal{M}$: target mixture, $\epsilon$: accuracy, $N$: Mixer-$N$
Output: Mixing Tree
1  $T$ = genMixing($\mathcal{M}, \epsilon, N$);
2  $T_1$ = Skeleton tree of $T$ after adding reagent nodes;
   /* Detailed modeling and removal of non-linearity are
      discussed in Section VII-C and Section VI-B,
      respectively                                          */
3  $M$ = SMT instance generated from $T_1$ after removing non-linearity;
   /* At least $N$ reagent units are required for filling $N$
      segments of a Mixer-$N$                               */
4  total_reagent_unit = $N$;
   /* total_reagent_unit denotes the summation of all input
      reagents                                              */
5  $M' = M \wedge (\sum_{x \in \text{set of all reagent variables}} x = reagent\_unit)$;
6  $checkSAT(M')$;
7  while $M'$ is unsatisfiable do
8      total_reagent_unit = total_reagent_unit + 1;
9      $M' = M \wedge (\sum_{x \in \text{Set of all reagent variables}} x = total\_reagent\_unit)$;
10     $checkSAT(M')$;
11 Obtain mixing tree from satisfiable assignments of $M'$;
12 return mixing tree;
```

Example 15. The skeleton tree derived from *genMixing* and the mixing tree for *FloSPA-M* are shown in Fig. 8(a) and Fig. 8(b), respectively. Note that the number of mixing steps and waste segments are 7 and 18 for *genMixing* (Fig. 7), whereas *FloSPA-M* produces the same mixture (Fig. 8(b)) with only 3 mixing steps and 4 waste segments. Moreover, the total reagent consumption in *FloSPA-D* (8) is significantly less than that required in *genMixing* (22). ■

We further improve the *FloSPA-M* algorithm so that the intermediate ratios produced at different nodes of the mixing tree can be used not only by its parent but by all ancestors in the mixing tree, thereby enhancing the possibility of sharing intermediate fluids compared to *FloSPA-M*. We call this algorithm as "*FloSPA* for Enhanced Mixing" (*FloSPA-EM*). In *FloSPA-EM*, we add a few additional edges in the skeleton tree that represent possible sharing of intermediate fluids between a node and its ancestors in the skeleton tree while constructing the "enhanced skeleton tree". For implementing of *FloSPA-EM*, only line-2 of Algorithm 5 needs to be modified by including "enhanced skeleton tree" in place of skeleton tree, keeping the rest of the codes unchanged.

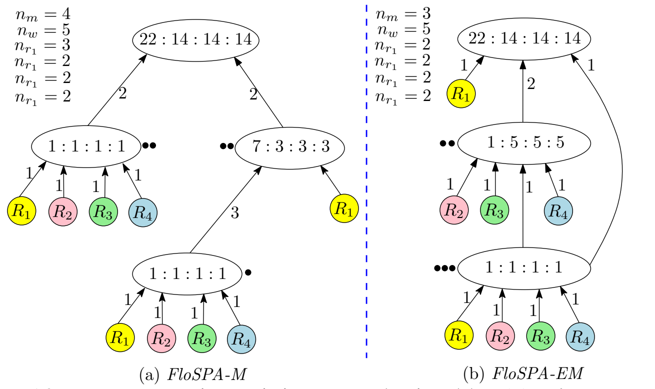


Fig. 10: Reagent saving mixing tree obtained by (a) *FloSPA-M* and (b) *FloSPA-EM*

Example 16. When we run *FloSPA-M* and *FloSPA-EM* for the target ratio $\{R_1 : R_2 : R_3 : R_4 = 22 : 14 : 14 : 14\}$, *FloSPA-EM* performs better compared to *FloSPA-M*. The corresponding mixing trees for *FloSPA-M* and *FloSPA-EM* are shown in Fig. 10(a) and Fig. 10(b), respectively. ■

VIII. Experimental evaluation

We have implemented, in Python, the proposed dilution algorithms *FloSPA-D* and *NWayMix* and compared their performance with *VOSPA* [18] and *TPG* [20]. We have also evaluated the proposed mixture preparation algorithms *genMixing, FloSPA-M*, and *FloSPA-EM*. The clauses are represented using the Python library *Z3Py* that can handle linear constraints with Boolean connectives for the *SMT*-solver *Z3* [22]. As an implementation platform, we have used 2 GHz Intel Core i5 computer with 8 GB memory running 64-bit Ubuntu 14.04 operating system.

