Efficient Regulation of Synthetic Biocircuits Using Droplet-Aliquot Operations on MEDA Biochips*

Mohamed Ibrahim, Zhanwei Zhong, Bhargab B. Bhattacharya, and Krishnendu Chakrabarty

***Abstract*—Microfluidic platforms have recently emerged as an invaluable component for studying synthetic biology as they are capable of emulating complex molecular networks of biological pathways (biocircuits) on a chip. A special type of biochemical assays, known as biocircuit-regulatory scanning (BRS) assays, are employed to regulate gene expression, enabling comprehensive exploration of related biocircuit parameters. Prior work has provided high-level design methodologies for implementing BRS; however, most of these methods are abstract and cannot be used in practice as they overlook the dynamics of interactions between the samples and the biochip. In this paper, we address this limitation by providing a comprehensive framework that implements BRS assays. The proposed framework, named BioScan, includes: (1) a statistical method that selects suitable volumetric ratios of biochemicals used to execute a BRS assay; (2) a high-level synthesis method that generates the specifications of the target BRS assay; (3) a translation technique enabling implementation of BRS on a microelectrode dot-array (MEDA) biochip; (4) a Dirichlet-regressor that constructs the parameter space of the associated biocircuit. Simulation results show that the proposed framework can efficiently perform parameter-space exploration while significantly reducing completion time and reagent cost.**

***Keywords*—Sample Preparation, MEDA Biochip, Synthesis Biology, Droplet Aliquoting, Microfluidics**

I. INTRODUCTION

Synthetic biology has emerged over the past decade with the goal of creating biological parts that can perform new and useful functions. Applications of synthetic biology include environmental monitoring, production of therapeutics, creation of new material, etc [1]. Such applications can be implemented via *synthetic biocircuits* [2]. Several biocircuits have been developed in the laboratories, including but not limited to biologic gates [3], bio oscillators [4] and genetic memory [5].

An example of a biocircuit is shown in Fig. 1, which is used to maintain certain bacterial cell density lower than the limits imposed by the environment (i.e., nutrient supply). Protein "I" (from the LuxI transcriptional regulator) synthesizes a small, diffusible acyl-homoserine lactone (AHL) signalling molecule. When the bacterial cell density increases, AHL accumulates in the experimental medium. At a sufficiently high concentration, AHL molecules will bind and activate the LuxR transcriptional regulator. The outcome protein "R" will activate the expression of a "Killer" gene to produce protein "K". A sufficiently high level of protein "K" causes the death of a bacterial cell.

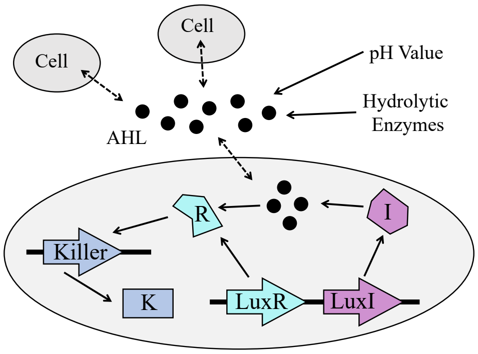


Fig. 1: A population-control circuit based on cell-cell communication (adapted from [6]).

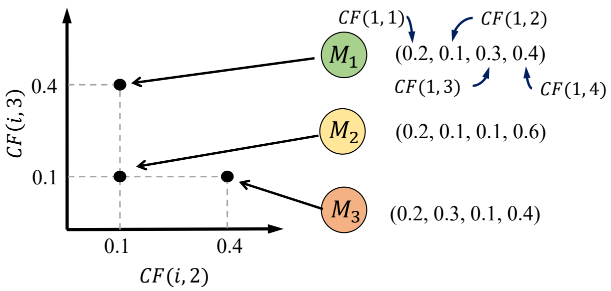


Fig. 2: CF profiles of three mixtures M_1, M_2 and M_3, and the corresponding $\mathcal{CF}$ space.

In order to fully understand the relationship between circuit function and the regulatory parameters, biologists need to carry out *parameter-space exploration* (PSE) [2] by checking all possible combinations of biocircuit parameters. While implementing PSE, a large number of mixtures with different proportions of components (denoted as M_i) are generated. A mixture consists of multiple reagents with a predefined ratio of concentration factors (CFs). The CF of a reagent R_j in mixture M_i is defined as $cf_{(i,j)} = V(R_j)/V(M_i)$, in which $V(M_i)$ is the volume of M_i, and $V(R_j)$ is the volume of R_j in M_i. The combination of CF values for all reagent in a mixture is referred to as a *CF profile*.

The generation of mixtures with certain CF-profiles that exhausts possible combinations under certain constraints (i.e.,

---

*This research was supported in part by the US National Science Foundation under grants CCF-1702596 and ECCS-1914796. A preliminary version of this paper was published in DATE, 2019.

M. Ibrahim is with the University of California, Berkeley, CA, 94720 USA, email: mohamed.ibrahim@berkeley.edu. He worked on this research as a graduate student at Duke University.

Z. Zhong is with Marvell Semiconductor Inc., Santa Clara, CA, 95054 USA, e-mail: zzwzhong698800@gmail.com. He worked on this research as a graduate student at Duke University.

B. B. Bhattacharya is with the CSE Dept., Indian Institute of Technology Kharagpur, WB 721302, India, email: bhargab@cse.iitkgp.ac.in.

K. Chakrabarty is with the Department of Electrical and Computer Engineering, Duke University, Durham, NC, 27705 USA, e-mail: krish@duke.edu.

finite sampling of an infinite space), is referred to as *biocircuit-regulatory scanning* (BRS). For example, to study the parameter space of the biocircuit in Fig. 1, we can generate three mixtures M_1, M_2, and M_3, as shown in Fig. 2. Each mixture consists of four types of reagents: (i) R_1 that modulates the cell density (P_1), (ii) R_2 that modulates the medium pH value (P_2), (iii) R_3 that modulates the concentration of hydrolytic enzymes (P_3), and (iv) distilled water R_4 that is added to keep the droplet volume constant. The CF profiles of three mixtures are shown in Fig. 2. For simplicity, let R_1 have a fixed CF value of 0.2 for all mixtures (i.e., $CF(i, 1) = 0.2, \forall i$), but the CF values of R_2, R_3, and R_4 are different. In this case, the CF profiles of three mixtures are mapped to three points in a 2D $\mathcal{CF}$ space; see Fig. 2. These mixtures are incubated and then analyzed using fluorescence detectors to detect fluorescent protein (i.e., examine the performance of the biocircuit).

A mixture can be represented by a point in $\mathcal{CF}$ space, and by generating numerous mixtures, we are able to explore $\mathcal{CF}$ space and find the optimal range that yields good performance for the biocircuit. However, this approach is cost-prohibitive, especially with the exponential growth in the number of parameters. Therefore, a major challenge is the development of a systematic methodology that enables dense scanning of $\mathcal{CF}$ space. This methodology, moreover, requires an experimental framework that offers fine-grained mixing capabilities to enable biochemical composition of CFs. Recently, a framework based on a flow-based microfluidic biochip has enabled PSE for the biocircuit [7]. Despite the novelty of this design, it suffers from the following drawbacks:

(1) The flow-based solution performs passive mixing, which is much slower than other active mixing schemes [8], [9].

(2) As the number of reagents is varied, new sets of configuration parameters must be computed. Such a process is time-consuming, and it can pose a significant challenge when a large number of reagents is involved.

(3) Because of viscosity and other issues, it may not be convenient to handle dense fluids with low content of distilled water (DW) using a flow-based chip. Thus, some portions of the CF-space cannot be uniformly sampled.

Prior work has provided high-level design methodologies for implementing BRS on micro-electrode-dot-array (MEDA) biochips [7]. However, this method is abstract and cannot be used in practice as it overlooks the dynamics of interactions between the samples and the biochip. In this paper, we address this limitation by presenting a comprehensive framework that implements BRS assays. The proposed framework, named BioScan, includes: (i) a statistical method that selects suitable volumetric ratios of biochemicals used to execute a BRS assay; (ii) a high-level synthesis method that generates the specifications of the target BRS assay; (iii) a physical-level mapping technique for implementing BRS on a microelectrode dot-array (MEDA) biochip; (iv) a Dirichlet-regressor that constructs the parameter space of the associated biocircuit. Simulation results show that the proposed framework can efficiently implement parameter-space exploration while significantly reducing completion time and reagent cost.

The rest of the paper is organized as follows. Section II presents an overview of the MEDA biochip and the droplet-aliquot operation. Section III introduces the proposed BioScan framework and present the sampling method used in the framework. Section IV describes the high-level synthesis method, droplet aliquoting constraints and the problem formulation of the physical-level synthesis problem. Next, Section V introduces the proposed physical-level synthesis method. Simulation results and evaluations are presented in Section VI. Finally, Section VII concludes the paper.

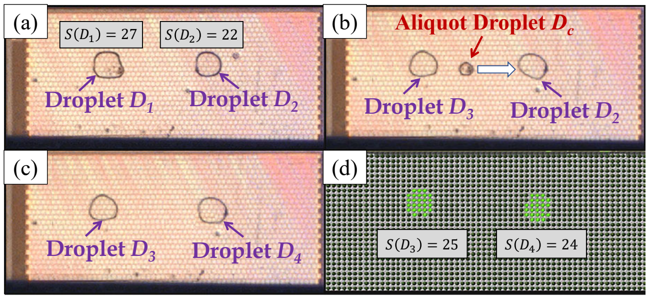


Fig. 3: Experimental demonstration of a droplet-aliquot operation [10]: (a) two droplets are with unequal volumes; (b) aliquot droplet extracted from D_1 combines with D_2; (c) two droplets are with equal volumes.

II. PRELIMINARIES

In this section, we present an overview of MEDA biochips and prior sample-preparation methods.

A. *MEDA Biochips and Droplet Aliquoting*

Micro-Electrode-Dot-Array (MEDA) is a biochip platform that is consisted of micro-electrode arrays. In contrast to conventional Digital Microfluidic Biochips (DMFBs), MEDA biochips can provide real-time capacitive sensing to detect the property (droplet- property sensing) and the location (droplet-location sensing) of the droplet [10], [11]. Droplet aliquot is an operation on the MEDA platform (and not feasible on a conventional DMFB) that allows us to derive a smaller droplet from a larger one. This operation can be used to adaptively recover from erroneous splitting in an efficient way.

For example, suppose a parent droplet is split into two child droplets D_1 and D_2. If the size of D_1 is larger than D_2, a small "aliquot" droplet D_a can be extracted from D_1 and be merged with D_2. As a result, the volumetric difference between D_1 and D_2 can be reduced to an acceptable level. This fluidic operation can only be achieved on MEDA and is referred to as a "droplet-aliquot" operation. It has been validated by experiments, and the illustration of this fluidic operation is shown in Fig. 3. The size of D_1 and D_2 are initially 27 and 22 (the unit is microelectrodes), respectively. Next, an "aliquot" droplet D_a (2 microelectrode in size) is extracted from D_1, and merged with D_2. After that, the resultant size of D_1 and D_2 are 25 and 24, respectively. The volumetric difference is reduced from 5 to 1 after the "droplet aliquot" operation.

B. *Previous Sample-Preparation Methods*

Early research on sample preparation focused on optimizing the dilution process for a single sample with the goal of

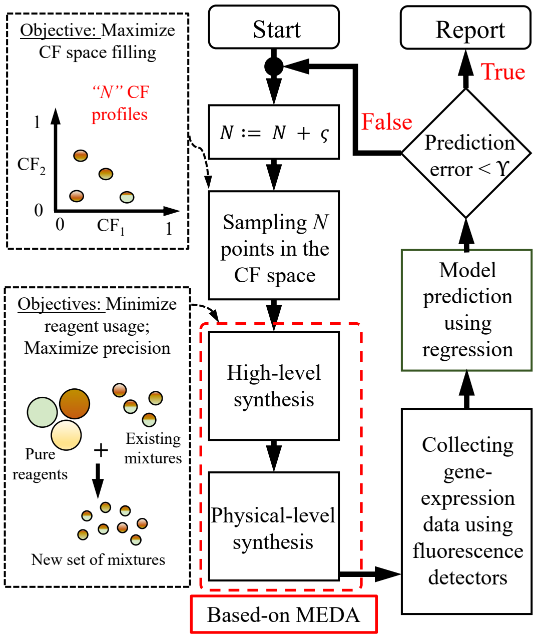

Objective: Maximize CF space filling
"N" CF profiles
1
CF$_2$
0
0
CF$_1$
1
Objectives: Minimize reagent usage; Maximize precision
Pure reagents
+
Existing mixtures
New set of mixtures
Start
Report
True
False
$N := N + \varsigma$
Prediction error < ϒ
Sampling N points in the CF space
Model prediction using regression
High-level synthesis
Physical-level synthesis
Collecting gene-expression data using fluorescence detectors
Based-on MEDA

Fig. 4: An illustration of the BioScan framework. The critical steps are: (1) $\mathcal{CF}$ space sampling; (2) high-level synthesis; (3) physical-level synthesis.

minimizing the amount of waste droplets [12]–[14]. In [10], the process of sample dilution has been optimized using the (M:N) mixing model offered by MEDA. However, these methods are not capable of handling mixtures that contain three or more reagents.

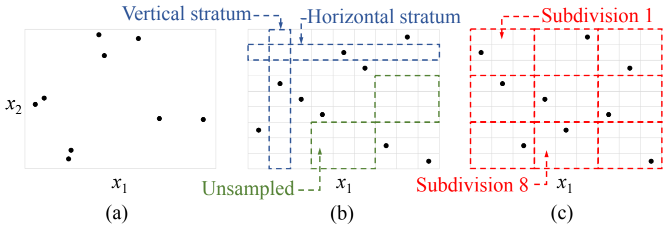
To support dilution gradients in quantitative analysis, multi-target sample-preparation techniques have been introduced [15]–[17]. These techniques generate multiple droplets of the same sample, but with different concentration levels. Each droplet therefore contains only a sample and a buffer solution. However, these methods are limited to (1:1) mixing and they cannot support the preparation of multiple mixtures that constitute a large number of reagents.

