A Waste-Efficient Algorithm for Single-Droplet Sample Preparation on Microfluidic Chips

MIGUEL COVIELLO GONZALEZ and MAREK CHROBAK,

University of California at Riverside

We address the problem of designing micro-fluidic chips for sample preparation, which is a crucial step in many experimental processes in chemical and biological sciences. One of the objectives of sample preparation is to dilute the sample fluid, called reactant, using another fluid called buffer, to produce desired volumes of fluid with prespecified reactant concentrations. In the model we adopt, these fluids are manipulated in discrete volumes called droplets. The dilution process is represented by a *mixing graph* whose nodes represent 1-1 micro-mixers and edges represent channels for transporting fluids. In this work, we focus on designing such mixing graphs when the given sample (also referred to as the *target*) consists of a single-droplet, and the objective is to minimize total fluid waste. Our main contribution is an efficient algorithm called RPRIS that guarantees a better provable worst-case bound on waste and significantly outperforms state-of-the-art algorithms in experimental comparison.

CCS Concepts: • **Applied computing** → **Computer-aided design**;

Additional Key Words and Phrases: Algorithms, graph theory, lab-on-chip, fluid mixing

ACM Reference format:

Miguel Coviello Gonzalez and Marek Chrobak. 2020. A Waste-Efficient Algorithm for Single-Droplet Sample Preparation on Microfluidic Chips. *J. Exp. Algorithmics* 25, 1, Article 1.10 (September 2020), 22 pages. <https://doi.org/10.1145/3408297>

1 INTRODUCTION

Microfluidic chips are miniature devices that can manipulate tiny amounts of fluids on a small chip and can perform, automatically, various laboratory functions such as dispensing, mixing, filtering, and detection. They play an increasingly important role in today's science and technology, with applications in environmental or medical monitoring, protein or DNA analysis, drug discovery, physiological sample analysis, and cancer research.

These chips often contain modules whose function is to mix fluids. One application where fluid mixing plays a crucial role is sample preparation for some biological or chemical experiments. When preparing such samples, one of the objectives is to produce desired volumes of the fluid of interest, called *reactant*, diluted to some specified concentrations by mixing it with another fluid called *buffer*. As an example, an experimental study may require a sample that consists of $6\mu\text{L}$ of reactant with concentration 10%, $9\mu\text{L}$ of reactant with concentration 20%, and $3\mu\text{L}$ of reactant

Research supported by NSF grant CCF-1536026.

Authors' addresses: M. Coviello Gonzalez and M. Chrobak, University of California at Riverside, Riverside, California, 92521; emails: marek@cs.ucr.edu, mcovi001@ucr.edu.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

© 2020 Copyright held by the owner/author(s).

1084-6654/2020/09-ART1.10

<https://doi.org/10.1145/3408297>

Fig. 1. On the left, a mixing graph G_1 that produces droplet set $\{\frac{1}{8}, \frac{1}{8}, \frac{1}{4}, \frac{13}{16}, \frac{13}{16}, \frac{7}{8}\}$ from input set $I = \{0, 0, 1, 1, 1, 1\}$. Numbers on the micro-mixers (internal nodes) represent droplet concentrations produced by these micro-mixers. If only some of the produced droplets are needed, the remaining droplets are designated as waste. This is illustrated by the mixing graph G_2 that produces droplets $\{\frac{1}{8}, \frac{1}{8}, \frac{13}{16}, \frac{13}{16}\}$. Small black circles labeled “w” on micro-mixers represent droplets of waste.

with concentration 40%. Such multiple-concentration samples are often required in toxicology or pharmaceutical studies, among other applications.

There are different models for fluid mixing in the literature and multiple technologies for manufacturing fluid-mixing microfluidic chips. (See the survey in Ref. [2] or the recent book [1] for more information on different models and algorithmic issues related to fluid mixing.) In this work, we assume the *droplet-based* model, where the fluids are manipulated in discrete quantities called *droplets*. For convenience, we will identify droplets by their reactant concentrations, which are numbers in the interval $[0, 1]$ with finite binary precision. In particular, a droplet of reactant is denoted by 1 and a droplet of buffer by 0. We focus on the mixing technology that utilizes modules called *1-1 micro-mixers*. A micro-mixer has two inlets and two outlets. It receives two droplets of fluid, one in each inlet, mixes these droplets perfectly, and produces two droplets of the mixed fluid, one on each outlet. (Thus, if the inlet droplets have reactant concentrations a and b , then the two outlet droplets each will have concentration $\frac{1}{2}(a + b)$.) Input droplets are injected into the chip via droplet dispensers and output droplets are collected in droplet collectors. All these components are connected via micro-channels that transport droplets, forming, naturally, an acyclic graph that we call a *mixing graph*, whose source nodes are fluid dispensers, internal nodes (of in-degree and out-degree 2) are micro-mixers, and sink nodes are droplet collectors. Graph G_1 in Figure 1 illustrates an example of a mixing graph.

Given some target set of droplets with specified reactant concentrations, the objective is to design a mixing graph that produces these droplets from pure reactant and buffer droplets, while optimizing some objective function. Some target sets can be produced only if we allow the mixing graph to also produce some superfluous amount of fluid that we refer to as *waste*; see graph G_2 in Figure 1. One natural objective function is to minimize the number of waste droplets (or, equivalently, the total number of input droplets). As reactant is typically more expensive than buffer, one other common objective is to minimize the reactant usage. Yet another possibility is to minimize the number of micro-mixers or the depth of the mixing graph. There is growing literature on developing techniques and algorithms for designing such mixing graphs that attempt to optimize some of the above criteria.

State-of-the-art. Most of the earlier papers on this topic studied designing mixing graphs for single-droplet targets. This line of research was pioneered by Thies et al. [12] who proposed an algorithm called Min-Mix that constructs a mixing graph for a single-droplet target with the minimum number of mixing operations. Roy et al. [10] developed an algorithm called DMRW designed to minimize waste. Huang et al. [7] considered minimizing reactant usage, and proposed an algorithm called REMIA. Another algorithm called GORMA, for minimizing reactant usage and based on a branch-and-bound technique, was developed by Chiang et al. [3].