A. Performance evaluation for dilution

We have compared the proposed dilution technique with two state-of-the-art sample preparation algorithms (*VOSPA* [18], *TPG*[1] [20]), and also with the baseline algorithm *NWayMix* using Mixer-N. We have conducted three sets of experiments. In the first set, we use a 4-segment mixer (Mixer-4) to generate all CFs ranging from $\frac{1}{256}$ to $\frac{255}{256}$ (i.e., $d = 4, N = 4$) and

[1] *TPG* uses unequal-segmented rotary mixer (Ring-N), whereas *VOSPA, NWayMix* and *FloSPA-D* use equally-segmented rotary mixer (Mixer-N). In our experiments, unequal segments of Ring-N are normalized with respect to the segments in Mixer-N. Additionally, in order to obtain the equivalent mixing models, we use Ring-3, Ring-4 in *TPG* corresponding to Mixer-4 and Mixer-8 used in *VOSPA, NWayMix* and *FloSPA-D*, respectively.

TABLE III: Performance of *FloSPA-D*

CF range: $\frac{1}{4^4}$ to $\frac{255}{4^4}$; $(N = 4, d = 4)$				
Method	$\bar{n}_m$	$\bar{n}_w$	$\bar{n}_s$	$\bar{n}_b$
FloSPA-D (Mixer-4)	3.68	3.67	3.35	4.32
NWayMix (Mixer-4)	3.68 (0%)	8.04 (54%)	6.02 (44%)	6.02 (28%)
VOSPA (Mixer-4)	4.13 (11%)	6.70 (45%)	3.34 (-0.3%)	7.36 (41%)
TPG (Ring-3)	4.98 (26%)	7.24 (49%)	3.77 (11%)	7.47 (42%)
CF range: $\frac{1}{4^5}$ to $\frac{1023}{4^5}$; $(N = 4, d = 5)$				
Method	$\bar{n}_m$	$\bar{n}_w$	$\bar{n}_s$	$\bar{n}_b$
FloSPA-D (Mixer-4)	4.67	4.41	3.45	4.96
NWayMix (Mixer-4)	4.67 (0%)	11.01 (60%)	7.51 (54%)	7.51 (34%)
VOSPA (Mixer-4)	5.38 (13%)	8.98 (51%)	3.47 (1%)	9.51 (48%)
TPG (Ring-3)	6.49 (28%)	9.96 (56%)	3.97 (13%)	9.99 (50%)
CF range: $\frac{1}{8^3}$ to $\frac{511}{8^3}$; $(N = 8, d = 3)$				
Method	$\bar{n}_m$	$\bar{n}_w$	$\bar{n}_s$	$\bar{n}_b$
FloSPA-D (Mixer-8)	2.86	5.73	5.64	8.08
NWayMix (Mixer-8)	2.86 (0%)	13.04 (56%)	10.52 (46%)	10.52 (23%)
VOSPA (Mixer-8)	3.26 (12 %)	11.26 (49%)	5.53 (-1%)	13.73 (41%)
TPG (Ring-4)	4.12 (31%)	10.80 (47%)	6.66 (15%)	12.15 (33%)

%-improvement $= \frac{X - \text{Proposed}}{X} \times 100$, $X =$*NwayMix, VOSPA, TPG*

report the average number of mixing steps ($\bar{n}_m$), the average number of *sample* ($\bar{n}_s$) and *buffer* ($\bar{n}_b$) units consumed, and the average number of wasted units ($\bar{n}_w$) while producing a target CF. These parameters are shown in Table III in which %-improvement of *FloSPA-D* compared to other methods are shown in parenthesis. The formula used for calculating savings is shown in the footnote of Table III. In the remaining two sets of experiments, we increase the accuracy (ϵ) of the target CFs and use more general mixing model, i.e., with a larger value of N. The experiments are repeated using Mixer-4 and Mixer-8 for each CFs ranging from $\frac{1}{1024}$ to $\frac{1023}{1024}$ (i.e., $d = 5, N = 4$) and $\frac{1}{512}$ to $\frac{511}{512}$ (i.e., $d = 3, N = 8$), respectively. The average value for each parameters is shown in Table III. It is evident from Table III that *FloSPA-D* uses minimum mixing steps as guaranteed by *NWayMix* and consumes fewer units of sample compared to *TPG*. Note that with respect to *VOSPA*, *sample* consumption in *FloSPA-D* is comparable. Furthermore, *FloSPA-D* consumes fewer *buffer* units, and thus, it outperforms all previous methods in terms of waste generation.