For producing a desired multi-reagent mixture, synthesis methods have been developed to generate a bottom-up mixing tree that encodes the successive composition of reagent volumetric ratios [18], [19]. These methods can be easily adapted to our space-exploration problem by running multiple iterations of the algorithm; every iteration specifies the mixing of an individual mixture. This approach, however, may lead to a significant increase in the amount of waste droplets and in the protocol completion time.

To make DMFBs useful for dense PSE in synthetic biology, we need a new top-down synthesis methodology that allows concurrent production of several mixtures with maximum precision, especially in the presence of reagent-usage constraints.

III. SAMPLING OF CONCENTRATION FACTOR SPACE

In this section, we first introduce the proposed BioScan framework. Next, we present a method of stratified sampling and explain the mapping to the $\mathcal{CF}$ space.



(a) (b) (c)

Fig. 5: Sampling nine points in a 2-D space using: (a) random sampling, (b) Latin Hypercube sampling (LHS), (c) Orthogonal Array-based Latin Hypercube Sampling (OA-LHS).

A. The BioScan Framework

An illustration of the BioScan PSE framework is shown in Fig. 4. First, to enable systematic scanning of $\mathcal{CF}$ space, a statistical sampling approach is used to generate N points in $\mathcal{CF}$ space, where N is initially specified by the user. By using this sampling scheme, the scanning process is ensured to cover all regions of $\mathcal{CF}$ space with equal probability (i.e., maximizing space filling). Second, after N CF profiles are sampled from $\mathcal{CF}$ space, an integer linear programming (ILP)-based high-level synthesis method is presented to specify the mixture-production strategy for all the samples on a MEDA biochip. This method can be optimized to reduce the usage of reagents and to maximize the precision of the mixture-production process. Third, the mixture-production strategy is implemented on MEDA using physical-level synthesis and data related to gene-expression analysis for all N samples is collected using on-chip sensors. From a design-automation perspective, it suffices to "simulate" the behavior of gene expression to generate appropriate gene-expression labels. Finally, the obtained data is processed using regression analysis to derive the model that represents the parameter space.

A potential drawback of this iterative approach is that repetitive generation of new sets of mixtures may lead to significant reagent usage; thus leading to an increased cost. A solution to this problem is to generate the new mixtures not only using reagents stored on chip, but also by exploiting mixtures generated during previous iterations of PSE. This approach may reduce reagent usage, which is useful especially in settings where reservoir storage is constrained. This scheme, however, is computationally challenging and therefore it needs to be designed carefully through proper modeling and synthesis.

B. Stratified Sampling: Latin Hypercubes

Consider an rg-dimension continuous space $\mathcal{X}$, in which each point X in the space is defined as $X = \{x_1, x_2, ..., x_{rg}\}$. If we select a total of tm samples $\{X_1, X_2, ..., X_{tm}\} \subset \mathcal{X}$, where $X_i = \{x_{(i,1)}, x_{(i,2)}, ..., x_{(i,rg)}\}$ $(1 \leq i \leq tm)$, the *Euclidean maximin distance* is defined as:

$$E_m(\mathcal{X}) = \min\{E(X_i, X_j) : X_i \neq X_j, X \in \mathcal{X}\} \tag{1}$$

A larger value of E_m indicates that the distance between the closest points provides a better space-filling. We are seeking a collection of tm samples that gives us a larger value of E_m.

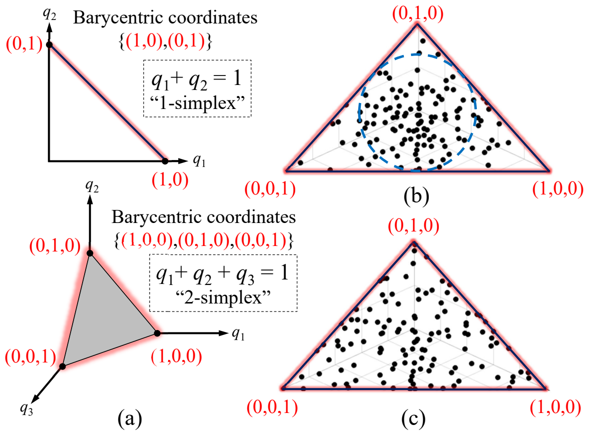


Fig. 6: Mapping OA-LHS-sampled data to simplex $\mathcal{CF}$ space: (a) graphical representation of 1-simplex and 2-simplex; (b) scaling-based mapping leads to poor space filling; (c) Dirichlet-based mapping ($\alpha = 3$) preserves enhanced space filling.

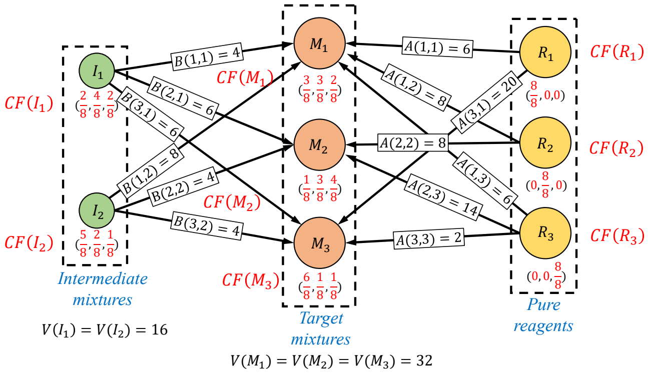


Fig. 7: A high-level synthesis solution to obtain three target mixtures using two intermediate mixtures and three reagents.

It is known that pseudo-random sampling (PRS) methods such as Monte Carlo sampling may result in poor space filling [20]. We overcome this limitation by using a classical *stratified* sampling technique known as *Latin Hypercube sampling* (LHS) [21], which divides the range of each dimension into tm equally probable strata and samples once from each stratum. LHS can be further enhanced by dividing the sampling space into ls equally probable subdivisions, and the value of ls is equal to the number of samples; such a method is known as *Orthogonal Array-based Latin Hypercube Sampling* (OA-LHS) [22]. Fig. 5 shows the PRS, the LHS, the OA-LHS methods for nine samples in a 2-D space (i.e., $rg = 2$, $tm = 9$).

In an rg-dimension $\mathcal{CF}$ space, a total of tm CF profiles are selected for a set of mixtures $\{M_1, M_2, ..., M_{tm}\}$. The CF profiles $\{CF(M_1), CF(M_2), ..., CF(M_{tm})\}$ are defined as $CF(M_i) = \{cf_{(i,1)}, cf_{(i,2)}, ..., cf_{(i,rg)}\}, \forall i \in \{1, 2, ..., tm\}$. However, different from the $\mathcal{X}$ space described above, the sample points (i.e., CF profiles) should satisfy the following two constraints: (1) $0 \leq cf_{(i,j)} < 1, \forall (i,j)$ and (2) $\sum_{j=1}^{rg} cf_{(i,j)} = 1, \forall i$). The $\mathcal{CF}$ space can be graphically represented using a simplex that is formed using a barycentric coordinate system [23]. Fig. 6(a) depicts the shapes of 1-simplex (2-D space) and 2-simplex (3-D space).

However, OA-LHS method can not be directly used in the $\mathcal{CF}$ space. To adapt OA-LHS to the "simplex" $\mathcal{CF}$ space, we seek a mapping function that is defined as $f_m : \mathcal{X} \rightarrow \mathcal{CF}$. A trivial implementation of f_m is to uniformly sample points in the space $\mathcal{X}$ using OA-LHS then re-scale the points using the relation $cf_{(i,j)} = x_{(i,j)} / \sum_{j=1}^{rg} x_{(i,j)}$; this method is referred to as *scaling-based mapping* . However, this approach severely degrades space filling since it tampers with the stratification property; see Fig. 6(b).

We develop an alternative implementation of f_m that does not change the uniformity of the sampling by using the *Dirichlet distribution*, which is an exponential family distribution over a simplex, i.e., positive vectors that sum to one. Formally, if $x_{(i,j)} \in [0, 1)$ is sampled using OA-LHS based on a uniform distribution, then $f_s : x_{(i,j)} \rightarrow cf_{(i,j)}$ can be computed as:

$$y_{(i,j)} = \frac{x_{(i,j)}^{\alpha-1} e^{-x_{(i,j)}}}{\Gamma(\alpha)}; \quad cf_{(i,j)} = \frac{y_{(i,j)}}{\sum_{j=1}^{rg} y_{(i,j)}} \tag{2}$$

The above sampling and mapping processes can be applied to a $\mathcal{CF}$ space with any dimension, i.e., they are scalable. To further improve space filling, we run the above method several times and report the result associated with the largest Euclidean maximin distance (i.e., E_m).

IV. HIGH-LEVEL SYNTHESIS METHOD

BioScan is designed to be an iterative PSE flow that enables composition of new mixtures on a MEDA biochip; new iterations are executed if the accuracy of the constructed model needs to be enhanced. A cost-effective design can generate the new mixtures not only using reagents stored on chip, but also by exploiting mixtures generated during previous iterations.

A. Definitions of the Mixing Model

According to [7], the high-level synthesis solution includes the following three components:

(1) Target Mixtures: A set of tm *mixtures* need to be generated, i.e., $\{M_1, M_2, ..., M_{tm}\}$, and the volume of them are $\{V(M_1), V(M_2), ..., V(M_{tm})\}$, respectively. Note that the CF profile of these target mixtures are sampled from the $\mathcal{CF}$ space using the adaptive OA-LHS method.

(2) Intermediate Mixtures and Reagents: To generate the target mixtures, a set of *intermediate mixtures* generated in a previous iteration and *reagent fluids* are used; see Fig. 7. We assume there are a total of im intermediate mixtures, i.e., $\{I_1, I_2, ..., I_{im}\}$, and the volume of them are $\{V(I_1), V(I_2), ..., V(I_{im})\}$, respectively. We also assume there are a total of rg reagents, i.e., $\{R_1, R_2, ..., R_{rg}\}$, and the volume of them are unlimited since they can be refilled in the reservoir.

(3) Aliquots: Utilizing droplet-aliquot operations, an aliquot of volume $B(i,k)$ from an intermediate mixture I_k contributes to the generation of a target mixture M_i. Similarly, an aliquot of volume $A(i,j)$ from a reagent R_j contributes to the generation of the same target mixture M_i. Note that the lower bounds on $B(i,k)$ and $A(i,j)$, denoted by $B_{min}(i,k)$ and $A_{min}(i,j)$, respectively, are controlled by the aliquoting constraints imposed by MEDA [24].

Fig. 7 illustrates high-level synthesis with three pure reagents R_1, R_2 and R_3, and two intermediate mixtures I_1 and I_2 with a volume of 16 units. In order to obtain three target mixture M_1, M_2 and M_3 with a volume of 32 units, we need to extract "aliquots" from pure reagents and intermediate mixtures, and then mixing them together. For example, in order to obtain M_1, we need to extract 6 units of M_1, 8 units of M_2, 6 units of M_3, 4 units of I_1, and 4 units of I_2 (see the number of the edges), and then combine them.

MEDA biochips discretize the $\mathcal{CF}$ space. We define parameter δ as the *degree of concentration accuracy*. The concentration factor of a reagent R_j in mixture M_i, which is defined as $cf_{(i,j)} = V(R_j)/V(M_i)$ in Section I, can be represented using an integer called *concentration factor integer* (cfi), which is defined as $cfi_{(i,j)} = \lceil cf_{(i,j)}/\frac{1}{\delta} \rceil$. For example, if $\delta = 128$, a cf of 0.64 can be represented using a cfi of $\lceil 0.64/\frac{1}{128} \rceil = 82$.

B. Problem Formulation for High-Level Synthesis

Based on the above discussion, we describe the optimization problem as follows:

Inputs: (1) The number of target mixtures, intermediate mixtures and reagents are tm, im and rg, respectively. (2) The degree of accuracy is δ. (3) The CF profiles of im intermediate mixtures $\{CF(I_1), CF(I_2), ..., CF(I_{im})\}$ and their volumes $\{V(I_1), V(I_2), ..., V(I_{im})\}$. (4) The CF profiles of tm target mixtures $\{CF(M_1), CF(M_2), ..., CF(M_{tm})\}$ and their volumes $\{V(M_1), V(M_2), ..., V(M_{tm})\}$.

Output: (1) Volumes of aliquots $B(i,k)$ and $A(i,j)$; (2) Actual CF profile of target mixtures $\{\hat{CF}(M_1), \hat{CF}(M_2), ..., \hat{CF}(M_{tm})\}$ and the actual volumes $\{\hat{V}(M_1), \hat{V}(M_2), ..., \hat{V}(M_{tm})\}$; (3) A mixture-production assay that can generate the target mixtures on MEDA.

Constraints: Aliquoting constraints of MEDA biochips.

Objective: Minimize reagent usage.

C. ILP-Based High-Level Synthesis

The above problem can be optimally solved by mapping it to an ILP model, as described below:

$$\forall i \in \{1, ..., tm\}; \forall j \in \{1, ..., rg\}; \forall k \in \{1, ..., im\} \tag{3}$$

Minimize:

$$\sum_{i=1}^{tm} \sum_{j=1}^{rg} \alpha(j) \times A(i,j) \tag{4}$$

where $\alpha(j)$ is the unit cost of reagent R_j.