The algorithms listed above are heuristics. They generally exhibit good performance for most inputs, but they do not provide any worst-case performance guarantees for fluid usage. An interesting attempt to develop an algorithm that minimizes waste, for target sets with multiple droplets, was reported by Dinh et al. [4]. Their algorithm, which we refer to as ILP, is based on a reduction to a minimum-cost integer flow problem in a layered graph and, since the resulting integer program could be exponential in the precision d of the target set (and, thus, also in terms of the input size), its worst-case running time is doubly exponential. Further, as this algorithm only considers mixing graphs of depth at most d , it does not always find an optimal solution, as there are examples of target sets with concentration d that require depth larger than d ; see Refs [5] and [6]. In spite of these deficiencies, for very small values of d , this algorithm is still likely to produce good mixing graphs. It was recently shown that for small d 's, its performance can be further improved by increasing the number of layers in the flow graph [5]. (We remark, however, that the example in Ref. [6] proves that for target precision d , this graph would need, at the minimum, $2d$ layers to produce optimal solutions, and it is not known whether this number is sufficient. Our computational experiments indicate that this integer programming approach becomes impractical when the layered graph from Ref. [4] has about 11–12 layers.)

Additional work regarding the design of mixing graphs for multiple droplets includes Huang et al.'s algorithm called WARA, which is an extension of Algorithm REMIA, that focuses on reactant minimization; see Ref. [8]. Mitra et al. [9] also proposed an algorithm for multiple droplet concentrations by modeling the problem as an instance of the Asymmetric TSP on a de Bruijn graph.

As discussed in Ref. [6], the computational complexity of computing mixing graphs with minimum waste is still open, even in the case of single-droplet targets. In fact, it is not even known whether the minimum-waste function is computable at all, or whether it is decidable to determine if a given target set of droplets can be produced without *any* waste. Some progress on computational complexity of designing mixing graphs was recently reported in Ref. [6], where it was shown that there is a polynomial-time algorithm that determines whether a given collection of droplets with specified concentrations can be mixed perfectly using a mixing graph.

Our results. Continuing the line of work in Refs [3], [7], [10], and [12], we develop a new efficient algorithm, *Recursive Precision Reduction with Initial Shift* (RPRIS), for designing mixing graphs for generating a *single-droplet* with a specified target concentration, with the objective to minimize waste. Our algorithm was designed to provide improved worst-case waste estimate; specifically, it aims to cut it by half for most concentrations. Its main idea is quite natural: The algorithm first recursively modifies the target concentration, at each step reducing its precision by 2. It then constructs the mixing graph while backtracking, only adding one waste droplet in each step.

Although designed with worst-case performance in mind, RPRIS also significantly outperformed algorithms Min-Mix, DMRW, and GORMA in our experimental study (see Section 7), producing on average about 50% less waste than Min-Mix, between 21% and 25% less waste than DMRW (with the percentage increasing with the precision d of the target droplet), and about 17% less waste than GORMA. It also produces about 40% less waste than REMIA. Additionally, when compared to ILP (which applies to only very small values of d), RPRIS produces on average only about 7% additional waste.

Unlike earlier work in this area, which was predominantly experimental, we introduce a performance measure for waste minimization algorithms and show that RPRIS has better worst-case performance than Min-Mix and DMRW. This measure is based on two attributes d and γ of the target concentration t . As defined earlier, d is the precision of t , and γ is the length of the leading block of equal bits in t 's binary representation, excluding the least-significant bit 1. For example, if $t = .00001011$, then $\gamma = 4$, and if $t = .1111$, then $\gamma = 3$. (Both d and γ are functions of t , but we skip the argument t , as it is always understood from context.) Intuitively, γ represents the “inevitable” waste that any mixing graph must produce (and, thus, independent of the algorithm used); therefore, the waste performance should be measured in terms of $d - \gamma$ to measure only the “evitable” waste. We elaborate more on this issue in the discussion below.

We show that Algorithm RPRIS works as designed: it produces at most $\frac{1}{2}(d + \gamma) + 2$ droplets of waste (see Theorem 6.1 in Section 6). In comparison, Algorithm Min-Mix from Ref. [12] produces exactly d droplets of waste to produce t , independently of γ . As $\frac{1}{2}(d + \gamma) + 2 = \gamma + \frac{1}{2}(d - \gamma) + 2$ and $d = \gamma + (d - \gamma)$, the evitable waste of RPRIS is about half that of Min-Mix. This also means that the waste of RPRIS is about half that of Min-Mix for *almost all concentrations* t . (Formally, for a uniformly chosen random t with precision d , the probability that the waste is larger than $(\frac{1}{2} - \epsilon)d$ vanishes when d grows, for any $\epsilon > 0$.) As for Algorithm DMRW, its average waste is better than that of Min-Mix, but its worst-case bound is still $d - O(1)$, even for small values of γ (say, when $t \in [\frac{1}{4}, \frac{3}{4}]$), while Algorithm RPRIS' waste is at most $d/2 + O(1)$ in this range.

In regard to time performance, for the problem of computing mixing graphs, it would be reasonable to express the time complexity of an algorithm as a function of its output, which is the size of the produced graph. This is because the output size is at least as large as the input size, which is equal to d — the number of bits of t . (In fact, the output size is typically larger than d .) Algorithm RPRIS runs in time that is linear in the size of the computed graph, and the graphs computed by Algorithm RPRIS have size $O(d^2)$.

We mentioned earlier that performance measures other than waste have been studied in the literature. These measures may provide useful information for practitioners when choosing a mixing graph for the application at hand. Addressing this, in Section 6, we show that our graphs have size $O(d^2)$, and in Section 7, we provide some experimental data comparing our algorithm with some algorithms from the literature in terms of the average reactant usage and graph size. Some unresolved question regarding tradeoffs between different performance measures are discussed in Section 8.

Discussion. To elaborate further on our performance measure for waste, observe that the optimum waste is indeed never smaller than $\gamma + 1$. This is because for any target concentration value t whose binary representation starts with γ 0's, any mixing graph has to use $\gamma + 1$ input droplets 0 and at least one droplet 1. (The case when the leading bits of t are 1's is symmetric.) For this reason, a natural approach is to express the waste in the form $\gamma + f(d - \gamma)$, for some function $f()$, with the first term representing the inevitable waste, and the second term the evitable waste. In Algorithm RPRIS, we have $f(x) \approx \frac{1}{2}x$. It is not known whether smaller waste-bound functions $f()$ can be achieved.

Ideally, one would like to develop efficient “approximation” algorithms for waste minimization, that measure waste performance in terms of the additive or multiplicative approximation error with respect to the optimum value. This is not realistic, however, given the current state of knowledge, since, at this time, no close and computable bounds for the optimum waste are known.