TABLE IV: CPU-time needed by *FloSPA-D*.

CPU time	Number of targets		
(t in seconds)	$d = 4, N = 4$	$d = 5, N = 4$	$d = 3, N = 8$
$0 \le t < 1$	231	436	483
$1 \le t < 2$	12	85	15
$2 \le t < 3$	6	87	4
$3 \le t < 4$	1	91	5
$4 \le t < 10$	2	220	3
$10 \le t < 20$	2	61	0
$20 \le t < 30$	1	25	1
$30 \le t < 50$	0	9	0
$50 \le t < 76$	0	7	0
$98 \le t < 126$	0	2	0
Total #target CFs	255	1023	511
Avg. running time	0.53 sec.	4.46 sec.	0.34 sec.

On the same hardware platform, we run all four methods over the full range of CFs. Table IV shows the distribution of CPU times needed to generate the mixing graph for target concentrations that lie between $\frac{1}{256}$ and $\frac{255}{256}$ (with $d = 4, N = 4$), $\frac{1}{1024}$ and $\frac{1023}{1024}$ (with $d = 5, N = 4$), and $\frac{1}{512}$ and $\frac{511}{512}$ (with $d = 3, N = 8$). On the average, *FloSPA-D* takes only 0.53 second for ($d = 4, N = 4$), and 4.46 seconds when the accuracy (ϵ) of each target CF is further increased, i.e., for $d = 5, N = 4$. Moreover, when a more general mixing model is used with Mixer-8, only 0.34 second of CPU time is required on the average to compute the mixing graph for a target CF that lies between $\frac{1}{512}$ and $\frac{511}{512}$ (with $d = 3, N = 8$). Note that CPU-time is highly dependent on the value of d. For practical bioassay applications, a high range of accuracy in CF can be achieved for $4 \le N \le 8$ and $3 \le d \le 5$.

B. Performance evaluation for reagent-saving mixing

We evaluate the proposed mixture preparation technique extensively on randomly generated synthetic test cases. We choose two datasets with different number of input reagents (Dataset 1 (Dataset 2) that consists of three (four) input reagents), and each data-set contains 100 randomly generated target ratios. We run *genMixing*, *FloSPA-M*, and *FloSPA-EM* for all target ratios with two different mixing models supported by Mixer-4 and Mixer-8. Table V and Table VI show comparative results where savings for different parameters in *FloSPA-M* and *FloSPA-EM* are calculated with respect to the average value of those for *genMixing*, and the CPU-time needed. Note that the running time for *genMixing* is negligibly small in most of the cases and can be ignored. Additionally, in Table V and Table VI, the standard deviation is shown using parenthesis along with the average value of each parameter.

In the case of Dataset 1, the depth of the mixing tree (d) is kept fixed, and hence the accuracy in *CF* is increased in Mixer-8 compared to Mixer-4. The mean and standard deviation of different parameters are listed in Table V.Note that the running time for Mixer-8 increases compared to Mixer-4 as the former provides higher accuracy in *CF*. In the case of Dataset 2 the error-tolerance (ϵ) is kept invariant, and hence, the depth of the mixing tree corresponding to Mixer-8 decreases compared to Mixer-4. Table VI shows the mean and standard deviation of different parameters. Compared to Mixer-4, the running time for Mixer-8 reduces because of the smaller value of d. It is evident from simulation results that *FloSPA-M* and *FloSPA-EM* algorithms save valuable reagents significantly compared to *genMixing*. Note that *FloSPA-EM* reduces the number of mixing operations, which, in turn, further reduces the amount of input reagents. However, it takes larger CPU-time compared to *FloSPA-M*, as the search space increases.