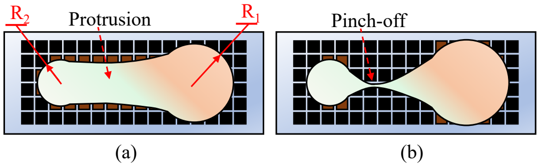
Subject to:

$$B_{min}(i,k) < B(i,k) \leq V(I_k); \ \sum_{k=1}^{im} B(i,k) \leq V(I_k) \tag{5}$$

$$A_{min}(i,j) < A(i,j) \tag{6}$$

$$\hat{V}(M_i) = \sum_{k}^{im} B(i,k) + \sum_{j}^{rg} A(i,j) \geq V(M_i) \tag{7}$$

$$\hat{cf}_{(i,j)} = \frac{\sum_{k=1}^{im} cf_{(i,k)} \cdot B(i,k) + A(i,j)}{\hat{V}(M_i)} \tag{8}$$



(a) (b)

Fig. 8: Steps in droplet aliquoting: (a) finger formation; (b) pinch-off.

$$\left| \hat{cf}_{(i,j)} \cdot \delta - cf_{(i,j)} \cdot \delta \right| < 0.5 \tag{9}$$

Equation (4) shows that we need to minimize the overall reagent cost in the ILP formulation. Equation (5) and Equation (6) specify the volume constraints of the aliquot droplet from an intermediate mixture I_k or a reagent fluid from the reservoir. Equation (7) calculates the actual volume of target mixture M_i, which should be equal to or larger than the designated value $V(M_i)$. Equation (8) calculates the actual concentration factor (CF) of a reagent R_j in mixture M_i, which should be within the calibrated range of the designated value of $cf_{(i,j)}$; see Equation (9).

V. PHYSICAL-LEVEL SYNTHESIS METHOD

The objective of physical-level synthesis is to map the high-level synthesis solution (see Section IV) to a sequence of MEDA-enabled operations with the lowest completion time and the smallest reagent cost. In addition, the constraints associated with droplet aliquoting must be satisfied.

A. Droplet-Aliquoting Constraints

In a droplet-aliquot operation on MEDA, a smaller target droplet, with principal radius of curvature R_2, is extracted from another droplet, with principal radius of curvature R_1 ($R_1 > R_2$). This operation is a key enabler of fine-grained mixture production. However, the main difficulty with droplet aliquoting, similar to droplet dispensing [25], is the control of the flow rate that leads to aliquot formation. Note that the original droplet, which is used before the aliquoting operation, has principal radius of curvature R_s that can be computed in terms of R_1 and R_2 as follows: $R_s \approx \sqrt{(R_1^2 + R_2^2)}$.

As shown in Fig. 8, droplet aliquoting is performed in two steps: (1) finger formation, which initiates aliquoting by inducing a protrusion from the bigger droplet; (2) pinch-off, which breaks the protrusion to form an aliquot. According to [25], a protrusion can be successfully maintained only if the following condition is fulfilled:

$$0 < \frac{1}{R_2} - \frac{1}{R_1} < \frac{\epsilon_0 \cdot \epsilon_r \cdot (V_e - V_e^{th})^2}{2\gamma_{LM} \cdot d_e \cdot s_e} \tag{10}$$

where ϵ_0 and ϵ_r are the permittivity of free space and the relative permittivity of the insulator, respectively, V_e and V_e^{th} are the actuation voltage and the threshold voltage, respectively, γ_{LM} is the liquid-medium interfacial tension, d_e is the insulator thickness, and s_e is the spacing between the parallel plates. For

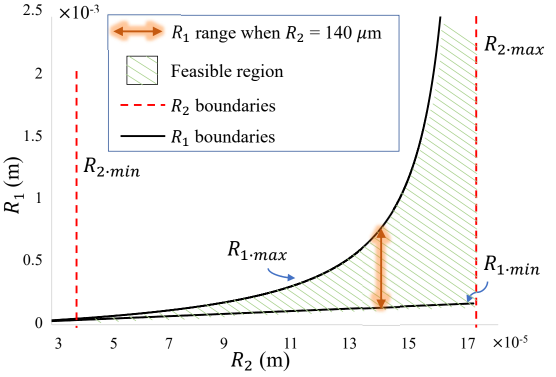


Fig. 9: Illustration of aliquoting constraints ($\varrho = 5761\ m^{-1}$; $s_e = 50\ \mu m^{-1}$).

simplicity, if we consider $\varrho = \frac{\epsilon_0 \cdot \epsilon_r \cdot (\mathcal{V}_e - \mathcal{V}_e^{th})^2}{2\gamma_{LM} \cdot d_e \cdot s_e}$, then according to Equation (10) we have:

$$R_2 = R_{1.min} < R_1 < R_{1.max} = \frac{R_2}{1 - \varrho \cdot R_2} \tag{11}$$

Note that $R_{1.max}$ must be positive in Equation (11), and the smallest droplet that can be moved on MEDA covers at least a microelectrode. Therefore, we have the following constraints:

$$\frac{L_e}{\sqrt{2}} = R_{2.min} \leq R_2 < R_{2.max} = \frac{1}{\varrho} \tag{12}$$

where L_e is the microelectrode pitch.

Fig. 9 shows a graphical representation of the relation between R_1 and R_2 based on the above constraints[1]. For example, a target droplet with principal radius of curvature $R_2 = 140\ \mu$m (i.e., of volume 11.5 nL) can be aliquoted from a droplet with principal radius of curvature R_1 only if 140 μm $< R_1 <$ 724 μm. If $R_1 \geq$ 724 μm, then the volume of the bigger droplet needs to be reduced until R_1 lies in the range (140, 724). The sequence of operations involved in this process is specified using physical-level synthesis as described in the following subsection.

B. Problem Formulation for Physical-Level Synthesis

In fact, the proposed physical-level synthesis consists of a sequence of MEDA-enabled operations (mixing, splitting, and droplet aliquoting), which can be modeled as a directed acyclic graph $G = (V, E)$, named a *composition graph*. A vertex $v_i \in V$ represents a MEDA-enabled operation, which can be one of four types: (1) mixing; (2) splitting; (3) aliquoting; (4) null operation (i.e., start/end point). Each operation type is associated with a cost value: $cost$(aliquoting) $> cost$(mixing) $> cost$(splitting) $> cost$(null) $= 0$. The mixing and splitting are easier to implement and take less time compared to aliquoting [10], [26]. In addition, an edge $e_{(i,j)} \in E$ models the dependency between a pair of operations v_i and v_j.

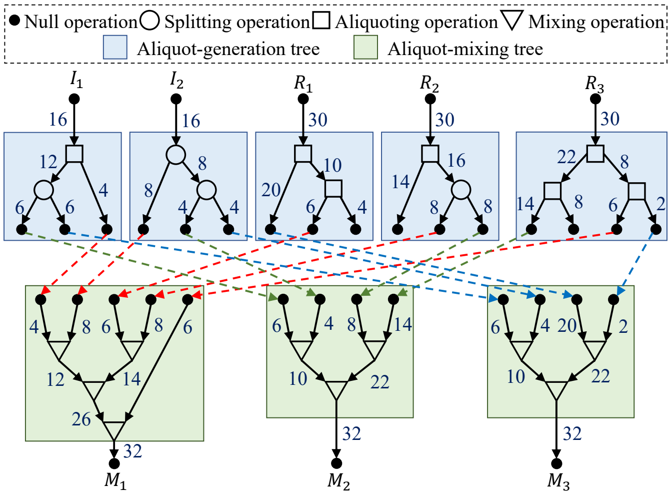


Fig. 10: A composition graph that generates three target mixtures from two intermediate mixtures, and three reagent fluids.

Fig. 10 shows the physical-level synthesis results derived from the high-level synthesis solution in Fig. 7. Note this is only one of many composition graphs that can be used to implement the solution in Fig. 7. For example, in Fig. 10 instead of generating the aliquots from I_2 using two splitting operations, an alternative (but more costly) solution may involve two aliquoting operations to generate aliquots with volumes of 8, 4 and 4, respectively. Note that a unit volume is defined as the volume of a droplet that covers one MC.

Based on the above discussion, we describe the physical-level synthesis problem as follows:

Inputs: (1) The volume $\hat{V}(M_i)$ of every target mixture M_i; (2) The volume $V(I_k)$ of every intermediate mixture I_k; (3) The volume $V(R_j)$ of reagent R_j being used; (4) Volumes of aliquots $A(i,j)$ and $B(i,k)$ from the high-level synthesis.
Output: A composition graph G.
Constraints: Droplet aliquoting constraints for MEDA biochips.
Objectives: Minimize the completion time of the physical-level synthesis.

Table I summarizes the notation used to describe the physical-level synthesis problem.

An optimal composition graph is one that has the lowest overall cost computed in Equation (13) and the lowest completion time, and it can be found by enumerating all possible composition graphs and exhaustively searching for the optimal graph. Clearly, this is an impractical solution that entails significant computation time. Therefore, we propose an alternative design methodology that divides the problem of computing a composition graph to two groups of subproblems: (1) computing $im + rg$ aliquot-generation binary trees, and (2) computing tm aliquot-mixing binary trees, where im, rg and tm are the numbers of intermediate mixtures, reagents and target mixtures, respectively.

---

[1] The intuition behind the above constraints is that aliquoting requires an electrowetting force, specified by ϱ, that is sufficient to overcome the pressure gradient between the two droplets. A large gap in the pressure between the two droplets (due to the variation in radii of curvature) may prevent protrusion formation [25].

TABLE I: Notation used to describe physical-level synthesis.

Notation	Meaning
$\mathcal{T}^g_{[o_1 \ldots o_N]}$	An aliquot-generation tree with N leafs
$C(\mathcal{T}^g_{[o_1 \ldots o_N]})$	Overall cost of $\mathcal{T}^g_{[o_1 \ldots o_N]}$
$\mathcal{T}^g_{([o_1 \ldots o_N],j)}$	A node j in $\mathcal{T}^g_{[o_1 \ldots o_N]}$
$O(\mathcal{T}^g_{([o_1 \ldots o_N],j)})$	Operation type of $\mathcal{T}^g_{([o_1 \ldots o_N],j)}$
$C(\mathcal{T}^g_{([o_1 \ldots o_N],j)})$	Cost of $\mathcal{T}^g_{([o_1 \ldots o_N],j)}$
$S(\mathcal{T}^g_{([o_1 \ldots o_N],j)})$	Start time of $\mathcal{T}^g_{([o_1 \ldots o_N],j)}$
$F(\mathcal{T}^g_{([o_1 \ldots o_N],j)})$	Finish time of $\mathcal{T}^g_{([o_1 \ldots o_N],j)}$
$H(\mathcal{T}^g_{([o_1 \ldots o_N],j)})$	Height of $\mathcal{T}^g_{([o_1 \ldots o_N],j)}$
$\mathcal{T}^m_{[o_1 \ldots o_N]}$	An aliquot-mixing tree with N leafs
$\mathcal{T}^m_{([o_1 \ldots o_N],j)}$	A node j in $\mathcal{T}^m_{[o_1 \ldots o_N]}$
$S(\mathcal{T}^m_{([o_1 \ldots o_N],j)})$	Start time of $\mathcal{T}^m_{([o_1 \ldots o_N],j)}$
$F(\mathcal{T}^m_{([o_1 \ldots o_N],j)})$	Finish time of $\mathcal{T}^m_{([o_1 \ldots o_N],j)}$

C. *Construction of Aliquot-Generation Binary Tree*

A tree $\mathcal{T}^g_i$ represents a hierarchy of MEDA-enabled operations needed to generate aliquots from an intermediate mixture I_k or from a reagent R_j; see Fig. 10. The root of the tree represents the first MEDA-enabled operation applied to I_k or R_j and the leaf nodes represent the generated aliquots. We focus our discussion on the construction of $\mathcal{T}^g_{[o_1 \ldots o_N]}$ that is associated with I_k—a tree $\mathcal{T}^g_{[o_1 \ldots o_N]}$ associated with R_j can be constructed similarly.

We consider a mapping function, denoted by f_{sort}, that is implemented by sorting the aliquot volumes $B(i,k)$ before assigning them to the leaf nodes, i.e., $f_{sort} : \{B(i,k)\} \rightarrow [o_1, o_2, ..., o_N]$, where $[o_1, o_2, ..., o_N]$ is a sorted list of aliquots derived from intermediate mixture I_k.

The completion time of a protocol is determined by the height of the binary tree. If we have a balanced tree, we will have shorter completion time. Therefore, modeling aliquots generation as a binary tree can let us map a scheduling problem to a problem of reducing the height of the tree. An objective function that captures the overall cost of the binary tree $\mathcal{T}^g_{[o_1 \ldots o_N]}$ can be computed as:

$$C(\mathcal{T}^g_{[o_1 \ldots o_N]}) = \sum_{j=1}^{M} C(\mathcal{T}^g_{([o_1 \ldots o_N],j)}) \cdot H(\mathcal{T}^g_{([o_1 \ldots o_N],j)}) \qquad (13)$$

where M is the number of nodes in $\mathcal{T}^g_{[o_1 \ldots o_N]}$.

Another relevant advantage of using binary trees is that binary search trees can be constructed optimally using recursive methods such as dynamic programming [27]. Similarly, we observe that the construction of $\mathcal{T}^g_{[o_1 \ldots o_N]}$ can be recursively decomposed into subproblems; each subproblem can be solved optimally, and the obtained solution can be reused to solve the original problem.

An illustrative example is shown in Fig. 11. Consider an intermediate mixture with a volume of 16 nL that needs to be used to generate three aliquots A, B, and C, with volumes of 8 nL, 4 nL, and 4 nL, respectively. To construct the tree $\mathcal{T}^g_{[A\ B\ C]}$ shown in Fig. 11(a), which is the tree with the lowest cost, we first construct two subtrees ($\mathcal{T}^g_{[A\ B]}$ and $\mathcal{T}^g_{[B\ C]}$) that represent two-aliquot problems (Step 1 in Fig. 11(b)). In $\mathcal{T}^g_{[A\ B]}$, the merging of A and B is modeled as an aliquoting operation, and the cost is therefore equal to 2. On the other hand, in $\mathcal{T}^g_{[B\ C]}$, the merging of B and C is modeled as a splitting operation, and the corresponding cost is therefore equal to 1.