Our work is focused on designing mixing graphs that produce the target concentration exactly. An alternative approach would be to relax this requirement and allow mixing graphs that produce a concentration sufficiently close to the target value. The intuition is that such additional

Fig. 2. Coupling of two mixing graphs G_1 and G_2 . $G_2 \bullet G_1$ is obtained by identifying inlet edges of two sinks of G_1 , one labeled $\frac{1}{4}$ and one $\frac{3}{8}$, with the outlet edges of the corresponding sources of G_2 . These new edges are shown as dotted arrows.

flexibility should help reduce the waste, as well as improve other performance measures. Indeed, this intuition was confirmed experimentally in Ref. [11], where a new algorithm was proposed that computes a mixing graph producing a concentration within a specified error bound. This algorithm compares favorably with earlier algorithms in terms of waste.

2 PRELIMINARIES

We use notation $\text{prec}(c)$ for the precision of concentration c , which is the number of fractional bits in the binary representation of c . (All concentration values will have finite binary representation.) In other words, $\text{prec}(c) = d \in \mathbb{Z}_{\geq 0}$ such that $c = a/2^d$ for an odd $a \in \mathbb{Z}$.

We will deal with sets of droplets, some possibly with equal concentrations. We define a *configuration* as a multiset of droplet concentrations. Let A be an arbitrary configuration. By $|A| = n$, we denote the number of droplets in A . We will often write a configuration as $A = \{f_1 : a_1, f_2 : a_2, \dots, f_m : a_m\}$, where each a_i represents a different concentration and f_i denotes the multiplicity of a_i in A . (If $f_i = 1$, then we will just write “ a_i ” instead of “ $f_i : a_i$ ”.) Naturally, we have $\sum_{i=1}^m f_i = n$.

We defined mixing graphs in the introduction. A mixing graph can be thought of, abstractly, as a linear mapping from the source values (usually 0’s and 1’s) to the sink values. Yet, in the article, for convenience, we will assume that the source concentration vector is part of a mixing graph’s specification, and that all sources, micro-mixers, and sinks are labeled by their associated concentration values.

We now define an operation of graph coupling. Consider two mixing graphs G_1 and G_2 . Let T_1 be the output configuration (the concentration labels of the sink nodes) of G_1 and I_2 be the input configuration (the concentration labels of the source nodes) for G_2 . To construct the *coupling* of G_1 and G_2 , denoted $G_2 \bullet G_1$, we identify inlet edges of the sinks of G_1 with labels from $T_1 \cap I_2$ with outlet edges of the corresponding sources in G_2 . More precisely, repeat the following steps as long as $T_1 \cap I_2 \neq \emptyset$: (1) choose any concentration $a \in T_1 \cap I_2$; (2) choose any sink node u of G_1 labeled a , and let (u', u) be its inlet edge; (3) choose any source node v of G_2 labeled a , and let (v, v') be its outlet edge; (4) remove u and v and their incident edges; and, finally, (5) add edge (u', v') . The remaining sources of G_1 and G_2 become sources of $G_2 \bullet G_1$, and the remaining sinks of G_1 and G_2 become sinks of $G_2 \bullet G_1$. See Figure 2 for an example.

Fig. 4. Graphical representation of intervals $S_1, S_2, \dots, S_5$. The thick shaded part of each interval $S_k = [l, r]$ marks its “middle section” $[l + \frac{1}{16}, r - \frac{1}{16}]$. Each concentration within interval $[\frac{1}{4}, \frac{3}{4}]$ belongs to the middle section of some S_k .

Precision Reduction to t_0 , and then we appropriately modify the computed mixing graph G_0 to obtain the final mixing graph. This process, namely converting t to t_0 and later modifying G_0 , is called *Initial Shift*.

We next describe these two processes in more detail, starting with Recursive Precision Reduction, followed by Initial Shift.

Recursive Precision Reduction (RPR). We start with concentration t_0 that, by applying Initial Shift (described next), we can assume to be in $[\frac{1}{4}, \frac{3}{4}]$.

Step (rpr1): computing t_{s+1} . We convert t_s into a carefully chosen concentration t_{s+1} for which $d_{s+1} = d_s - 2$. One key idea is to maintain an invariant so that at each recursive step, this new concentration value satisfies $t_{s+1} \in [\frac{1}{4}, \frac{3}{4}]$. To accomplish this, we consider five intervals $S_1 = [\frac{1}{8}, \frac{3}{8}]$, $S_2 = [\frac{1}{4}, \frac{1}{2}]$, $S_3 = [\frac{3}{8}, \frac{5}{8}]$, $S_4 = [\frac{1}{2}, \frac{3}{4}]$, and $S_5 = [\frac{5}{8}, \frac{7}{8}]$. We choose an interval S_k that contains t_s “in the middle;” that is $S_k = [l, r]$ for k such that $t_s \in [l + \frac{1}{16}, r - \frac{1}{16}]$. These intervals are illustrated in Figure 4. (For concentrations $t_s \in \{\frac{5}{16}, \frac{7}{16}, \frac{9}{16}, \frac{11}{16}\}$, there are two such intervals S_k ; and for these four values t_s , any of these two S_k can be chosen.) We then compute $t_{s+1} = 4(t_s - l)$. Note that t_{s+1} satisfies both $t_{s+1} \in [\frac{1}{4}, \frac{3}{4}]$ (that is, our invariant) and $d_{s+1} = d_s - 2$.

Step (rpr3): converting G_{s+1} into G_s . Let G_{s+1} be the mixing graph obtained for t_{s+1} in step (rpr2). We first modify G_{s+1} to obtain a graph G'_{s+1} , which is then coupled with an appropriate converter C_{s+1} to obtain mixing graph $G_s = G'_{s+1} \bullet C_{s+1}$. Figure 5 illustrates this process.

Next, we explain how to construct G'_{s+1} . G'_{s+1} consists of the same nodes and edges as G_{s+1} , only the concentration labels are changed. Specifically, every concentration label c from G_{s+1} is changed to $l + c/4$ in G'_{s+1} . Note that this is simply the inverse of the linear function that maps t_s to t_{s+1} . In particular, this will map the 0- and 1-labels of the source nodes in G_{s+1} to the endpoints l and r of the corresponding interval S_k .