C. Performance of FloSPA for real-life dilution/mixing ratios

We consider several real-life test cases where some specific dilution and mixing ratios are used. For example, a sample of 70% ethanol is required in Glucose Tolerance Test in mice, and in E.coli Genomic DNA Extraction [30]. Also, 95% ethanol is required for total RNA extraction from worms. Assuming $\epsilon = 0.001$ and $N = 4$, 70% and 90% ethanol can be approximated as $\frac{179}{4^4}$ and $\frac{243}{4^4}$, respectively. Hence, the depth of dilution tree $d = 4$. A sample of 10% $(\frac{26}{4^4})$ Fetal Bovine Serum (FBS) is required for in-vitro culture of human Peripheral Blood Mononuclear Cells (PBMCs) [30]. A mixture of three reagents is required in "One step miniprep method" [7] and

TABLE V: Performance evaluation of mixing algorithms for fixed depth $(d = 3)$ of the mixing tree

Dataset 1: # input reagents = 3, # test cases = 100; Mixing model: Mixer-4																			
genMixing $(\epsilon = 0.007)$						*FloSPA-M* $(\epsilon = 0.007)$							*FloSPA-EM* $(\epsilon = 0.007)$						
n_m	n_w	n_{r_1}	n_{r_2}	n_{r_3}	n_r	n_m	n_w	n_{r_1}	n_{r_2}	n_{r_3}	n_r	Time (in sec.)	n_m	n_w	n_{r_1}	n_{r_2}	n_{r_3}	n_r	Time (in sec.)
3.87	8.61	4.9	3.95	3.76	12.61	3.51	5.12	3.97	2.78	2.37	9.12	0.3	3.19	3.81	3.69	2.36	1.76	7.81	0.45
(0.75)	(2.24)	(1.55)	(1.07)	(1.07)	(2.24)	(0.52)	(1.32)	(1.12)	(1.01)	(0.98)	(1.32)	(0.2)	(0.38)	(0.42)	(1.34)	(1.02)	(0.82)	(0.81)	(1.34)
% savings w.r.t *genMixing*						9%	40%	19%	30%	37%	28%	-	18%	56%	25%	40%	53%	38%	-
Dataset 1: # input reagents = 3, # test cases = 100; Mixing model: Mixer-8																			
genMixing $(\epsilon = 0.0009)$						*FloSPA-M* $(\epsilon = 0.0009)$							*FloSPA-EM* $(\epsilon = 0.0009)$						
n_m	n_w	n_{r_1}	n_{r_2}	n_{r_3}	n_r	n_m	n_w	n_{r_1}	n_{r_2}	n_{r_3}	n_r	Time (in sec.)	n_m	n_w	n_{r_1}	n_{r_2}	n_{r_3}	n_r	Time (in sec.)
3.8	19.59	10.95	9.07	7.57	27.59	3.25	9.04	8.01	5.37	3.66	17.04	3.45	2.99	6.65	7.12	4.51	3.02	14.65	5.89
(0.77)	(5.36)	(2.97)	(2.96)	(3.2)	(5.36)	(0.5)	(2.65)	(2.2)	(2.14)	(1.83)	(2.65)	(5.3)	(0.22)	(2.02)	(1.49)	(1.62)	(1.21)	(2.02)	(5.81)
% savings w.r.t *genMixing*						14%	54%	27%	41%	52%	38%	-	21%	66%	35%	50%	60%	47%	-

n_m: number of mixing, n_w: number of waste segments, $n_{r_1}, n_{r_2}, n_{r_3}$: number of segments filled up with reagents R_1, R_2, and R_3, respectively, n_r: Total number of segments filled up with input reagents