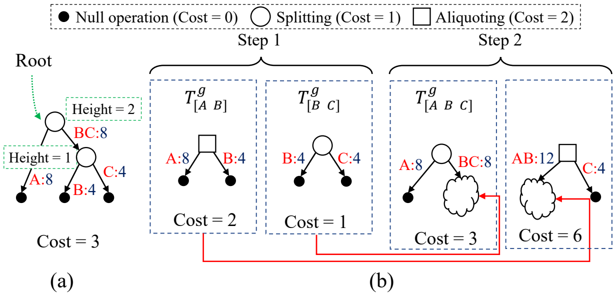


Fig. 11: Construction of a tree $\mathcal{T}^g_{[A\ B\ C]}$ that models the generation of 3 aliquots: (a) the minimal-cost tree; (b) solution methodology.

Next, these solutions are reused to compute the trees for a three-aliquot problem (Step 2 in Fig. 11(b)), and the solution/tree with the minimal cost is selected for $\mathcal{T}^g_{[A\ B\ C]}$. Dynamic programming is successful (i.e., optimal) if both “optimal substructure” and “overlapping subproblems” properties are satisfied. In our setting, the preprocessing of the leaf nodes by sorting them (using f_{sort}) ensures that these requirements are satisfied. Unsorted leaf nodes may lead to suboptimal solutions being propagated throughout the construction of the tree.

The above example can be generalized to the following result:

Lemma 1. *An aliquot-generation tree $\mathcal{T}^g_{[o_1 \ldots o_N]}$ has the “overlapping-subproblems” and the “optimal-substructure” properties.*

Based on the above lemma, the construction of $\mathcal{T}^g_{[o_1 \ldots o_N]}$ can be mapped to a dynamic programming problem [27], and the the mapping is defined as follows (the proof of Lemma 1 and Theorem 1 can be found in the appendix).

Theorem 1. *An optimal aliquot-generation tree $\mathcal{T}^g_{([o_1 \ldots o_N])}$ with root $\mathcal{T}^g_{([o_1 \ldots o_N],1)}$ can be constructed using dynamic programming, where the recursion can be described as follows:*

$$C(T^g_{[o_1 \ldots o_N]}) = \begin{cases} \underset{1 \le k < N}{\arg\min}\{C(T^g_{[o_1 \ldots o_k]}) + C(T^g_{[o_{k+1} \ldots o_N]})\} & \\ \quad +C(T^g_{([o_1 \ldots o_N],1)}) \cdot H(T^g_{([o_1 \ldots o_N],1)}) & N \ge 1 \\ \qquad 0 & N < 1 \end{cases}$$

A description of our dynamic programming implementation is shown in Fig. 12. If list $[o_1 \ldots o_N]$ exists in the lookup table, the aliquot tree will be returned immediately (Lines 1-2). If the length of list $[o_1 \ldots o_N]$ is equal to one (i.e., only one node), we will store the result into the lookup table (Lines 3-5). Otherwise, we iterate the value of k from 1 to $N-1$, and find out the value of kt that minimizes $C(\mathcal{T}^g_{[o_1 \ldots o_N]})$ (Lines 6-14). Note that in Line 6, we use “9999” to indicate a value that is very large. Next, we merge two subtrees $C(\mathcal{T}^g_{[o_1 \ldots o_{kt}]})$ and $C(\mathcal{T}^g_{[o_{kt+1} \ldots o_N]})$ to construct $C(\mathcal{T}^g_{[o_1 \ldots o_N]})$, and store the result into the lookup table (Lines 15-16). Note that, this merge is performed by

```
Algorithm 1 create_aliquot_tree([$o_1...o_N$], LT)
Input: a list of nodes [$o_1...o_N$] and the lookup table $LT$
Output: the aliquot-generation tree $T^g_{[o_1...o_N]}$ and the cost $C(T^g_{[o_1...o_N]})$
if has_record([$o_1...o_N$], LT) then
    return LT[$o_1...o_N$];
if len([$o_1...o_N$]) = 1 then
    LT[$o_1...o_N$] := ([$o_1...o_N$], 0);
    return LT[$o_1...o_N$];
$C_{min}$ := 9999;
$kt$ := 0;
for $o_k$ in [$o_1...o_N$] do
    $T^g_{[o_1...o_k]}$, $C(T^g_{[o_1...o_k]})$ := create_aliquot_tree([$o_1...o_k$], LT);
    $T^g_{[o_{k+1}...o_N]}$, $C(T^g_{[o_{k+1}...o_N]})$ := create_aliquot_tree([$o_{k+1}...o_N$], LT);
    $C(T^g_{[o_1...o_N]}) := C(T^g_{([o_1...o_N],1)}) \cdot H(T^g_{([o_1...o_N],1)})$;
    $C(T^g_{[o_1...o_N]}) := C(T^g_{[o_1...o_N]}) + C(T^g_{[o_1...o_k]}) + C(T^g_{[o_{k+1}...o_N]})$;
    if $C(T^g_{[o_1...o_N]}) < C_{min}$ then
        $kt := k$;
$T^g_{[o_1...o_N]}$ := merge($T^g_{([o_1...o_N],1)}, T^g_{[o_1...o_{kt}]}, T^g_{[o_{kt+1}...o_N]}$);
LT[$o_1...o_N$] := ($T^g_{[o_1...o_N]}$, $C(T^g_{[o_1...o_N]})$);
return LT[$o_1...o_N$];
```

Fig. 12: Pseudocode describing the construction of $\mathcal{T}^g_{[o_1...o_N]}$.

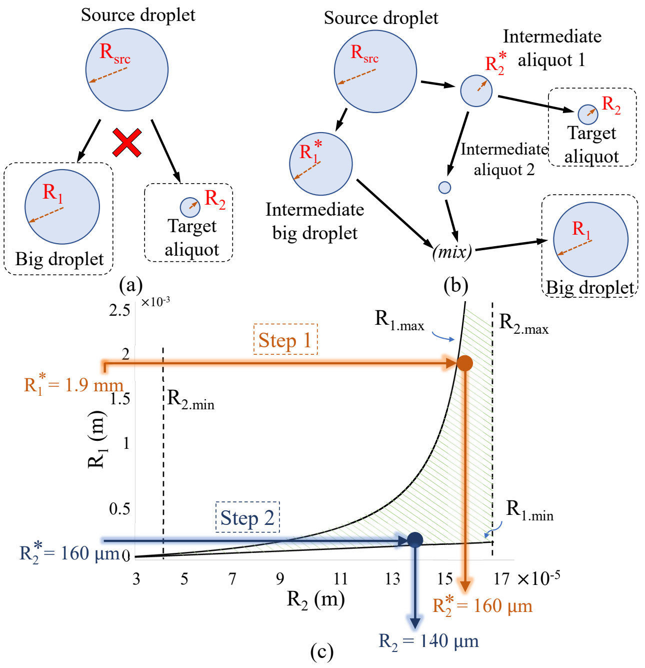


Fig. 13: MEDA-enabled aliquoting: (a) aliquoting is not possible when the constraints are not satisfied; (b) an alternative sequence of MEDA operations for aliquoting (i.e., *fragment* step defined in Section V.C.); (c) illustration of fragment steps using the aliquoting-constraints curve.

selecting a suitable MEDA-enabled operation. If an aliquoting operation needs to be implemented but the aliquoting constraints cannot be satisfied, then a single aliquoting operation is not adequate to generate the target volume.

To explain the steps needed to overcome the above challenge, suppose that an aliquot with a radius of curvature $R_2 = 140$ μm is needed, and the source droplet has a radius of curvature $R_{src} = 2.004$ mm. Based on the relation $R_{src} \approx \sqrt{(R_1^2 + R_2^2)}$, the radius of curvature of the bigger droplet is $R_1 = 2$ mm; see Fig. 13(a). However, based on Fig. 13, the aliquoting constraints for $R_2 = 140$ μm are $R_1 > 140$ μm and $R_1 < 724$ μm; thus aliquoting based on the above values of R_1 and R_2 is not possible[2].

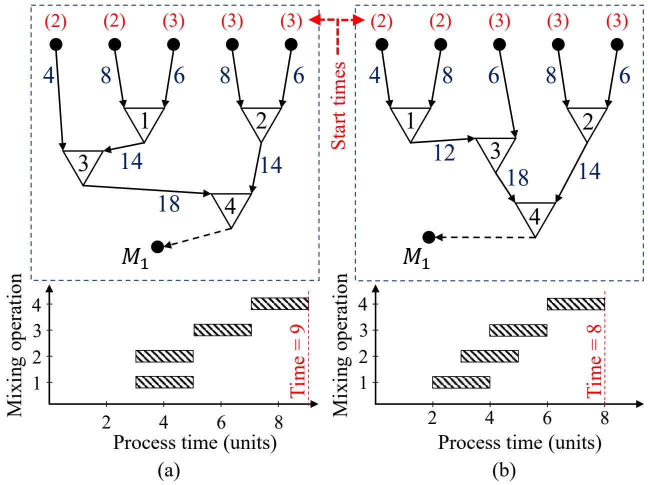


Fig. 14: Computing an aliquot-mixing tree and the corresponding completion time using: (a) a naive method; (b) a priority-based method. Each mixing operation takes two time steps.

To overcome this problem, we choose a different value for R_2 in the interval $(140, R_{2.max})$ so that aliquoting can be performed. This implies that the target aliquot may be obtained after executing two aliquoting operations. Fig. 13(b)-(c) explains the steps needed to address this challenge. For convenience, we consider the steps in Fig. 13(b) as a single operation $\mathcal{T}^g_{([o_1...o_N],j)}$, named **fragment**. The cost value $C(\mathcal{T}^g_{([o_1...o_N],j)})$ of this operation is determined based on the constituent sequence of aliquoting operations.

Post tree-construction, we use a one-pass algorithm to compute the start time $S(\mathcal{T}^g_{([o_1...o_N],j)})$ and finish time $F(\mathcal{T}^g_{([o_1...o_N],j)})$ of each node $\mathcal{T}^g_{([o_1...o_N],j)}$ [28].

D. *Construction of Aliquot-Mixing Binary Tree*

An aliquot-mixing tree $\mathcal{T}^m_{[o_1...o_N]}$ represents a hierarchy of mixing operations that merge aliquots to form a target mixture M_i; see Fig. 10. The primitive aliquots are modeled as leaf nodes, whereas the last mixing operation before the target mixture is modeled as a root node. An intermediate tree node corresponds to a MEDA-enabled mixing operation that merges two intermediate aliquots. Note that, in reality, the mixing operations of different types and sizes of droplets have different time cost. Our physical-level synthesis model can be easily adapted to support mixing operations with different time costs. However, for simplicity, we assume that all mixing operations have the same time cost.

Recall that the primitive aliquots are generated by the aliquot-generation trees. Therefore their finish times, which indicate their availability, may not be equal. Such a variation in the availability of the input primitive aliquots causes the generation of the aliquot-mixing trees $\mathcal{T}^m_{[o_1...o_N]}$ to be computationally challenging. In other words, using a naive method that determines

---

[2] Note that R_{src} is fixed.

```
Algorithm 2 create_mixing_tree($[o_1...o_N]$)
Input: a list of nodes $[o_1...o_N]$
Output: the mixing-generation tree $T^m_{[o_1...o_N]}$ and its finish time $F(T^m_{[o_1...o_N]})$
 1: $T^m_{[o_1...o_N]}$ := null;
 2: $PQ$ := create_priority_queue($[o_1...o_N]$);
 3: while len($PQ$) != 0 do
 4:     if len($PQ$) = 1 then
 5:         $o_j$, $ft_j$ := $PQ$.pop_front();
 6:         return $o_j$, $ft_j$
 7:     else
 8:         $o_j$, $ft_j$ := $PQ$.pop_front();
 9:         $o_k$, $ft_k$ := $PQ$.pop_front();
10:         $o_l$ := create_binary_mixing_tree($o_j$, $o_k$);
11:         $ft_l$ := max($ft_j$, $ft_k$) + $t_{mix}$;
12:         $PQ$ := update_priority_queue($PQ$, $o_l$, $ft_l$);
```

Fig. 15: Procedure describing the constructing of $\mathcal{T}^m_{[o_1...o_N]}$.

the mixing of aliquots randomly, without considering their availability, may unnecessarily increase the completion time, whereas adopting a priority scheme can enhance the synthesis performance. Fig. 14 shows an example that compares the completion time based on the two approaches.

Note that if $\mathcal{T}^m_{([o_1...o_N],j)}$ is a leaf, then $S(\mathcal{T}^m_{([o_1...o_N],j)}) = F(\mathcal{T}^m_{([o_1...o_N],j)})$. Moreover, if a leaf node $\mathcal{T}^m_{([o_1...o_N],j)}$ is directly connected to a leaf node $\mathcal{T}^g_{([o_1...o_N],k)}$ (see Fig. 10), then the start time of $\mathcal{T}^m_{([o_1...o_N],j)}$ is defined as $S(\mathcal{T}^m_{([o_1...o_N],j)}) = F(\mathcal{T}^g_{([o_1...o_N],k)})$. Our goal in this stage is to minimize completion time, which can be computed as:

$$\max_j F(\mathcal{T}^m_{([o_1...o_N],j)}) \ \forall \mathcal{T}^m_{([o_1...o_N],j)} \tag{14}$$

To construct $\mathcal{T}^m_{([o_1...o_N],j)}$, we develop a greedy method based on a priority queue PQ that stores MEDA-enabled operations and sort them based on their finish time (Fig. 15); more specifically, the queue uses a *nearest-finish-time-first* approach. Initially, the primitive aliquots are sorted in ascending order in the priority queue PQ according to their finish times (Line 2). Then, the algorithm runs as a time-wheel. At each time, it will retrieve two elements o_j and o_k from the priority queue PQ. Each element can represent either a primitive aliquot or a mixing tree. The finish time of these two elements are ft_j and ft_k, respectively (Lines 8-9). Then, we construct a new element o_l by merging o_j and o_k, and compute the finish time ft_l (Lines 10-11). Next, we add this element into PQ and update the order (Line 12). This algorithm stops when the queue becomes empty, meaning that the target mixing tree $\mathcal{T}^m_{([o_1...o_N],j)}$ is successfully formed (Line 3).