The converter C_{s+1} used in G_s needs to have sink nodes with labels equal to the source nodes for G'_{s+1} . That is, if the labeling of the source nodes of G'_{s+1} is $I'_{s+1} = \{i : l, j : r\}$, then C_{s+1} will be an $(i : l, j : r)$ -converter. As a general rule, C_{s+1} should produce at most one waste droplet, but there will be some exceptional cases where it produces two. (Nonetheless, we will show that, at most, one such “bad” converter is used during the RPR process.) The construction of these converters is somewhat intricate, and is deferred to the next section.

Two examples of our construction are given in Section 5, after all details of the algorithm are provided.

The base case. We now specify the set of base concentration values and their mixing graphs. Let $\mathcal{B} = \{\frac{1}{2}, \frac{1}{4}, \frac{3}{4}, \frac{3}{8}, \frac{5}{8}, \frac{5}{16}, \frac{11}{16}\}$. (Concentrations $\frac{5}{16}$ and $\frac{11}{16}$ are not strictly necessary for correctness but are included in the base case to improve the waste bound.) Figure 6 illustrates the mixing graphs for concentrations $\frac{1}{2}, \frac{1}{4}, \frac{3}{8}$, and $\frac{5}{16}$; the mixing graphs for the remaining concentrations are symmetric.

Fig. 5. Conversion from G_{s+1} to G_s . The left image illustrates the computed mixing graph G_{s+1} with input labels I_{s+1} (consisting of only 0's and 1's) that produces t_{s+1} along with some waste. The middle figure illustrates G'_{s+1} , which is obtained from G_{s+1} by changing concentration labels. The last figure illustrates the complete mixing graph $G_s = G'_{s+1} \bullet C_{s+1}$ for t_s , shown within a dotted rectangle.

Fig. 6. Base mixing graphs B_1, B_2, B_3 , and B_4 for concentrations $\frac{1}{2}, \frac{1}{4}, \frac{3}{8}$, and $\frac{5}{16}$, respectively.

Initial Shift (IS). At the fundamental level, the idea of the IS procedure is similar to a single step of RPR, although the involved linear mappings and the converter are significantly different.

We can assume that $t < \frac{1}{4}$ (because for $t > \frac{3}{4}$, the process is symmetric). Thus, the binary representation of t starts with $\gamma \geq 2$ fractional 0's. Since $2^{\gamma-1}t \in [\frac{1}{4}, \frac{1}{2})$, we could use this value as the result of the initial shift, but to improve the waste bound, we refine this choice as follows: Let $\sigma \in \{0, 1\}$ be the $(\gamma + 2)^{th}$ bit in t 's binary representation (namely, the bit that follows the first 1). If $2^{\gamma-1}t \in [\frac{3}{8}, \frac{1}{2})$, then let $t_0 = 2^{\gamma-1}t$ and $\sigma = 1$. Otherwise, we have $2^{\gamma-1}t \in [\frac{1}{4}, \frac{3}{8})$, in which case we let $t_0 = 2^{\gamma}t$ and $\sigma = 0$. In either case, $t_0 = 2^{\gamma-\sigma}t \in [\frac{1}{4}, \frac{3}{4}]$ and $d_0 = d - \gamma + \sigma$.

Let G_0 be the mixing graph obtained by applying the RPR process to t_0 . It remains to show how to modify G_0 to obtain the mixing graph G for t . This is analogous to the process shown in Figure 5. We first construct a mixing graph G'_0 that consists of the same nodes and edges as G_0 , only each concentration label c is replaced by $c/2^{\gamma-\sigma}$. Thus, the label set of the source nodes in G'_0 will have the form $I'_0 = \{i : 0, j : 1/2^{\gamma-\sigma}\}$. We then construct a $(i : 0, j : 1/2^{\gamma-\sigma})$ -converter C_0 and couple it with G'_0 to obtain G ; that is, $G = G'_0 \bullet C_0$. This C_0 is easy to construct: The i 0's in I'_0

Fig. 7. Initial converters and extenders for the case $I = \{i : \frac{1}{4}, j : \frac{1}{2}\}$.

don't require any mixing, and to produce the j droplets $1/2^{\gamma-\sigma}$ we start with a sufficient number of 1's and repeatedly mix them with 0's, making sure to generate at most one waste droplet at each step. More specifically, after z rounds, we will have j_z droplets with concentration $1/2^z$, where $j_z = \lceil j/2^{\gamma-\sigma-z} \rceil$. In particular, at the beginning, we have $z = 0$ and $j_0 = \lceil j/2^{\gamma-\sigma} \rceil$ —the total reactant volume in I'_0 rounded up to the nearest integer. In round z , we have j_z parallel mixing operations that mix these j_z droplets with j_z new 0's, producing $2j_z$ droplets with concentration $1/2^{z+1}$. We then either have $j_{z+1} = 2j_z$, in which case there is no waste, or $j_{z+1} = 2j_z - 1$, in which case one waste droplet $1/2^{z+1}$ is produced. After round $\gamma - \sigma$, we have $j_{\gamma-\sigma} = \lceil j \rceil = j$. Overall, C_0 produces at most $\gamma - \sigma$ waste droplets.

4 CONSTRUCTION OF CONVERTERS

In this section, we detail the construction of the converters used to transform G_s into G_{s+1} . Let t_s denote the target droplet's concentration at the s^{th} recursive step in the RPR process. We can assume that $t_s \in [\frac{1}{4}, \frac{1}{2}]$, because the case $t_s \in (\frac{1}{2}, \frac{3}{4}]$ is symmetric. Recall that for a t_s in this range, in Step (rpr1), we will choose an appropriate interval S_k for some $k \in \{1, 2, 3\}$. Let $S_k = [l, r]$ (that is, $l = k \cdot \frac{1}{8}$ and $r = l + \frac{1}{4}$). For each such k and all $i, j \geq 1$, we give a construction of an $(i : l, j : r)$ -converter that we will denote $C_{i,j}^k$. Our main objective here is to design these converters so that they produce as little waste as possible—ideally, none.

4.1 $(i : \frac{1}{4}, j : \frac{1}{2})$ -Converters $C_{i,j}^2$

We start with the case $k = 2$ because, in this case, the construction is relatively simple. We show how to construct, for all $i, j \geq 1$, our $(i : \frac{1}{4}, j : \frac{1}{2})$ -converter $C_{i,j}^2$, that produces at most one droplet of waste. These converters are constructed via an iterative process. We first give initial converters $C_{i,j}^2$, for some small values of i and j , by providing specific graphs. All other converters are obtained from these initial converters by repeatedly coupling them with other mixing graphs that we refer to as *extenders*.