TABLE VI: Performance evaluation for proposed mixing algorithms for fixed error tolerance limit $(\epsilon = 0.001)$

Dataset 2: # input reagents = 4, # test cases = 100; Mixing model: Mixer-4																						
genMixing $(d = 4)$							*FloSPA-M* $(d = 4)$								*FloSPA-EM* $(d = 4)$							
n_m	n_w	n_{r_1}	n_{r_2}	n_{r_3}	n_{r_4}	n_r	n_m	n_w	n_{r_1}	n_{r_2}	n_{r_3}	n_{r_4}	n_r	Time†	n_m	n_w	n_{r_1}	n_{r_2}	n_{r_3}	n_{r_4}	n_r	Time†
5.52	13.55	6.06	5.15	4.28	2.05	17.55	5.12	9.58	5.22	3.82	3.20	1.34	13.58	6.30	4.84	8.01	4.88	3.33	2.60	1.20	12.01	20.72
(0.97)	(2.90)	(2.02)	(1.81)	(1.66)	(0.76)	(2.90)	(0.90)	(1.61)	(1.53)	(1.33)	(1.46)	(0.50)	(1.61)	(20.13)	(0.56)	(1.07)	(1.30)	(1.16)	(1.09)	(0.43)	(1.07)	(19.05)
% savings w.r.t *genMixing*							7%	29%	14%	26%	25%	35%	23%	-	12%	41%	19%	35%	39%	41%	32%	-
Dataset 2: # input reagents = 4, # test cases = 100; Mixing model: Mixer-8																						
genMixing $(d = 3)$							*FloSPA-M* $(d = 3)$								*FloSPA-EM* $(d = 3)$							
n_m	n_w	n_{r_1}	n_{r_2}	n_{r_3}	n_{r_4}	n_r	n_m	n_w	n_{r_1}	n_{r_2}	n_{r_3}	n_{r_4}	n_r	Time†	n_m	n_w	n_{r_1}	n_{r_2}	n_{r_3}	n_{r_4}	n_r	Time†
4.03	21.21	10.27	8.15	6.93	3.86	29.21	3.27	11.29	8.51	5.3	4.08	1.4	19.29	2.74	3.42	10.49	8.54	5.35	3.37	1.23	18.49	6.94
(0.74)	(5.21)	(3.28)	(3.38)	(2.95)	(1.87)	(5.21)	(0.47)	(1.9)	(2.3)	(2.13)	(2.1)	(0.62)	(1.9)	(3.05)	(0.51)	(1.89)	(2.27)	(2.08)	(1.87)	(0.42)	(1.89)	(7.14)
% savings w.r.t *genMixing*							19%	47%	17%	35%	41%	64%	34%	-	15%	50%	17%	34%	51%	68%	37%	-

n_m: number of mixing, n_w: number of waste segments, $n_{r_1}, n_{r_2}, n_{r_3}, n_{r_4}$: number of segments filled up with reagents R_1, R_2, R_3, and R_4, respectively, n_r: Total number of segments filled up with input reagents
† Time is measured in seconds

also for the preparation of plasmid DNA by alkaline lysis with SDS-minipreparation [8]. The corresponding mixing ratios are: $\{R_1 : R_2 : R_3 = 50\% : 48\% : 2\%\}$ (approximated as $\{128 : 122 : 6\}$ for $d = N = 4$) and $\{R_1 : R_2 : R_3 = 22\% : 44\% : 34\%\}$ (approximated as $\{56 : 113 : 87\}$ for $N = d = 4$), respectively. Moreover, a target ratio $\{R_1 : R_2 : R_3 : R_4 : R_5 = 40\% : 10\% : 1\% : 1\% : 48\%\}$ is used in "Splinkerette PCR method" [7]. Given $\epsilon = 0.001$, the target ratio can be approximated as $\{102 : 26 : 2 : 2 : 124\}$ for Mixer-4 by choosing $d = 4$. Table VII shows the performance of *FloSPA* on these real-life examples compared to other methods.