VI. SIMULATION RESULTS

In this section, we evaluate the sampling method of $\mathcal{CF}$ space and the proposed sample-preparation method.

A. Analysis of CF Sampling

Recall that the sampling of $\mathcal{CF}$ space is accomplished in two steps: regular sampling within the interval $[0, 1]$ followed by mapping of samples to the simplex CF space. Therefore, we evaluate the space filling of the CF profiles based on these two steps; we compare four sampling approaches: (1) OA-LHS (stratified sampling) followed by Dirichlet-based mapping where $\widetilde{\alpha} = 3$; (2) OA-LHS followed by Dirichlet-based mapping where $\widetilde{\alpha} = 20$; (3) OA-LHS followed by scaling-based mapping; (4) uniform sampling followed by scaling-based mapping. The number of CFs, i.e., reagents rg, is set to 8, and the number of sampling trials in each case is 1000. Results based on other values of rg also lead to the same conclusion, showing that our methodology is scalable with rg.

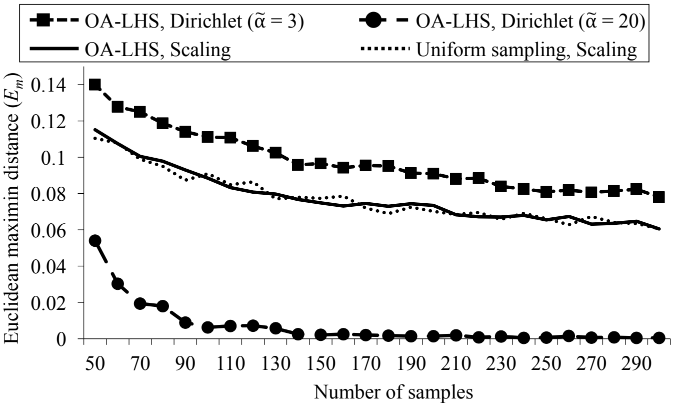


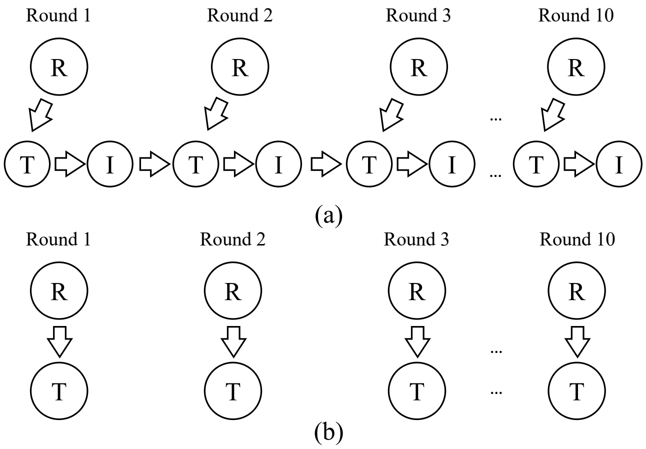
Fig. 16: The performance of (1) OA-LHS using Dirichlet-based mapping; (2) OA-LHS using scaling-based mapping; (3) uniform sampling techniques on space filling.

Fig. 16 compares the above sampling approaches using E_m as a metric when we vary the number of targeted samples tm. We observe that scaling-based mapping degrades the space-filling property, i.e., reduces E_m, regardless of which sampling method is utilized. This is expected since scaling-based mapping tampers with the stratification property. We also observe that space filling can be severely degraded if the Dirichlet-based mapping is not properly tuned. We have varied $\widetilde{\alpha}$ from 0 to 20, and found that large values of $\widetilde{\alpha}$ cause the generated samples to be highly concentrated instead of being uniformly distributed, i.e., as shown in Fig. 16, $\widetilde{\alpha} = 20$ leads to a low value of E_m. We also found that $\widetilde{\alpha} = 3$ give us the highest value of E_m. Therefore, in the following evaluations, we use OA-LHS with Dirichlet-based mapping where $\widetilde{\alpha} = 3$.

B. Simulation Setup

To evaluate the effectiveness of the proposed sample-preparation method, we compare it with three existing sample-preparation methods, i.e., BS [29], CoDOS [30], and e-MRCM [31]. While BS and CoDOS methods were developed for traditional DMFBs, e-MRCM method is designed for MEDA biochips. To compare these methods properly, we consider that the volumes of the target droplet are the same across a DMFB and a MEDA biochip in our experiments. We assume that the gaps between the top plate and the bottom plate are the same, and the electrode size of the DMFB is 1×1 mm^2 [32]. We also assume that the minimum aliquot droplet spans an area of 0.25×0.25 mm^2 [10] on a MEDA biochip, i.e., the target droplet is 16 times as big as the aliquot droplet.

Experiments with different numbers of reactants N_r ($3 \leq N_r \leq 5$) and target mixtures N_t ($2 \leq N_t \leq 5$) are carried out for all the methods. We adopt a degree of accuracy $\delta = 512$



(a)

(b)

Fig. 17: Sample-preparation flow for (a) the proposed method and (b) prior methods. “R” represents pure reagent, “T” represents target mixture, and “I” represents intermediate mixture.

(see in Section IV.A) because precision levels for traditional DMFBs can only be powers of 2. A total of 10 rounds of sample preparations are simulated. In each round, a statistical sampling method is used to choose N_t CF points in the N_r dimension CF space. Next, a total of N_t target mixtures are generated. As shown in Fig. 17(b), in each round, prior methods only use the reagent to generate target mixtures. However, except for the first round, the proposed method uses both the intermediate mixtures from the last round and the reagent to generate the target mixtures; see Fig. 17(a).

Two metrics are used to evaluate the effectiveness of the proposed method, namely average reagent usage and average completion time.

The *average reagent usage* $\bar{RU}$ is defined as follows:

$$\bar{RU} = \frac{\sum_{i=1}^{10} RU(i)}{10} \tag{15}$$

where $RU(i)$ is the reagent usage of round i. Note that we define the volume of a droplet that occupies an electrode of 1×1 mm^2 as the unit reagent usage.

The *average completion time* $\bar{CT}$ is defined as follows:

$$\bar{CT} = \frac{\sum_{i=1}^{10} CT(i)}{10} \tag{16}$$

where $CT(i)$ is the completion time of round i. Since module placement and droplet routing are not considered in our simulation, the completion time of each round is estimated as the shortest time needed to execute the fluidic operation steps in the dilution tree(s).

All simulations are performed in Python on a workstation with a 3.6 GHz octa-core AMD processor and 32 GB memory.

C. Experimental Results

In this section, we consider sample preparation scenarios where the number of reactants N_r varies from 3 to 5. For each choice of N_r, we conducted experiments by varying the number of target mixtures N_t from 2 to 5. The average reagent usages and average completion times for all methods are shown in Table II and Table III, respectively. Note that the unit of time is in seconds.

We can see that e-MRCM performs better than BS and CoDoS in terms of the reagent usage, because e-MRCM is based on the MEDA platform and the $M : N$ mixing model is utilized, which enables a fine-grained mixing strategy. However, the proposed method achieves at least 20% reduction in reagent usage compared with e-MRCM, because e-MRCM only uses the reagents to generate target mixtures while the proposed method uses both the reagent and the intermediate mixtures to generate target mixtures.

Considering the average completion time, the proposed method uses a smaller amount of time to finish each round. The significant time reduction is because the aliquot-generation trees and the aliquot-mixing trees in Fig. 10 can be performed in parallel. However, the construction of the dilution trees in BS, CoDOS and e-MRCM are more “serial”, and therefore they use more time to finish each round.

VII. CONCLUSION

We have introduced an optimization framework for parameter-space exploration in synthetic biology. The proposed framework uses statistical sampling to select reagent mixtures, a high-level synthesis method to provide specification for the biocircuit-regulatory scanning assays. We have also presented a technique that translates the high-level synthesis solutions to a sequence of fluidic operations for implementing them on MEDA-chips with reduced completion time and reagent cost. Simulation results have shown the effectiveness of the proposed method in emulating important problems of synthetic biology.

REFERENCES

[1] W. Weber and M. Fussenegger, “Emerging biomedical applications of synthetic biology,” *Nature Reviews Genetics*, vol. 13, no. 1, p. 21, 2012.

[2] Y. Hori, C. Kantak, R. M. Murray, and A. R. Abate, “Cell-free extract based optimization of biomolecular circuits with droplet microfluidics,” *Lab on a Chip*, vol. 17, no. 18, pp. 3037–3042, 2017.

[3] T. S. Moon, C. Lou, A. Tamsir, B. C. Stanton, and C. A. Voigt, “Genetic programs constructed from layered logic gates in single cells,” *Nature*, vol. 491, no. 7423, pp. 249–253, 2012.

[4] L. Potvin-Trottier, N. D. Lord, G. Vinnicombe, and J. Paulsson, “Synchronous long-term oscillations in a synthetic gene circuit,” *Nature*, vol. 538, no. 7626, pp. 514–517, 2016.

[5] J. W. Kotula, S. J. Kerns, L. A. Shaket, L. Siraj, J. J. Collins, J. C. Way, and P. A. Silver, “Programmable bacteria detect and record an environmental signal in the mammalian gut,” *Proceedings of the National Academy of Sciences*, vol. 111, no. 13, pp. 4838–4843, 2014.

[6] L. You, R. S. Cox, R. Weiss, and F. H. Arnold, “Programmed population control by cell–cell communication and regulated killing,” *Nature*, vol. 428, no. 6985, pp. 868–871, 2004.

[7] M. Ibrahim, B. B. Bhattacharya, and K. Chakrabarty, “Bioscan: Parameter-space exploration of synthetic biocircuits using meda biochips*,” in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2019, pp. 1519–1524.

[8] D. J. Beebe, R. J. Adrian, M. G. Olsen, M. A. Stremler, H. Aref, and B.-H. Jo, “Passive mixing in microchannels: Fabrication and flow experiments,” *Mécanique & Industries*, vol. 2, no. 4, pp. 343–348, 2001.

[9] V. Hessel, H. Löwe, and F. Schönfeld, “Micromixers—a review on passive and active mixing principles,” *Chemical Engineering Science*, vol. 60, no. 8-9, pp. 2479–2501, 2005.

[10] Z. Li, K. Y.-T. Lai, K. Chakrabarty, T.-Y. Ho, and C.-Y. Lee, “Droplet size-aware and error-correcting sample preparation using micro-electrode-dot-array digital microfluidic biochips,” *IEEE Transactions on Biomedical Circuits and Systems (TBioCAS)*, 2017.

TABLE II: Average Reagent Usage for Different Sample-Preparation Methods (unit: 0.25×0.25 mm^2 droplet volume).

Method	$N_r = 3$				$N_r = 4$				$N_r = 5$			
	$N_t = 2$	$N_t = 3$	$N_t = 4$	$N_t = 5$	$N_t = 2$	$N_t = 3$	$N_t = 4$	$N_t = 5$	$N_t = 2$	$N_t = 3$	$N_t = 4$	$N_t = 5$
BS	56.7	88.2	114.9	144.3	84.7	128.5	172.5	200.5	90.9	140.8	180.6	236.4
CoDOS	48.0	72.3	94.2	118.1	60.1	90.6	122.3	154.2	66.1	100.4	134.1	164.1
e-MRCM	24.7	37.1	49.4	61.8	29.4	44.1	58.8	73.5	33.2	49.6	66.7	84.3
Proposed	19.3	25.0	32.9	43.6	28.0	38.2	49.4	57.5	33.0	48.2	62.5	76.5

TABLE III: Average Completion Time for Different Sample-Preparation Methods (unit: second).

Method	$N_r = 3$				$N_r = 4$				$N_r = 5$			
	$N_t = 2$	$N_t = 3$	$N_t = 4$	$N_t = 5$	$N_t = 2$	$N_t = 3$	$N_t = 4$	$N_t = 5$	$N_t = 2$	$N_t = 3$	$N_t = 4$	$N_t = 5$
BS	16.0	16.0	16.0	16.0	26.1	26.1	26.1	26.1	33.8	33.8	33.8	33.8
CoDOS	13.8	13.8	13.8	13.8	21.2	21.2	21.2	21.2	23.8	23.8	23.8	23.8
e-MRCM	9.1	9.9	10.5	11.6	11.3	12.7	13.9	14.6	13.3	13.9	14.8	15.8
Proposed	7.6	8.8	9.6	10.5	8.0	9.8	10.4	11.1	9.7	10.4	10.8	11.7

[11] Z. Li, K. Y.-T. Lai, P.-H. Yu, K. Chakrabarty, M. Pajic, T.-Y. Ho, and C.-Y. Lee, "Error recovery in a micro-electrode-dot-array digital microfluidic biochip?" in *International Conference on Computer-Aided Design (ICCAD)*, 2016, pp. 1–8.

[12] S. Roy, B. B. Bhattacharya, and K. Chakrabarty, "Optimization of dilution and mixing of biochemical samples using digital microfluidic biochips," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 29, no. 11, pp. 1696–1708, 2010.

[13] T. A. Dinh, S. Yamashita, and T.-Y. Ho, "A network-flow-based optimal sample preparation algorithm for digital microfluidic biochips," in *Design Automation Conference (ASP-DAC), 2014 19th Asia and South Pacific*, 2014, pp. 225–230.

[14] S. Poddar, S. Ghoshal, K. Chakrabarty, and B. B. Bhattacharya, "Error-correcting sample preparation with cyberphysical digital microfluidic lab-on-chip," *ACM Transactions on Design Automation of Electronic Systems (TODAES*, vol. 22, no. 1, pp. 2:1–29, 2016.

[15] D. Mitra, S. Roy, S. Bhattacharjee, K. Chakrabarty, and B. B. Bhattacharya, "On-chip sample preparation for multiple targets using digital microfluidics," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 33, no. 8, pp. 1131–1144, 2014.

[16] S. Poddar, S. Bhattarcharjee, S. C. Nandy, K. Chakrabarty, and B. B. Bhattacharya, "Optimization of multi-target sample preparation on-demand with digital microfluidic biochips," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2018.