Let $J_{\text{init}}^2 = \{(i, j)\}_{i,j \in \{1,2\}}$. The initial converters $C_{i,j}^2$ are defined for the four index pairs $(i, j) \in J_{\text{init}}^2$. Figure 7 illustrates the initial converters $C_{2,1}^2, C_{1,2}^2$ and two extenders X_1^2, X_2^2 . Converter $C_{1,2}^2$ produces one waste droplet and converter $C_{2,1}^2$ does not produce any waste. Converter $C_{1,2}^2$ can be obtained from $C_{2,1}^2$ by designating one of the $\frac{1}{4}$ outputs as waste. Converter $C_{2,2}^2$ is defined as $C_{2,2}^2 = X_1^2 \bullet C_{2,1}^2$, and produces one waste droplet of $\frac{1}{2}$. (Thus, $C_{2,2}^2$ is simply a disjoint union of $C_{2,1}^2$ and X_1^2 with one output $\frac{1}{2}$ designated as waste.)

Fig. 8. Initial converters for the case $I = \{i : \frac{3}{8}, j : \frac{5}{8}\}$.

The construction of other converters $C_{i,j}^2$ is based on the following observation: Suppose that we already have constructed some $C_{i,j}^2$. Then, (i) $X_1^2 \bullet C_{i,j}^2$ is a $C_{i,j+2}^2$ converter that produces the same waste as $C_{i,j}^2$, and (ii) provided that $j \geq 2$, $X_2^2 \bullet C_{i,j}^2$ is a $C_{i+2,j-1}^2$ converter that produces the same waste as $C_{i,j}^2$.

Let now $i, j \geq 1$ with $(i, j) \notin J_{\text{init}}^2$ be arbitrary. To construct $C_{i,j}^2$, using the initial converters and the above observation, express the integer vector (i, j) as $(i, j) = (i', j') + \phi(0, 2) + \psi(2, -1)$, for some $i', j' \in J_{\text{init}}^2$ and integers $\psi = \lceil \frac{i}{2} \rceil - 1$ and $\phi = \lceil \frac{j+\psi}{2} \rceil - 1$. Then, $C_{i,j}^2$ is constructed by starting with $C_{i',j'}^2$ and coupling it ϕ times with X_1^2 and then ψ times with X_2^2 . (This order of coupling is not unique but is also not arbitrary, because each extender X_2^2 requires a droplet of concentration $\frac{1}{2}$ as input.) Since X_1^2 and X_2^2 do not produce waste, $C_{i,j}^2$ will produce at most one waste droplet.

4.2 $(i : \frac{3}{8}, j : \frac{5}{8})$ -Converters $C_{i,j}^3$

Next, for each pair $i, j \geq 1$, we construct an $(i : \frac{3}{8}, j : \frac{5}{8})$ -converter $C_{i,j}^3$. These converters are designed to produce one droplet of waste. ($C_{1,1}^3$ will be an exception; see the discussion below). Our approach follows the scheme from Section 4.1: we start with some initial converters, which then can be repeatedly coupled with appropriate extenders to produce all other converters. Since concentrations $\frac{3}{8}$ and $\frac{5}{8}$ are symmetric (as $\frac{5}{8} = 1 - \frac{3}{8}$), we will only show the construction of converters $C_{i,j}^3$ for $i \geq j$; the remaining converters can be computed using symmetric mixing graphs.

Let $J_{\text{init}}^3 = \{(i, 1)\}_{i \in \{1, 2, \dots, 9\}} \cup \{(2, 2)\}$. The initial converters $C_{i,j}^2$ are defined for all index pairs $(i, j) \in J_{\text{init}}^3$. Figure 8 shows converters $C_{3,1}^3, C_{4,1}^3, \dots, C_{7,1}^3$ and $C_{9,1}^3$. Converter $C_{8,1}^3$ can be obtained from $C_{9,1}^3$ by designating an output of $\frac{3}{8}$ as waste. Converter $C_{2,2}^3$ is almost identical to X_1^3 in Figure 9; except that the source labels $\frac{3}{8}$ and $\frac{5}{8}$ are replaced by 0 and 1, respectively (the result of mixing is still $\frac{1}{2}$, so other concentrations in the graph are not affected). Converters $C_{1,1}^3$ and $C_{2,1}^3$ are obtained from $C_{2,2}^3$ by designating outputs of $\{\frac{3}{8}, \frac{5}{8}\}$ and $\frac{5}{8}$, respectively, as waste. Note that all initial converters except for $C_{1,1}^3$ produce at most one droplet of waste.

Now, consider extenders X_1^3 and X_2^3 in Figure 9. The construction of other converters $C_{i,j}^3$ follows the next observation: Assume that we have already constructed some $C_{i,j}^3$, with $i \geq j$. Then,

Fig. 9. X_1^3 and X_2^3 extenders for the case $I = \{i : \frac{3}{8}, j : \frac{5}{8}\}$.

(i) $X_1^3 \bullet C_{i,j}^3$ is a $C_{i+1,j+1}^3$ converter that produces the same waste as $C_{i,j}^3$, and (ii) $X_2^3 \bullet C_{i,j}^3$ is a $C_{i+8,j}^3$ converter that produces the same waste as $C_{i,j}^3$.

Consider now arbitrary $i \geq j \geq 1$ with $(i, j) \notin J_{\text{init}}^3$. To construct $C_{i,j}^3$, using the initial converters and the above observation, express the integer vector (i, j) as $(i, j) = (i', j') + \phi(1, 1) + \psi(8, 0)$, for some integers $\phi, \psi \geq 0$, and $(i', j') \in J_{\text{init}}^3 - \{(1, 1)\}$. Specifically, for $i = j \geq 2$, let $\phi = j - 2$ and $\psi = 0$, and for $i > j \geq 1$, let $\phi = j - 1$ and $\psi = \lfloor (i - \phi - 2)/8 \rfloor$. Then, $C_{i,j}^3$ is constructed by starting with $C_{i',j'}^3$ and coupling it ϕ times with X_1^3 and then ψ times with X_2^3 (in arbitrary order). Since X_1^3 and X_2^3 do not produce waste (and we do not use the initial converter $C_{1,1}^3$), $C_{i,j}^3$ will produce at most one waste droplet.

Overall, all converters $C_{i,j}^3$, except for $C_{1,1}^3$ produce at most one waste droplet. Converter $C_{1,1}^3$ produces two droplets of waste; however, as we later show in Section 6, it is not actually used in the algorithm.