TABLE VII: Performance of *FloSPA* for real-life test cases

Real-life dilution: $\epsilon = 0.001, d = 4$																	
CF	*NWayMix* (Mixer-4)				*FloSPA-D* (Mixer-4)					*VOSPA* (Mixer-4)				*TPG* (Ring-3)			
	n_m	n_w	n_s	n_b	n_m	n_w	n_s	n_b	Time	n_m	n_w	n_s	n_b	n_m	n_w	n_s	n_b
$\frac{26}{4^4}$	4	9	5	8	4	3	1	6	0.00 sec.	4	5	1	8	6	8	1	11
$\frac{29}{4^4}$	4	9	5	8	4	5	1	8	0.09 sec.	5	7	1	10	6	7	1	10
$\frac{179}{4^4}$	4	9	8	5	4	3	5	2	1.67 sec.	5	7	4	7	5	6	4	6
$\frac{243}{4^4}$	4	9	9	4	4	4	7	1	0.51 sec.	4	7	7	4	4	7	7	4

Real-life mixture preparation: $\epsilon = 0.001, d = 4$; Mixing model: Mixer-4											
Ratio	*genMixing*			*FloSPA-M*				*FloSPA-EM*			
	n_m	n_w	n_r	n_m	n_w	n_r	Time	n_m	n_w	n_r	Time
$\{56 : 113 : 87\}$	5	12	16	5	10	14	4.56 sec.	4	5	9	10.03 sec.
$\{128 : 122 : 6\}$	4	9	13	4	6	10	0.25 sec.	4	6	10	1.00 sec.
$\{102 : 26 : 2 : 2 : 124\}$	7	18	22	6	10	14	8.25 sec.	6	8	12	30.09 sec.

n_m: number of mixing, n_w: number of waste segments, $n_s(n_b)$: number of segments filled up with *sample* (*buffer*), n_r: Total number of segments filled up with input reagents

IX. Conclusion and future directions

In this work, we have addressed the problem of automated sample preparation using flow-based microfluidic biochips and proposed a dilution algorithm that minimizes reagent-usage under the constraint of deploying minimum mixing steps. Additionally, two algorithms for reagent-saving mixture-preparation using flow-based biochips have been developed. We use an N-segment rotary mixer for implementing a generalized mixing model. The proposed algorithms utilize an *SMT*-based solving engine that is capable of handling both dilution and mixing problems within the same framework. Although for mixture preparation, we have assumed the same cost for each of the input reagents, the proposed modeling can be modified for finding solutions when different cost metrics are assigned to them. It may also be worthwhile to expand the scope of these algorithms by addressing several error management issues concerning imprecise metering that affects fluid flow. Further, one may explore related issues in algorithmic sample preparation for other hybrid mixer architectures, e.g., Ring-*M* [20] in conjunction with Mixer-*N*.

References

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

[2] W. Thies, J. P. Urbanski, T. Thorsen, and S. P. Amarasinghe, "Abstraction layers for scalable microfluidic biocomputing," *Natural Computing*, vol. 7, no. 2, pp. 255–275, 2008.

[3] S. Roy, B. B. Bhattacharya, and K. Chakrabarty, "Optimization of dilution and mixing of biochemical samples using digital microfluidic biochips," *IEEE Trans. on CAD*, vol. 29, no. 11, pp. 1696–1708, 2010.

[4] C.-H. Liu, T.-W. Chiang, and J.-D. Huang, "Reactant minimization in sample preparation on digital microfluidic biochips," *IEEE Trans. on CAD*, vol. 34, no. 9, pp. 1429–1440, 2015.

[5] S. Roy, B. B. Bhattacharya, S. Ghoshal, and K. Chakrabarty, "Theory and analysis of generalized mixing and dilution of biochemical fluids using digital microfluidic biochips," *ACM JETC*, vol. 11, no. 1, pp. 2.1–2.33, 2014.

[6] D. Mitra, S. Roy, S. Bhattacharjee, K. Chakrabarty, and B. B. Bhattacharya, "On-chip sample preparation for multiple targets using digital microfluidics," *IEEE Trans. on CAD*, vol. 33, no. 8, pp. 1131–1144, 2014.

[7] S. Roy, P. P. Chakrabarti, S. Kumar, K. Chakrabarty, and B. B. Bhattacharya, “Layout-aware mixture preparation of biochemical fluids on application-specific digital microfluidic biochips,” *ACM TODAES*, vol. 20, no. 3, pp. 45.1–45.34, 2015.
[8] Y.-L. Hsieh, T.-Y. Ho, and K. Chakrabarty, “Biochip synthesis and dynamic error recovery for sample preparation using digital microfluidics,” *IEEE Trans. on CAD*, vol. 33, no. 2, pp. 183–196, 2014.
[9] S.-Y. Teh, R. Lin, L.-H. Hung, and A. P. Lee, “Droplet microfluidics,” *Lab Chip*, vol. 8, pp. 198–220, 2008.