[17] Y.-L. Hsieh, T.-Y. Ho, and K. Chakrabarty, "Biochip synthesis and dynamic error recovery for sample preparation using digital microfluidics," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 33, no. 2, pp. 183–196, 2014.

[18] 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 Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 20, no. 3, p. 45, 2015.

[19] S. Bhattacharjee, Y.-L. Chen, J.-D. Huang, and B. B. Bhattacharya, "Concentration-resilient mixture preparation with digital microfluidic lab-on-chip," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 17, no. 2, p. 49, 2018.

[20] H. Niederreiter, *Random number generation and quasi-Monte Carlo methods*. SIAM, 1992, vol. 63.

[21] T. M. Cioppa and T. W. Lucas, "Efficient nearly orthogonal and space-filling Latin hypercubes," *Technometrics*, vol. 49, no. 1, pp. 45–55, 2007.

[22] B. Tang, "Orthogonal array-based Latin hypercubes," *Journal of the American statistical association*, vol. 88, no. 424, pp. 1392–1397, 1993.

[23] P. Schneider and D. H. Eberly, *Geometric tools for computer graphics*. Elsevier, 2002.

[24] Z. Zhong, Z. Li, and K. Chakrabarty, "Adaptive error recovery in MEDA biochips based on droplet-aliquot operations and predictive analysis," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2017, pp. 615–622.

[25] H. Ren, R. B. Fair, and M. G. Pollack, "Automated on-chip droplet dispensing with volume control by electro-wetting actuation and capacitance metering," *Sensors and Actuators B: Chemical*, vol. 98, no. 2, pp. 319–327, 2004.

[26] G. Wang, "Field-programmable microfluidic test platform for point-of-care diagnostics," Ph.D. dissertation, University of Saskatoon, Saskatchewan, Canada, 2013.

[27] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. MIT Press Cambridge, 2001.

[28] R. Wille, O. Keszocze, R. Drechsler, T. Boehnisch, and A. Kroker, "Scalable one-pass synthesis for digital microfluidic biochips," *IEEE Design & Test*, vol. 32, no. 6, pp. 41–50, 2015.

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

[30] C.-H. Liu, H.-H. Chang, T.-C. Liang, and J.-D. Huang, "Sample preparation for many-reactant bioassay on dmfbs using common dilution operation sharing," in *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2013, pp. 615–621.

[31] T.-C. Liang, Y.-S. Chan, T.-Y. Ho, K. Chakrabarty, and C.-Y. Lee, "Multi-target sample preparation using meda biochips," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2019.

[32] H. Moon and J. Nikapitiya, "Digital microfluidic devices and methods of dispensing and splitting droplets in digital microfluidic devices," Jun. 23 2016, uS Patent App. 14/976,019.

Proof of Lemma 1

We provide a proof for **Lemma 1** that is stated in Section V of the paper. First we restate this lemma:

Lemma 1. *An aliquot-generation tree* $\mathcal{T}^g_{[o_1 \ldots o_N]}$ *has the "overlapping-subproblems" and the "optimal-substructure" properties.*

Table I summarizes the notation used in this proof. Consider an aliquot-generation tree $\mathcal{T}^g_{[o_1 \ldots o_N]}$, which contains N *leaf* nodes (aliquots), as shown in Fig. 1. The tree $\mathcal{T}^g_{[o_1 \ldots o_N]}$ can be decomposed into a root node $\mathcal{T}^g_{([o_1 \ldots o_N],1)}$, a left subtree $\mathcal{T}^g_{([o_1 \ldots o_k])}$ and a right subtree $\mathcal{T}^g_{([o_{k+1} \ldots o_N])}$. Suppose the left subtree has a total of a nodes and the right subtree has a total of b nodes. The illlustration of the node IDs for $\mathcal{T}^g_{([o_1 \ldots o_N],1)}$, $\mathcal{T}^g_{[o_1 \ldots o_k]}$ and $\mathcal{T}^g_{[o_{k+1} \ldots o_N]}$ are shown in Fig. 1.

We define the overall cost of $\mathcal{T}^g_{[o_1 \ldots o_N]}$ as follows:

$$C(\mathcal{T}^g_{[o_1 \ldots o_N]}) = \sum_{j=1}^{M} C(\mathcal{T}^g_{([o_1 \ldots o_N],j)}) \cdot H(\mathcal{T}^g_{([o_1 \ldots o_N],j)}) \tag{1}$$

where $M = a + b + 1$ is the number of nodes in $\mathcal{T}^g_{[o_1 \ldots o_N]}$.

I. OVERLAPPING-SUBPROBLEMS PROPERTY

The overall cost of the left subtree $T^g_{[o_1 \ldots o_k]}$ (bounded by a red-colored triangle in Fig. 1), can be computed as follows:

$$\begin{aligned} C(\mathcal{T}^g_{[o_1 \ldots o_k]}) &= C(\mathcal{T}^g_{([o_1 \ldots o_k],1)}) \cdot H(\mathcal{T}^g_{([o_1 \ldots o_k],1)}) \\ &+ \ldots + C(\mathcal{T}^g_{([o_1 \ldots o_k],a)}) \cdot H(\mathcal{T}^g_{([o_1 \ldots o_k],a)}) \end{aligned} \tag{2}$$

Since $\mathcal{T}^g_{[o_1 \ldots o_k]}$ is a subtree of $\mathcal{T}^g_{[o_1 \ldots o_N]}$, then the tree nodes $\mathcal{T}^g_{([o_1 \ldots o_k],j)}$ also belong to the tree $\mathcal{T}^g_{[o_1 \ldots o_N]}$ (Fig. 1), therefore Equation (2) can be rewritten as follows:

$$\begin{aligned} C(\mathcal{T}^g_{[o_1 \ldots o_k]}) &= C(\mathcal{T}^g_{([o_1 \ldots o_N],2)}) \cdot H(\mathcal{T}^g_{([o_1 \ldots o_N],2)}) \\ &+ \ldots + C(\mathcal{T}^g_{([o_1 \ldots o_N],a+1)}) \cdot H(\mathcal{T}^g_{([o_1 \ldots o_N],a+1)}) \end{aligned} \tag{3}$$

TABLE I: Notation used in the physical-level synthesis.

Notation	Meaning
$\mathcal{T}^g_{[o_1 \ldots o_N]}$	An aliquot-generation tree with N leafs
$C(\mathcal{T}^g_{[o_1 \ldots o_N]})$	Overall cost of $\mathcal{T}^g_{[o_1 \ldots o_N]}$
$\mathcal{T}^g_{([o_1 \ldots o_N],j)}$	A node j in $\mathcal{T}^g_{[o_1 \ldots o_N]}$
$O(\mathcal{T}^g_{([o_1 \ldots o_N],j)})$	Operation type of $\mathcal{T}^g_{([o_1 \ldots o_N],j)}$
$C(\mathcal{T}^g_{([o_1 \ldots o_N],j)})$	Cost of $\mathcal{T}^g_{([o_1 \ldots o_N],j)}$
$S(\mathcal{T}^g_{([o_1 \ldots o_N],j)})$	Start time of $\mathcal{T}^g_{([o_1 \ldots o_N],j)}$
$F(\mathcal{T}^g_{([o_1 \ldots o_N],j)})$	Finish time of $\mathcal{T}^g_{([o_1 \ldots o_N],j)}$
$H(\mathcal{T}^g_{([o_1 \ldots o_N],j)})$	Height of $\mathcal{T}^g_{([o_1 \ldots o_N],j)}$

Similarly, the overall cost of the right subtree $T^g_{[o_{k+1} \ldots o_N]}$ (bounded by a green-colored triangle in Fig. 1), can be computed as follows:

$$\begin{aligned} C(\mathcal{T}^g_{[o_{k+1} \ldots o_N]}) &= c(\mathcal{T}^g_{([o_{k+1} \ldots o_N],1)}) \cdot H(\mathcal{T}^g_{([o_{k+1} \ldots o_N],1)}) \\ &+ \ldots + C(\mathcal{T}^g_{([o_{k+1} \ldots o_N],b)}) \cdot H(\mathcal{T}^g_{([o_{k+1} \ldots o_N],b)}) \end{aligned} \tag{4}$$

Since $\mathcal{T}^g_{[o_{k+1} \ldots o_N]}$ is a subtree of $\mathcal{T}^g_{[o_1 \ldots o_N]}$, then the tree nodes $\mathcal{T}^g_{([o_{k+1} \ldots o_N],j)}$ also belong to the tree $\mathcal{T}^g_{[o_1 \ldots o_N]}$ (Fig. 1), therefore Equation (4) can be rewritten as follows:

$$\begin{aligned} C(\mathcal{T}^g_{[o_{k+1} \ldots o_N]}) &= C(\mathcal{T}^g_{([o_1 \ldots o_N],a+2)}) \cdot H(\mathcal{T}^g_{([o_1 \ldots o_N],a+2)}) \\ &+ \ldots + C(\mathcal{T}^g_{([o_1 \ldots o_N],a+1+b)}) \cdot H(\mathcal{T}^g_{([o_1 \ldots o_N],a+1+b)}) \end{aligned} \tag{5}$$

Next, we compute the overall cost of the tree $\mathcal{T}^g_{[o_1 \ldots o_N]}$ (bounded by a blue-colored triangle in Fig. 1) as follows:

$$\begin{aligned} C(\mathcal{T}^g_{[o_1 \ldots o_N]}) &= C(\mathcal{T}^g_{([o_1 \ldots o_N],1)}) \cdot H(\mathcal{T}^g_{([o_1 \ldots o_N],1)}) \\ &+ C(\mathcal{T}^g_{([o_1 \ldots o_N],2)}) \cdot H(\mathcal{T}^g_{([o_1 \ldots o_N],2)}) \\ &+ \ldots + C(\mathcal{T}^g_{([o_1 \ldots o_N],a+1)}) \cdot H(\mathcal{T}^g_{([o_1 \ldots o_N],a+1)}) \\ &+ C(\mathcal{T}^g_{([o_1 \ldots o_N],a+2)}) \cdot H(\mathcal{T}^g_{([o_1 \ldots o_N],a+2)}) \\ &+ \ldots + C(\mathcal{T}^g_{([o_1 \ldots o_N],a+1+b)}) \cdot H(\mathcal{T}^g_{([o_1 \ldots o_N],a+1+b)}) \end{aligned} \tag{6}$$

By substituting Equation (3) and Equation (5) into Equation (6), we obtain the following result:

$$\begin{aligned} C(\mathcal{T}^g_{[o_1 \ldots o_N]}) &= C(\mathcal{T}^g_{([o_1 \ldots o_N],1)}) \cdot H(\mathcal{T}^g_{([o_1 \ldots o_N],1)}) \\ &+ C(\mathcal{T}^g_{[o_1 \ldots o_k]}) + C(\mathcal{T}^g_{[o_{k+1} \ldots o_N]}) \end{aligned} \tag{7}$$

Hence, we observe that $C(\mathcal{T}^g_{[o_1 \ldots o_N]})$ can be computed in terms of $C(\mathcal{T}^g_{[o_1 \ldots o_k]})$ and $C(\mathcal{T}^g_{[o_{k+1} \ldots o_N]})$. Therefore, we obtain the *overlapping-subproblems* property as follows:

Property 1. *An aliquot-generation tree* $\mathcal{T}^g_{[o_1 \ldots o_N]}$ *with a root* $\mathcal{T}^g_{([o_1 \ldots o_N],1)}$ *exhibits overlapping subproblems, and the overall cost can be computed as follows:*

$$\begin{aligned} C(\mathcal{T}^g_{[o_1 \ldots o_N]}) &= C(\mathcal{T}^g_{([o_1 \ldots o_N],1)}) \cdot H(\mathcal{T}^g_{([o_1 \ldots o_N],1)}) \\ &+ C(\mathcal{T}^g_{[o_1 \ldots o_k]}) + C(\mathcal{T}^g_{[o_{k+1} \ldots o_N]}) \end{aligned} \tag{8}$$

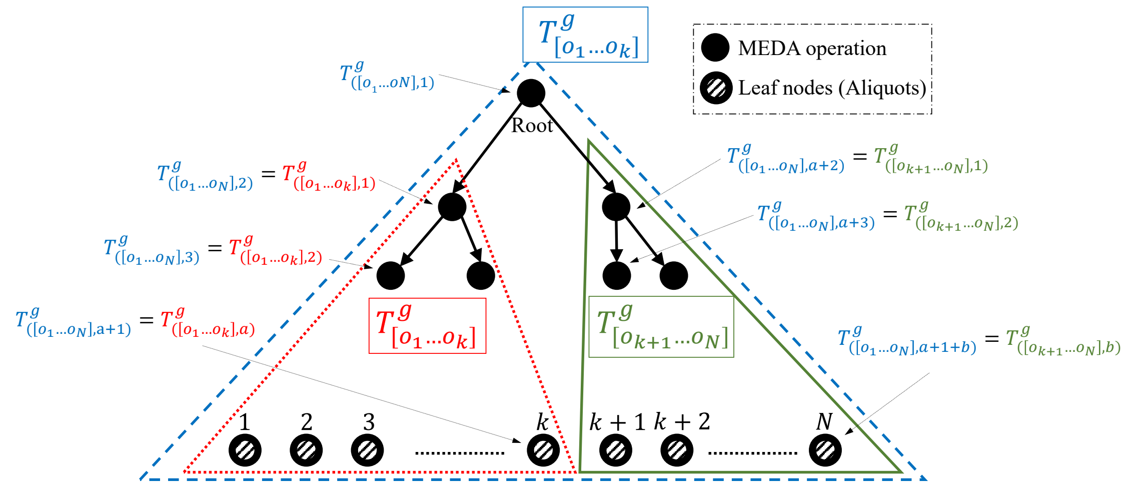

$T^g_{[o_1 \ldots o_k]}$
MEDA operation
Leaf nodes (Aliquots)
$T^g_{([o_1 \ldots o_N],1)}$
Root
$T^g_{([o_1 \ldots o_N],2)} = T^g_{([o_1 \ldots o_k],1)}$
$T^g_{([o_1 \ldots o_N],3)} = T^g_{([o_1 \ldots o_k],2)}$
$T^g_{([o_1 \ldots o_N],a+1)} = T^g_{([o_1 \ldots o_k],a)}$
$T^g_{[o_1 \ldots o_k]}$
$T^g_{[o_{k+1} \ldots o_N]}$
$T^g_{([o_1 \ldots o_N],a+2)} = T^g_{([o_{k+1} \ldots o_N],1)}$
$T^g_{([o_1 \ldots o_N],a+3)} = T^g_{([o_{k+1} \ldots o_N],2)}$
$T^g_{([o_1 \ldots o_N],a+1+b)} = T^g_{([o_{k+1} \ldots o_N],b)}$
1
2
3
k
$k+1$
$k+2$
N

Fig. 1: An aliquot-generation tree $\mathcal{T}^g_{[o_1 \ldots o_N]}$ and its subtrees $\mathcal{T}^g_{[o_1 \ldots o_k]}$ and $\mathcal{T}^g_{[o_{k+1} \ldots o_N]}$.

where $C(\mathcal{T}^g_{[o_1 \ldots o_k]})$ *and* $C(\mathcal{T}^g_{[o_{k+1} \ldots o_N]})$ *are the overall cost of the left and right subtrees, and* $1 \leq k < N$.