4.3 $(i : \frac{1}{8}, j : \frac{3}{8})$ -Converters $C_{i,j}^1$

In this section, for each pair $i, j \geq 1$, we construct an $(i : \frac{1}{8}, j : \frac{3}{8})$ -converter $C_{i,j}^1$. Most of these converters produce at most one droplet of waste, but there will be four exceptional converters with waste two. (See the comments at the end of this section.) The idea of the construction follows the same scheme as in Sections 4.1 and 4.2: we start with some initial converters and repeatedly couple them with appropriate extenders to obtain other converters.

Let $J_{\text{init}}^1 = \{(i, j)\}_{i,j \in \{1,2,3\}} \cup \{(4, 2), (2, 5)\}$. The initial converters $C_{i,j}^1$ are defined for all index pairs $(i, j) \in J_{\text{init}}^1$. Converters $C_{2,2}^1, C_{2,3}^1, C_{2,5}^1, C_{3,1}^1, C_{3,3}^1$, and $C_{4,2}^1$ are shown in Figure 10. Converters $C_{1,1}^1, C_{1,2}^1$, and $C_{2,1}^1$ are obtained from $C_{2,2}^1$ by designating outputs of $\{\frac{1}{8}, \frac{3}{8}\}, \frac{1}{8}$, and $\frac{3}{8}$, respectively, as waste. Converter $C_{1,3}^1$ is obtained from $C_{2,3}^1$ by designating an output of $\frac{1}{8}$ as waste, and $C_{3,2}^1$ is obtained from $C_{4,2}^1$ by designating an output of $\frac{1}{8}$ as waste. Thus, among the initial converters, $C_{1,1}^1, C_{1,3}^1$, and $C_{3,2}^1$ each produces two droplets of waste; all other converters have at most one droplet of waste.

Next, we provide an observation leading to the construction of other converters $C_{i,j}^1$. Consider extenders X_1^1 and X_2^1 in Figure 11, and assume that we have already constructed some $C_{i,j}^1$. Then, (i) provided that $j \geq 2$, $X_1^1 \bullet C_{i,j}^1$ is a $C_{i+3,j-1}^1$ converter that produces the same waste as $C_{i,j}^1$, and (ii) provided that $i \geq 2$, $X_2^1 \bullet C_{i,j}^1$ is a $C_{i-1,j+3}^1$ converter that produces the same waste as $C_{i,j}^1$. We also need the following, less obvious observation:

Fig. 10. Initial converters for the case $I = \{i : \frac{1}{8}, j : \frac{3}{8}\}$. Fig. 11. X_1^1 and X_2^1 extenders for the case $I = \{i : \frac{1}{8}, j : \frac{3}{8}\}$.

OBSERVATION 1. If $i, j \geq 1$ and $(i, j) \notin J_{\text{init}}^1 \cup \{(6, 1)\}$, then $(i, j) = (i', j') + \phi(-1, 3) + \psi(3, -1)$, for some integers $\phi, \psi \geq 0$, and $(i', j') \in J_{\text{init}}^1 - \{(1, 1), (1, 3), (3, 2)\}$.

PROOF. Let $i, j \geq 1$ and $(i, j) \notin J_{\text{init}}^1 \cup \{(6, 1)\}$. We show first that we can represent (i, j) as $(i, j) = (\tilde{i}, \tilde{j}) + \tilde{\phi}(-1, 3) + \tilde{\psi}(3, -1)$, for $(\tilde{i}, \tilde{j}) \in J_{\text{init}}^1 - \{(1, 1), (2, 5), (4, 2)\}$ and integers $\tilde{\phi}, \tilde{\psi} \geq 0$. Indeed, we let $\tilde{\phi} = \lceil (i + 3j - 4)/8 \rceil - 1$ and $\tilde{\psi} = \lceil (3i + j - 4)/8 \rceil - 1$. For these values of $\tilde{\phi}$ and $\tilde{\psi}$, by routine verification, we end up with $(\tilde{i}, \tilde{j}) \in J_{\text{init}}^1 - \{(1, 1), (2, 5), (4, 2)\}$. (For example, a simple argument can be obtained by expressing $\tilde{\phi}$ and $\tilde{\psi}$ as $\tilde{\phi} = (i + 3j - 4)/8 - \epsilon_1$ and $\tilde{\psi} = (3i + j - 4)/8 - \epsilon_2$, for $\epsilon_1, \epsilon_2 \in (0, 1]$, which gives $(\tilde{i}, \tilde{j}) = (1 - \epsilon_1 + 3\epsilon_2, 1 + 3\epsilon_1 - \epsilon_2)$. Then, use the assumption about the ranges of ϵ_1, ϵ_2 and the integrality of $\tilde{i}$ and $\tilde{j}$.)

Now, if we also have $(\tilde{i}, \tilde{j}) \notin \{(1, 3), (3, 2)\}$, then we are done. Otherwise, we show how to modify the values of parameters $\tilde{i}, \tilde{j}, \tilde{\phi}$ and $\tilde{\psi}$ so that they satisfy the condition in the observation.

Case 1: $(\tilde{i}, \tilde{j}) = (1, 3)$. For this case, $\tilde{\psi} \geq 1$ must hold, because $i \geq 1$. Therefore, we can write (i, j) as $(i, j) = (4, 2) + \tilde{\phi}(-1, 3) + (\tilde{\psi} - 1)(3, -1)$.

Case 2: $(\tilde{i}, \tilde{j}) = (3, 2)$. For this case, it is sufficient to prove that $\tilde{\phi} \geq 1$, since then we could write (i, j) as $(i, j) = (2, 5) + (\tilde{\phi} - 1)(-1, 3) + \tilde{\psi}(3, -1)$. To show that $\tilde{\phi} \geq 1$, we argue, by contradiction,

Fig. 12. Using Procedure RPR() to construct a mixing graph for $t = \frac{11}{32}$. The computed mixing graph G_0 is shown on the right. To emphasize the correspondence to Figure 5, the drawing of G_0 shows the concentration values on the microchannels connecting $C_{2,1}^2$ to G'_1 . The total waste of G_0 is 2, while the precision of t is 5.

as follows. Suppose that $\tilde{\phi} = 0$. Then, $(i, j) = (3, 2) + \tilde{\psi}(3, -1)$. For $\tilde{\psi} \in \{0, 1\}$, this contradicts that $(i, j) \notin J_{\text{init}}^1 \cup \{(6, 1)\}$, and for $\tilde{\psi} \geq 2$, it contradicts that $j \geq 1$. $\square$