[10] D. Mark, S. Haeberle, G. Roth, F. von Stetten, and R. Zengerle, “Microfluidic lab-on-a-chip platforms: Requirements, characteristics and applications,” *Chem. Soc. Rev.*, vol. 39, pp. 1153–1182, 2010.
[11] C. D. Chin, V. Linder, and S. K. Sia, “Commercialization of microfluidic point-of-care diagnostic devices,” *Lab Chip*, vol. 12, pp. 2118–2134, 2012.
[12] S. W. Dutse and N. A. Yusof, “Microfluidics-based lab-on-chip systems in DNA-based biosensing: An overview,” *Lab Chip*, vol. 11, pp. 5754–5768, 2011.
[13] P. Neuži, S. Giselbrecht, K. Länge, T. J. Huang, and A. Manz, “Revisiting lab-on-a-chip technology for drug discovery,” *Nat. Rev. Drug Discovery*, vol. 11, pp. 620–632, 2012.
[14] Y. Sun, T. L. Quyen, T. Q. Hung, W. H. Chin, A. Wolffa, and D. D. Bang, “A lab-on-a-chip system with integrated sample preparation and loop-mediated isothermal amplification for rapid and quantitative detection of *salmonella* spp. in food samples,” *Lab Chip*, vol. 15, pp. 1898–1904, 2015.
[15] J. Ducrée, R. Zengerle, and J. Newman, *FlowMap: Microfluidics Roadmap for the Life Sciences*. FlowMap Consortium, 2004. [Online]. Available: https://microfluidics-roadmap.com
[16] C. Kim, K. Lee, J. H. Kim, K. S. Shin, K.-J. Lee, T. S. Kim, and J. Y. Kang, “A serial dilution microfluidic device using a ladder network generating logarithmic or linear concentrations,” *Lab-on-a-Chip*, vol. 8, no. 3, pp. 473–479, 2008.
[17] C. Neils, Z. Tyree, B. Finlayson, and A. Folch, “Combinatorial mixing of microfluidic streams,” *Lab-on-a-Chip*, vol. 4, no. 4, pp. 342–350, 2004.
[18] C.-M. Huang, C.-H. Liu, and J.-D. Huang, “Volume-oriented sample preparation for reactant minimization on flow-based microfluidic biochips with multi-segment mixers,” in *Proc. of DATE*, 2015, pp. 1114–1119.
[19] J. Melin and S. Quake, “Microfluidic large-scale integration: The evolution of design rules for biological automation,” *Annual Reviews in Biophysics and Biomolecular Structure*, vol. 36, pp. 213–231, 2007.
[20] C.-H. Liu, K.-C. Shen, and J.-D. Huang, “Reactant minimization for sample preparation on microfluidic biochips with various mixing models,” *IEEE Trans. on CAD*, vol. 34, no. 12, pp. 1918–1927, 2015.
[21] C.-Y. Lee, C.-L. Chang, Y.-N. Wang, and L.-M. Fu, “Microfluidic mixing: A review,” *Int. J. Mol. Sci.*, vol. 12, no. 5, pp. 3263–3287, 2011.
[22] L. M. de Moura and N. Bjørner, “Z3: An efficient SMT solver,” in *Proc. of TACAS*, 2008, pp. 337–340, [Z3 is available at https://github.com/Z3Prover/z3].
[23] P. Pop, I. Araci, and K. Chakrabarty, “Continuous-flow biochips: Technology, physical-design methods, and testing,” *IEEE Design & Test*, vol. 32, no. 6, pp. 8–19, 2015.
[24] H.-P. Chou, M. A. Unger, and S. R. Quake, “A microfabricated rotary pump,” *Biomedical Microdevices*, vol. 3, no. 4, pp. 323–330, 2001.
[25] R. Nieuwenhuis, A. Oliveras, and C. Tinelli, “Solving SAT and SAT Modulo Theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T),” *J. ACM*, vol. 53, no. 6, pp. 937–977, 2006.
[26] D. Kroening and O. Strichman, *Decision Procedures - An Algorithmic Point of View*. Springer, 2008.