II. OPTIMAL-SUBSTRUCTURE PROPERTY

Suppose that the tree $\mathcal{T}^g_{[o_1 \ldots o_N]}$ is associated with an optimal cost $C(\mathcal{T}^g_{[o_1 \ldots o_N]})$, i.e., the minimum cost. Based on Property 1, we aim to examine whether the values of the cost functions $C(\mathcal{T}^g_{[o_1 \ldots o_k]})$ and $C(\mathcal{T}^g_{[o_{k+1} \ldots o_N]})$ are optimal when $C(\mathcal{T}^g_{[o_1 \ldots o_N]})$ is optimal. If $C(\mathcal{T}^g_{[o_1 \ldots o_k]})$ and $C(\mathcal{T}^g_{[o_{k+1} \ldots o_N]})$ are proven to be optimal, then the tree $\mathcal{T}^g_{[o_1 \ldots o_N]}$ exhibits the optimal-substructure property.

We present a proof by contradiction using the generic form in Fig. 1. Suppose that the tree $\mathcal{T}^g_{[o_1 \ldots o_N]}$ is *optimal* since it is associated with an optimal cost $C(\mathcal{T}^g_{[o_1 \ldots o_N]})$. Also, the tree $\mathcal{T}^g_{[o_1 \ldots o_N]}$ consists of two subtrees $\mathcal{T}^g_{[o_1 \ldots o_k]}$ and $\mathcal{T}^g_{[o_{k+1} \ldots o_N]}$. Hence, our proof is presented considering two cases.

Case 1: Suppose for contradiction that the left subtree $\mathcal{T}^g_{[o_1 \ldots o_k]}$ is not optimal. This assumption means that there is another subtree structure, denoted by $\widetilde{\mathcal{T}}^g_{[o_1 \ldots o_k]}$, that has a lower cost value $C(\widetilde{\mathcal{T}}^g_{[o_1 \ldots o_k]})$, where $C(\widetilde{\mathcal{T}}^g_{[o_1 \ldots o_k]}) < C(\mathcal{T}^g_{[o_1 \ldots o_k]})$.

Based on this assumption, if we replacing the subtree $\mathcal{T}^g_{[o_1 \ldots o_k]}$ with the optimal subtree $\widetilde{\mathcal{T}}^g_{[o_1 \ldots o_k]}$. Then, the overall cost of the new tree $\widetilde{\mathcal{T}}^g_{[o_1 \ldots o_N]}$ can be computed as follows:

$$\begin{aligned} C(\widetilde{\mathcal{T}}^g_{[o_1 \ldots o_N]}) &= C(\mathcal{T}^g_{([o_1 \ldots o_N],1)}) \cdot H(\mathcal{T}^g_{([o_1 \ldots o_N],1)}) \\ &\quad + C(\widetilde{\mathcal{T}}^g_{[o_1 \ldots o_k]}) + C(\mathcal{T}^g_{[o_{k+1} \ldots o_N]}) \end{aligned} \tag{9}$$

By comparing Equation (7) and Equation (9), we observe that the assumption $(C(\widetilde{\mathcal{T}}^g_{[o_1 \ldots o_k]}) < C(\mathcal{T}^g_{[o_1 \ldots o_k]}))$ implies that $C(\widetilde{\mathcal{T}}^g_{[o_1 \ldots o_N]}) < C(\mathcal{T}^g_{[o_1 \ldots o_N]})$. This result contradicts with the assumption that $\mathcal{T}^g_{[o_1 \ldots o_N]}$ is optimal (i.e., minimized). As a result, if $\mathcal{T}^g_{[o_1 \ldots o_N]}$ is optimal, then the subtree $\mathcal{T}^g_{[o_1 \ldots o_k]}$ must also be optimal.

Case 2: A similar proof to Case 1 can be derived, while considering the right subtree $\mathcal{T}^g_{[o_{k+1} \ldots o_N]}$, to show that the subtree $\mathcal{T}^g_{[o_{k+1} \ldots o_N]}$ must be optimal if $\mathcal{T}^g_{[o_1 \ldots o_N]}$ is optimal.

Based on the presented proof, we obtain the *optimal-substructure* property as follows:

Property 2. *Consider an aliquot-generation tree* $\mathcal{T}^g_{[o_1 \ldots o_N]}$ *that contains two subtrees* $\mathcal{T}^g_{[o_1 \ldots o_k]}$ *and* $\mathcal{T}^g_{[o_{k+1} \ldots o_N]}$. *The tree* $\mathcal{T}^g_{[o_1 \ldots o_N]}$ *exhibits optimal substructure, therefore* $\mathcal{T}^g_{[o_1 \ldots o_N]}$ *is optimal if both subtrees* $\mathcal{T}^g_{[o_1 \ldots o_k]}$ *and* $\mathcal{T}^g_{[o_{k+1} \ldots o_N]}$ *are optimal.*

Proof of Theorem 1

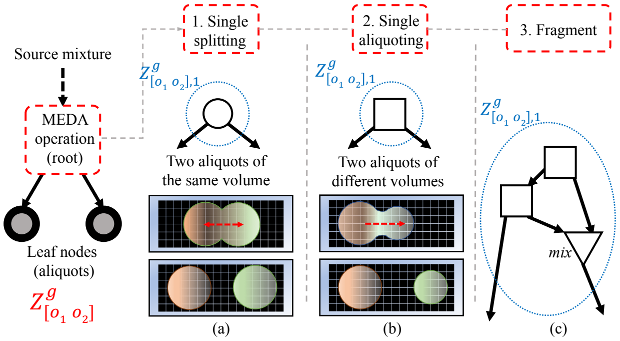


Fig. 1: Possible MEDA operations to generate $\mathcal{T}^g_{[o_1\ o_2]}$: (a) a single splitting operation; (b) a single aliquoting operation; (c) a fragment operation.

We present the proof of Theorem 1 in the appendix. First we restate the theorem:

Theorem 1. *An optimal aliquot-generation tree* $\mathcal{T}^g_{([o_1\ldots o_N])}$ *with root* $\mathcal{T}^g_{([o_1\ldots o_N],0)}$ *can be constructed using dynamic programming, where the recursion can be described as follows:*

$$C(T^g_{[o_1\ldots o_N]}) = \begin{cases} \underset{1\le k<N}{\arg\min}\{C(T^g_{[o_1\ldots o_k]}) + C(T^g_{[o_{k+1}\ldots o_N]})\} & \\ \quad +C(T^g_{([o_1\ldots o_N],1)})\cdot H(T^g_{([o_1\ldots o_N],1)}) & N\ge 1 \\ \qquad 0 & N<1 \end{cases}$$

Our objective here is to show that, by using the properties in Lemma 1 (i.e., overlapping subproblems and optimal substructure), an *optimal* aliquot-generation tree $\mathcal{T}^g_{[o_1\ldots o_N]}$ can be constructed using dynamic programming.

The Base Case: We start with the smallest scale of an aliquot-generation tree, which contains only two leaf nodes, i.e., $\mathcal{T}^g_{[o_1\ o_2]}$, as shown in Fig. 1. At this scale, a single MEDA operation is needed to generate the aliquots, on condition that the aliquoting constraints (Section V.) can be satisfied. If the two aliquots have the same volume, then a splitting operation is performed Fig. 1(a). However, if the two aliquots have different volumes, then an aliquoting operation is executed Fig. 1(b).

On the other hand, if the aliquoting constraints cannot be satisfied directly, then two consecutive aliquoting operations are performed to generate the target aliquot volume; these steps are referred to as a *fragment* operation (Fig. 1(c))—the optimality of the fragment operation based on these steps is discussed as part of the presented proof. For convenience, we consider the steps of a fragment operation as a single operation $\mathcal{T}^g_{([o_1\ o_2],j)}$ and its associated cost is $C(\mathcal{T}^g_{([o_1\ o_2],j)}) > cost(\text{aliquot})$. Based on the above discussion, we introduce the following lemma:

Lemma 2. *An aliquot-generation tree* $\mathcal{T}^g_{[o_1\ o_2]}$ *with root* $\mathcal{T}^g_{([o_1\ o_2],1)}$ *and two leaf nodes* $\{\mathcal{T}^g_{([o_1\ o_2],2)}, \mathcal{T}^g_{([o_1\ o_2],3)}\}$ *is optimal, and the cost is computed as follows:*

$$\begin{aligned} C(\mathcal{T}^g_{[o_1\ o_2]}) = {} & C(\mathcal{T}^g_{([o_1\ o_2],1)})\cdot H(\mathcal{T}^g_{([o_1\ o_2],1)}) \\ & + \underset{1\le k<2}{\arg\min}\{C(T^g_{[o_1\ldots o_k]}) + C(T^g_{[o_{k+1}\ldots o_2]})\} \end{aligned} \tag{1}$$

Proof. We present the proof by contradiction. Suppose that the tree $\mathcal{T}^g_{[o_1\ o_2]}$ is not optimal. This implies that there is another tree $\widetilde{\mathcal{T}}^g_{[o_1\ o_2]}$ that has a lower cost, i.e., $C(\widetilde{\mathcal{T}}^g_{[o_1\ o_N]}) < C(\mathcal{T}^g_{[o_1\ o_N]})$. Since the leaf nodes are the same in both trees, then this implies that $C(\widetilde{\mathcal{T}}^g_{([o_1\ o_2],1)}) < C(\mathcal{T}^g_{([o_1\ o_2],1)})$. This inequality is studied based on three cases related to aliquots' volumes:

Case 1: Suppose that the two leaf nodes represent two aliquots that have the same volume, meaning that $C(\mathcal{T}^g_{([o_1\ o_2],1)}) = cost(\text{split})$. Since $cost(\text{split}) < cost(\text{aliquot}) < cost(\text{fragment})$, then $C(\mathcal{T}^g_{([o_1\ o_2],1)})$ exhibits the lowest value. This contradicts the original assumption that $C(\widetilde{\mathcal{T}}^g_{([o_1\ o_2],1)}) < C(\mathcal{T}^g_{([o_1\ o_2],1)})$. As a result, $\mathcal{T}^g_{[o_1\ o_2]}$ must be optimal if $\mathcal{T}^g_{([o_1\ o_2],1)}$ represents a splitting operation.

Since $\mathcal{T}^g_{[o_1\ o_2]}$ is optimal and using Lemma 1, the subtrees $\mathcal{T}^g_{[o_1}$ and $\mathcal{T}^g_{[o_2]}$ are also optimal. Note that $C(\mathcal{T}^g_{[o_1]}) = C(\mathcal{T}^g_{([o_1\ o_2],2)})$ and $C(\mathcal{T}^g_{[o_2]}) = C(\mathcal{T}^g_{([o_1\ o_2],3)})$. Hence, the cost of $\mathcal{T}^g_{[o_1\ o_2]}$ is computed as follows:

$$\begin{aligned} C(\mathcal{T}^g_{[o_1\ o_2]}) &= C(\mathcal{T}^g_{([o_1\ o_2],1)}) \cdot H(\mathcal{T}^g_{([o_1\ o_2],1)}) \\ &+ C(\mathcal{T}^g_{[o_1]}) + C(\mathcal{T}^g_{[o_2]}) \\ &= C(\mathcal{T}^g_{([o_1\ o_2],1)}) \cdot H(\mathcal{T}^g_{([o_1\ o_2],1)}) \\ &+ \arg\min_{1\le k<2}\{C(T^g_{[o_1\ldots o_k]}) + C(T^g_{[o_{k+1}\ldots o_2]})\} \end{aligned} \tag{2}$$

which is the required result.