Using the observations above, for any pairs $i, j \geq 1$ such that $(i, j) \notin J_{\text{init}}^1$, we can construct converter $C_{i,j}^1$ as follows. If $(i, j) = (6, 1)$ we let $C_{6,1}^1 = X_1^1 \bullet C_{3,2}^1$ (so $C_{6,1}^1$ has two droplets of waste). If $(i, j) \neq (6, 1)$, we construct $C_{i,j}^1$ by starting with $C_{i',j'}^1$ and repeatedly coupling it with ϕ copies of X_2^1 and ψ copies of X_1^1 , choosing a suitable order of couplings to ensure that each intermediate converter has at least one output $\frac{1}{8}$ and at least one $\frac{3}{8}$. (For example, if $j' = 1$, then we begin by coupling X_2^1 first.) As X_1^1 and X_2^1 do not produce any waste, these $C_{i,j}^1$'s will each produce at most one droplet of waste.

Overall, the converters $C_{i,j}^1$ we construct have at most one droplet of waste, with the exception of the following four: $C_{1,1}^1$, $C_{1,3}^1$, $C_{3,2}^1$, and $C_{6,1}^1$. (It is easy to prove that for these converters, waste 2 cannot be avoided.) As we show later in Section 6, of these four converters, only $C_{1,3}^1$ is actually used in the RPR process of Algorithm RPRIS, and it is used at most once.

5 EXAMPLES

We now illustrate our Algorithm RPRIS with two examples. We first give a simple but complete example that involves only one recursive step. In the second example, there are multiple recursive steps. For this example, we tabulate the steps in the construction of the mixing graph and we illustrate graphically its high-level structure.

5.1 Example 1: $t = \frac{11}{32}$

Let $t = \frac{11}{32}$. As $t \in [\frac{1}{4}, \frac{3}{4}]$, there is no need for the initial shift. We start with $s = 0$, where $t_0 = t$. The construction is illustrated in Figure 12.

Fig. 13. A high-level illustration of the mixing graph G_0 for target concentration $t = \frac{1509}{2048}$. To conserve space, we draw G_0 horizontally. $\dot{C}_{i,j}^k$ denotes converter $C_{i,j}^k$ with all concentration labels adjusted according to the algorithm. We use the same convention for $\dot{B}_5$. Arrows represent micro-channels connecting consecutive modules. The numbers above and below the arrows represent concentration values of the droplets transmitted through those micro-channels. For example, there are three droplets of $\frac{5}{8}$ and two droplets of $\frac{7}{8}$ flowing from the first converter $\dot{C}_{3,2}^5$ to the second converter $\dot{C}_{3,1}^3$. Circled letters W represent droplets of waste. Thus, the total waste of G_0 is 5, which is less than half of the precision 11 of t .

In Procedure $\text{RPR}(t_0)$, according to the description of step (rpr1), we first need to choose the interval S_k that contains t_0 in its middle part. Since $t_0 \in [\frac{1}{4} + \frac{1}{16}, \frac{1}{2} - \frac{1}{16}]$, we choose $S_2 = [\frac{1}{4}, \frac{1}{2}]$. This will give us $t_1 = 4(\frac{11}{32} - \frac{1}{4}) = \frac{3}{8}$, and we make a recursive call to Procedure $\text{RPR}(t_1)$.

In Procedure $\text{RPR}(t_1)$, as $t_1 \in \mathcal{B}$, we let G_1 be graph B_3 given in Figure 6. This completes this recursive call, and we backtrack to Procedure $\text{RPR}(t_0)$.

Continuing Procedure $\text{RPR}(t_0)$, the algorithm will convert G_1 into G'_1 by changing each concentration label c to $\frac{1}{4} + c/4$. Next, we need to identify the appropriate converter. As $k = 2$ and the inputs to G'_1 are $\{2 : \frac{1}{4}, 1 : \frac{1}{2}\}$, we use converter $C_{2,1}^2$. Combining $C_{2,1}^2$ with G'_1 , we obtain the final graph G_0 .

5.2 Example 2: $t = \frac{1509}{2048}$

In this example we use Algorithm RPRIS to construct a mixing graph for target concentration $t = \frac{1509}{2048}$. So, initially, $t_0 = t = \frac{1509}{2048}$. The table below shows the consecutive values of t_s , and its corresponding intervals S_k and converters. The converters in this table are defined in Section 4: $C_{3,2}^5$ is symmetric to $C_{2,3}^1$ in Figure 10, $C_{3,1}^3$ is given in Figure 8, $C_{1,2}^1$ is described in Section 4.3, and $C_{1,2}^4$ is symmetric to $C_{2,1}^2$ in Figure 7. Graph B_5 in the last column is the base mixing graph for $\frac{5}{8}$ that is symmetric to graph B_3 in Figure 6.

target	$t_0 = \frac{1509}{2048}$	$t_1 = \frac{229}{512}$	$t_2 = \frac{37}{128}$	$t_3 = \frac{21}{32}$	$t_4 = \frac{5}{8}$
interval	$S_5 = [\frac{5}{8}, \frac{7}{8}]$	$S_3 = [\frac{3}{8}, \frac{5}{8}]$	$S_1 = [\frac{1}{8}, \frac{3}{8}]$	$S_4 = [\frac{1}{2}, \frac{3}{4}]$	(base case)
converter/base graph	$C_{3,2}^5$	$C_{3,1}^3$	$C_{1,2}^1$	$C_{1,2}^4$	B_5

Recall that for each t_s , its interval S_k is the one that contains t_s in its middle section. The converters are added during the backtracking phase. If the interval selected for t_s is S_k and if G_{s+1} has inputs $\{i : 0, j : 1\}$, then in step s , we attach converter $C_{i,j}^k$ to G_{s+1} to obtain G_s . The computed graph G_0 is illustrated schematically in Figure 13.

6 PERFORMANCE BOUNDS

In this section, we provide the analysis of Algorithm RPRIS , including the worst-case bound on produced waste, a bound on the size of computed mixing graphs, and the running time.

Bound on waste. We first estimate the number of waste droplets of Algorithm RPRIS. Let G be the mixing graph constructed by RPRIS for a target concentration t with its corresponding values $d = \text{prec}(t)$ and γ (as defined in Section 1). Below, we prove the following theorem.