[27] O. Keszocze, R. Wille, T.-Y. Ho, and R. Drechsler, “Exact one-pass synthesis of digital microfluidic biochips,” in *Proc. of DAC*, 2014, pp. 1–6.
[28] O. Keszocze, R. Wille, and R. Drechsler, “Exact routing for digital microfluidic biochips with temporary blockages,” in *Proc. of ICCAD*, 2014, pp. 405–410.
[29] L. M. de Moura and N. Bjørner, “Satisfiability modulo theories: introduction and applications,” *Commun. ACM*, vol. 54, no. 9, pp. 69–77, 2011.
[30] Bio-protocol. Accessed: April, 2016. [Online]. Available: http://www.bio-protocol.org/

Sukanta Bhattacharjee received the B.Sc. (Hons.) degree in computer science and the B.Tech. degree in computer science and engineering from the University of Calcutta, India, in 2006 and 2009, respectively. He received the M.Tech. degree in computer science from the Indian Statistical Institute, Kolkata, India, in 2012. He is currently a doctoral student in computer science at the same institute.

His current research interests include algorithms for computer-aided design of microfluidic biochips.

Sudip Poddar received the B.Tech. degree in computer science and engineering from the Maulana Abul Kalam Azad University of Technology (formerly known as West Bengal University of Technology), West Bengal, India, in 2008. He received the M.Tech degree in computer science and engineering from the University of Kalyani, India, in 2012. He is currently a Research Scientist at the Indian Statistical Institute, Kolkata, and a Doctoral candidate at the Indian Institute of Engineering Science and Technology (IIEST), Shibpur, India.

His research interests include computer-aided design for microfluidic lab-on-a-chip and Soft computing.

Sudip Roy (M’14-S’10) received his BSc (honors) degree in Physics and BTech degree in Computer Science and Engineering from the University of Calcutta, India, in 2001 and 2004, respectively, the MS (by research) and PhD degrees in Computer Science and Engineering from the Indian Institute of Technology (IIT) Kharagpur, India, in 2009 and 2014, respectively. Currently, he is an Assistant Professor in the Department of Computer Science and Engineering of Indian Institute of Technology (IIT) Roorkee, India.

His research interests include electronic design automation (EDA) of microfluidic lab-on-a-chips and digital VLSI integrated circuits. He has published in 13 peer-reviewed journals and 20 international conferences in the aforementioned areas. He has authored one book, one book chapter, two granted US patents and one filed US patent.

Junin-Dar Huang (M’96) received the B.S. and Ph.D. degrees in electronics engineering from National Chiao Tung University, Hsinchu, Taiwan, in 1992 and 1998, respectively. He is currently a Professor and the Curriculum Committee Chair in the Department of Electronics Engineering, National Chiao Tung University, Taiwan. He received an Outstanding Teaching Award of the University in 2014.

His research interests include behavioral and logic synthesis, design automation for biochips, and synthesis for single-electron transistor arrays. He received the Best Paper Award in IEEE Computer Society Annual Symposium on VLSI 2011. He was the Technical Program Committee Chair of SASIMI 2015. Dr. Huang is a member of the IEEE, ACM, IEICE, and Phi Tau Phi.

Bhargab B. Bhattacharya (F’07) is currently professor of computer science and engineering and INAE Chair Professor at the Indian Statistical Institute, Kolkata, India. He received the B.Sc. degree in physics from the Presidency College in 1971, the B.Tech. and M.Tech. degrees in radiophysics and electronics in 1974 and 1976, respectively, and the Ph.D. degree in computer science in 1986, all from the University of Calcutta. He held visiting professorship at the University of Nebraska-Lincoln, and at Duke University, USA, at the University of Potsdam, Germany, at the Kyushu Institute of Technology, Iizuka, Japan, at Tsinghua University, Beijing, China, at IIT Kharagpur and IIT Guwahati, India. His research interest includes design and testing of integrated circuits, nano-biochips, digital geometry, and image processing architecture. He has published more than 300 technical articles, and he holds 10 United States Patents. He is a Fellow of the Indian National Academy of Engineering and a Fellow of the National Academy of Sciences (India). He is currently on the editorial board of the *Journal of Electronic Testing: Theory and Applications* (JETTA).