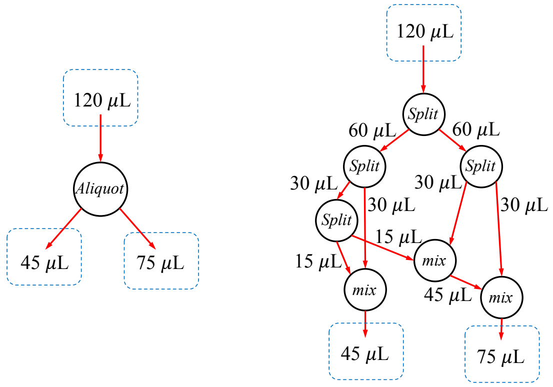
Case 2: Suppose that the two leaf nodes $\{\mathcal{T}^g_{([o_1\ o_2,2)}, \mathcal{T}^g_{([o_1\ o_2,3)}\}$ represent aliquots of different volumes, and that the aliquoting constraints are satisfied. This implies that $C(\mathcal{T}^g_{([o_1\ o_2],1)}) = cost(\text{aliquot})$. Hence, the original assumption that $\widetilde{ct}^g_{([o_1\ o_2],1)} < ct^g_{([o_1\ o_2],1)}$ cannot be true unless $C(\widetilde{\mathcal{T}}^g_{([o_1\ o_2],1)}) = cost(\text{split})$. However, since the aliquots' volumes are different, then these aliquots cannot be generated using a single splitting operation. The only possible method to generate two aliquots of different volumes using splitting is to perform serial splitting followed by mixing steps to attain the right volumes[1]. Fig. 2 compares the aliquoting and the serial splitting mechanisms. The best case scenario for the latter mechanism is achieved when only three splitting and two mixing operations are performed. Clearly, this still exhibits a higher cost $C(\widetilde{\mathcal{T}}^g_{([o_1\ o_2],1)})$ compared to the aliquoting cost $C(\mathcal{T}^g_{([o_1\ o_2],1)})$; this contradicts the original assumption that $C(\widetilde{\mathcal{T}}^g_{([o_1\ o_2],1)}) < C(\mathcal{T}^g_{([o_1\ o_2],1)})$. As a result, $\mathcal{T}^g_{[o_1\ o_2]}$ must be optimal if $\mathcal{T}^g_{([o_1\ o_2],1)}$ represents an aliquoting operation.

Also, similar to Case 1, the cost $C(\mathcal{T}^g_{[o_1\ o_2]})$ can now be computed as follows:

$$\begin{aligned} C(\mathcal{T}^g_{[o_1\ o_2]}) &= C(\mathcal{T}^g_{([o_1\ o_2],1)}) \cdot H(\mathcal{T}^g_{([o_1\ o_2],1)}) \\ &+ C(\mathcal{T}^g_{[o_1]}) + C(\mathcal{T}^g_{[o_2]}) \\ &= C(\mathcal{T}^g_{([o_1\ o_2],1)}) \cdot H(\mathcal{T}^g_{([o_1\ o_2],1)}) \\ &+ \arg\min_{1\le k<2}\{C(T^g_{[o_1\ldots o_k]}) + C(T^g_{[o_{k+1}\ldots o_2]})\} \end{aligned} \tag{3}$$

which is the required result.

Case 3: Suppose that the two leaf nodes $\{\mathcal{T}^g_{([o_1\ o_2],2)}, \mathcal{T}^g_{([o_1\ o_2],3)}\}$ represent aliquots of different volumes, and that the aliquoting constraints are not satisfied based on these volumes. This implies that $C(\mathcal{T}^g_{([o_1\ o_2],1)}) = cost(\text{fragment})$. Hence, the original assumption that $C(\widetilde{\mathcal{T}}^g_{([o_1\ o_2],1)}) < C(\mathcal{T}^g_{([o_1\ o_2],1)})$ cannot be true unless $C(\widetilde{\mathcal{T}}^g_{([o_1\ o_2],1)}) = cost(\text{split})$ or $C(\widetilde{\mathcal{T}}^g_{([o_1\ o_2],1)}) = cost(\text{aliquot})$.



(a) (b)

Fig. 2: Generation of two aliquots with different volumes using: (a) a single MEDA-enabled aliquoting operation; (b) serial splitting and mixing operations.

Case 3.A: We first examine the proposition that $C(\widetilde{\mathcal{T}}^g_{([o_1\ o_2],1)}) = cost(\text{split})$. Since the aliquots' volumes are different, then these aliquots cannot be generated using a single splitting operation. Hence, this proposition is not valid.

Case 3.B: We now examine the proposition that $C(\widetilde{\mathcal{T}}^g_{([o_1\ o_2],1)}) = cost(\text{aliquot})$. Since the aliquoting conditions are not satisfied, then these droplets cannot also be generated using a single aliquoting operation. Hence, this proposition is also not valid.

As a result, using a single primitive operation to generate the target aliquot is not possible[2]; we need to implement more than one primitive operation to obtain the required volume. Below we present a proof for the optimal sequence of primitive operations that can be used to implement the steps of a fragment operation.

Suppose that the source droplet, e.g., the 120 μL droplet in Fig. 2(a), has a principal radius of curvature W_{src}, the bigger droplet, e.g., the 75 μL droplet in Fig. 2(a), has a principal radius of curvature W_1, and the target aliquot, e.g., the 45 μL droplet in Fig. 2(a), has a principal radius of curvature W_2.

---

[1] This is equivalent to solving the sample-preparation problem using conventional DMFBs.

[2] A primitive operation is either splitting or aliquoting. We use the keyword "primitive" to distinguish between these operations and the fragment operation.

There are only three possible implementations of the fragment operation to obtain the target aliquot (Fig. 3(a-c)):

Case 3.C.I (Splitting and Mixing): We carry out consecutive splitting operations on W_{src} and its successors until we obtain multiple instances of the target droplet W_2; see Fig. 3(a). This approach is similar to Case 2, which proves that using splitting only is more costly compared with using aliquoting operations. This implies that this implementation does not lead to the optimal cost for fragment.

Case 3.C.II (Splitting, Aliquoting, and Mixing): We carry out consecutive splitting operations on W_{src} and its successors until we obtain multiple instances of droplets with a radius of $\frac{1}{k}W_{src}$, which can be used to generate a droplet with radius W_2 through droplet aliquot; see Fig. 3(b). Similar to Case 2, this implementation is costly and it may not lead to the optimal cost for fragment. Note that this proof is also applicable to the case where an aliquoting operation is used first followed by consecutive splitting operations. More specifically, the aliquoting operation generates an intermediate aliquot with radius $W_2^* > W_2$, which satisfies the aliquoting constraints. Next, this aliquot is processed through consecutive splitting operations.

Case 3.C.III (Aliquoting and Mixing): We carry out droplet-aliquot operation to generate an intermediate aliquot with a radius of W_2^*, where $W_2 < W_2^* < W_{2.max}$. This droplet will then be used to generate the target droplet with a radius of W_2; see Fig. 3(c). Note that only two droplet-aliquot operations are executed, and this assumption is based on the following facts: (1) an intermediate aliquot can always be generated when W_2^* lies in the interval $(W_2, W_{2.max})$; (2) the difference between W_2^* and W_2 is significantly small compared to the scale of the bigger droplet W_1, therefore the aliquoting constraints are always satisfied for the second aliquoting operation.

Based on the above discussion, the approach described in Case 3.C.III, which is guaranteed to use only two aliquoting steps, is the optimal method to perform the fragment operation. The cost $C(\mathcal{T}^g_{[o_1\ o_2]})$ can now be computed as follows:

$$\begin{aligned} C(\mathcal{T}^g_{[o_1\ o_2]}) &= C(\mathcal{T}^g_{([o_1\ o_2],1)}) \cdot H(\mathcal{T}^g_{([o_1\ o_2],1)}) \\ &\quad + C(\mathcal{T}^g_{[o_1]}) + C(\mathcal{T}^g_{[o_2]}) \\ &= C(\mathcal{T}^g_{([o_1\ o_2],1)}) \cdot H(\mathcal{T}^g_{([o_1\ o_2],1)}) \\ &\quad + \operatorname*{arg\,min}_{1 \le k < 2}\{C(T^g_{[o_1 \ldots o_k]}) + C(T^g_{[o_{k+1} \ldots o_2]})\} \end{aligned} \tag{4}$$

where $\mathcal{T}^g_{([o_1\ o_2],1)}$ is the fragment operation. □

Inductive Step: Next, we study a generic structure of $\mathcal{T}^g_{[o_1\ o_N]}$. Suppose that $\mathcal{T}^g_{[o_1\ o_N]}$ is an optimal aliquot-generation tree. Then, by splitting the tree into two subtrees, $\mathcal{T}^g_{[o_1\ o_k]}$ and $\mathcal{T}^g_{[o_{k+1}\ o_N]}$, we obtain the following result.

Lemma 3. *If an aliquot-generation tree $\mathcal{T}^g_{[o_1 \ldots o_N]}$ with root $\mathcal{T}^g_{([o_1 \ldots o_N],1)}$ is optimal, then:*

(1) its subtrees $\mathcal{T}^g_{[o_1 \ldots o_k]}$ and $\mathcal{T}^g_{[o_{k+1} \ldots o_N]}$ are also optimal, and

(2) the parameter k is specified as follows: $k = \arg\min_{1 \le k < N}\{C(\mathcal{T}^g_{[o_l \ldots o_k]}) + C(\mathcal{T}^g_{[o_{k+1} \ldots o_N]})\}$

(3) the cost $C(\mathcal{T}^g_{[o_1 \ldots o_N]})$ is computed as follows:

$$\begin{aligned} C(\mathcal{T}^g_{[o_1 \ldots o_N]}) &= C(\mathcal{T}^g_{([o_1 \ldots o_N],1)}) \cdot H(\mathcal{T}^g_{([o_1 \ldots o_N],1)}) \\ &\quad + \min_{1 \le k < N}\{C(\mathcal{T}^g_{[o_1 \ldots o_k]}) + C(\mathcal{T}^g_{[o_{k+1} \ldots o_N]})\} \end{aligned}$$

Proof. The proof for (1) is similar to the proof of Lemma 1. The proof for (2) and (3) is presented using construction as follows. Since both subtrees $\mathcal{T}^g_{[o_1 \ldots o_k]}$ and $\mathcal{T}^g_{[o_{k+1} \ldots o_N]}$ are optimal, then there is no other subtrees $\mathcal{T}^g_{[o_1 \ldots o_j]}$ and $\mathcal{T}^g_{[o_{j+1} \ldots o_N]}$ (where $j \neq k$) that can have lower cost values. In other words, the following conditions must be satisfied:

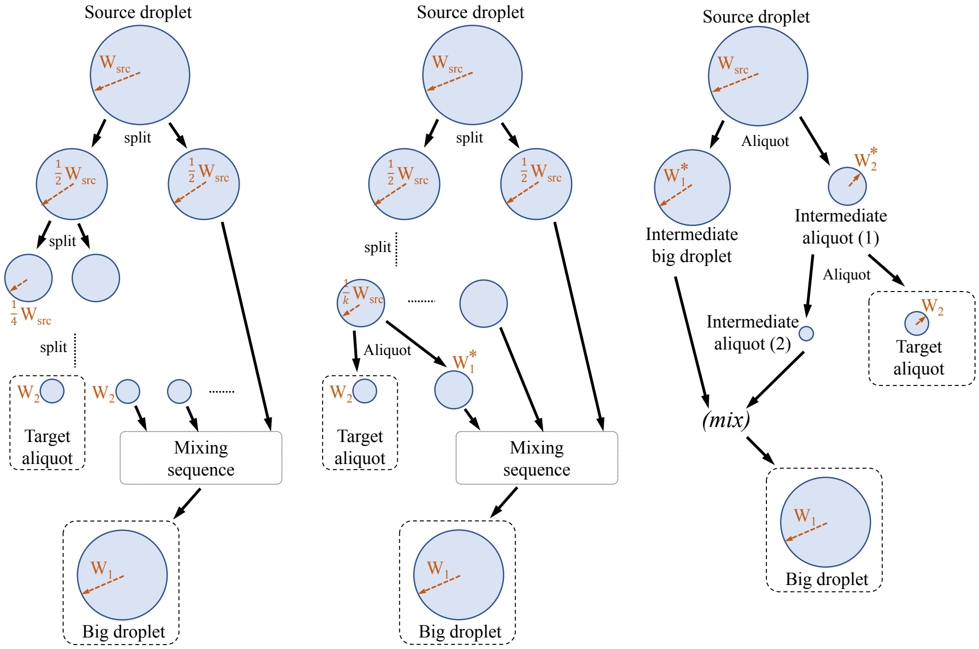
$$\begin{aligned} C(\mathcal{T}^g_{[o_1 \ldots o_k]}) + C(\mathcal{T}^g_{[o_{k+1} \ldots o_N]}) &> C(T^g_{[o_1]}) + C(\mathcal{T}^g_{[o_2 \ldots o_N]}) \\ C(\mathcal{T}^g_{[o_1 \ldots o_k]}) + C(\mathcal{T}^g_{[o_{k+1} \ldots o_N]}) &> C(T^g_{[o_1\ o_2]}) + C(\mathcal{T}^g_{[o_3 \ldots o_N]}) \\ C(\mathcal{T}^g_{[o_1 \ldots o_k]}) + C(\mathcal{T}^g_{[o_{k+1} \ldots o_N]}) &> C(T^g_{[o_1 \ldots o_j]}) + C(\mathcal{T}^g_{[o_{j+1} \ldots o_N]}) \\ C(\mathcal{T}^g_{[o_1 \ldots o_k]}) + C(\mathcal{T}^g_{[o_{k+1} \ldots o_N]}) &> C(T^g_{[o_1 \ldots o_{N-1}]}) + C(\mathcal{T}^g_{[o_N]}) \end{aligned} \tag{5}$$

where $j > 2$ and $j \neq k$.

By combining the above constraints, we obtain the required result for k as follows:

$$k = \operatorname*{arg\,min}_{1 \le k < N}\{C(\mathcal{T}^g_{[o_l \ldots o_k]}) + C(\mathcal{T}^g_{[o_{k+1} \ldots o_N]})\}$$

Based on the above result and by using Lemma 1, we obtain the required result for $CT^g_{[o_1 \ldots o_N]}$ as follows:



(a) (b) (c)

Fig. 3: Possible implementations of the fragment operation: (a) using splitting and mixing operations; (b) using splitting, aliquoting, and mixing operations; (c) using aliquoting and mixing operations.

$$\begin{aligned} C(\mathcal{T}^g_{[o_1\ldots o_N]}) &= C(\mathcal{T}^g_{([o_1\ldots o_N],1)}) \cdot H(\mathcal{T}^g_{([o_1\ldots o_N],1)}) \\ &+ \min_{1\le k<N}\{C(\mathcal{T}^g_{[o_1\ldots o_k]}) + C(\mathcal{T}^g_{[o_{k+1}\ldots o_N]})\} \end{aligned} \tag{6}$$

□

Using Lemma 2 and Lemma 3, and by induction, we are able to complete the proof of Theorem 1.