THEOREM 6.1. *The number of waste droplets in G is at most $\frac{1}{2}(d + \gamma) + 2$.*

To prove Theorem 6.1, we show that the total number of sink nodes in G is at most $\frac{1}{2}(d + \gamma - \sigma) + 3$, for the value of $\sigma \in \{0, 1\}$ corresponding to t , as defined in Section 3. This is sufficient, as one sink node is used to produce t .

Following the algorithm's description in Section 3, let $G = G'_0 \bullet C_0$. From our construction of C_0 (at the end of Section 3), we get that C_0 contributes at most $\gamma - \sigma$ sink nodes to G . (Each waste droplet produced by C_0 represents a sink node in G .) Therefore, to prove Theorem 6.1, it remains to show that G'_0 contains at most $\frac{1}{2}(d - \gamma + \sigma) + 3$ sink nodes. This is equivalent to showing that G_0 , computed by process RPR for t_0 (and used to compute G'_0), contains at most $\frac{1}{2}d_0 + 3$ sink nodes, where $d_0 = \text{prec}(t_0) = d - \gamma + \sigma$. Lemma 6.2 next proves this claim.

LEMMA 6.2. *The number of sink nodes in G_0 is at most $\frac{1}{2}d_0 + 3$.*

PROOF. Let t_b be the concentration used for the base case of the RPR process and $d_b = \text{prec}(t_b) \leq d_0$ its precision. We prove the lemma in three steps:

- (i) First, we observe that the number of sink nodes in the base mixing graph for t_b is at most 3.
- (ii) Then, we show that if $t_0 \neq t_b$, then the number of converters used in the construction of G_0 is at most $\frac{1}{2}d_0 - 1$.
- (iii) Thirdly, we argue that at most one of the converters in (ii) contains two waste sink nodes.

Note that the claims in (i), (ii), and (iii) indeed give us a complete proof for Lemma 6.2. Part (i) implies the lemma for $t_0 = t_b$. So assume that $t_0 \neq t_b$. Then, from (ii) and (iii), all converters in the construction of G_0 , except possibly one, will have at most one waste sink, and this one other converter has at most two waste sinks. Accounting also for the base mixing graph, the total number of sinks can be bounded by $3 + (\frac{1}{2}d_0 - 2) + 2 = \frac{1}{2}d_0 + 3$, as claimed in the lemma.

It is thus sufficient to prove the claims in (i), (ii), and (iii).

The proof of (i) is by straightforward inspection. By definition of the base case, $t_b \in \mathcal{B} = \{\frac{1}{2}, \frac{1}{4}, \frac{3}{4}, \frac{3}{8}, \frac{5}{8}, \frac{5}{16}, \frac{11}{16}\}$. The mixing graphs for base concentrations are shown in Figure 6. (The graphs for $\frac{3}{4}$, $\frac{5}{8}$, and $\frac{11}{16}$ are symmetric to B_2 , B_3 , and B_4 .) All these graphs have at most three sink nodes.

Next, we prove part (ii). In each step of the RPR process, we reduce the precision of the target concentration by two until we reach the base case, which gives us that the number of converters is exactly $\frac{1}{2}(d_0 - d_b)$. It is thus sufficient to show that $d_b \geq 2$, as this immediately implies (ii). Indeed, the assumption that $t_0 \neq t_b$ and the definition of the base case implies that $d_0 \geq 4$. (This is because the algorithm maintains the invariant that its target concentration is in $[\frac{1}{4}, \frac{3}{4}]$ and all concentrations in this interval with precision at most three are in $\mathcal{B}$.) This, and the precision of the target concentration decreasing by exactly two in each step of the recursion, imply that $d_b \in \{2, 3\}$ holds.

We now address part (iii). First, we observe that converters $C_{1,1}^k$ are not used in the construction of G_0 . This can be proved by contradiction: Suppose that we did use some $C_{1,1}^k$ in the construction of G_0 , say that $G_s = G'_{s+1} \bullet C_{s+1}$ where $C_{s+1} = C_{1,1}^k$. This $C_{1,1}^k$ produces output concentrations $I'_{s+1} = \{l : 1, r : 1\}$. This means that the source labels for G_{s+1} are $\{0, 1\}$ and, therefore, G_{s+1} would be the base-case graph B_1 (see Figure 6). Hence, $b = s + 1$ and $t_b = \frac{1}{2}$. This means that at step s , the target concentration was $t_s = \frac{1}{2}(l + r)$. Therefore, by the definition of $S_k = [l, r]$ for $k \in \{1, 2, 3, 4, 5\}$, we

would have $t_s \in \{\frac{1}{4}, \frac{3}{8}, \frac{1}{2}, \frac{5}{8}, \frac{3}{4}\} \subset \mathcal{B}$, so Algorithm RPRIS would actually use a base-case mixing graph for t_s , instead of continuing the recursion.

Therefore, to show (iii), it is sufficient to consider $C_{i,j}^k$ converters that satisfy $i + j \geq 3$ with $i, j \geq 1$. Now, from Sections 4.1, 4.2, and 4.3, we observe that the only such converters that contain two waste sink nodes are $C_{1,3}^1$, $C_{3,2}^1$, and $C_{6,1}^1$. Claim 1 below shows that converters $C_{6,1}^1$ and $C_{3,2}^1$ are not used in the construction of G_0 . (Note that there are symmetric converters $C_{3,1}^5$, $C_{2,3}^5$, and $C_{1,6}^5$, where superscript 5 is associated with interval S_5 . A similar argument holds for these converters.)

Regarding $C_{1,3}^1$, first we note that this converter has exactly six source nodes and four output nodes; see $C_{2,3}^1$ in Figure 10 on Section 4.3. ($C_{1,3}^1$ is obtained from $C_{2,3}^1$ by designating an output of $\frac{1}{8}$ as waste.) This implies that $C_{1,3}^1$ can not be used more than once in the construction of G_0 , since the number of source nodes at each recursive step in the RPR process is non-increasing. Thus, step (iii) holds.

CLAIM 1. *Converters $C_{6,1}^1$ and $C_{3,2}^1$ are not used by Algorithm RPRIS.*

We first present the following observations. Consider the s^{th} recursive step of the RPR process, when we construct graph $G_s = G'_{s+1} \bullet C_{s+1}$ for target concentration t_s . If a converter $C_{i,j}^1$ is used in this step, that is $C_{s+1} = C_{i,j}^1$, then $t_s \in (\frac{1}{4}, \frac{5}{16}]$ must hold; that is, t_s is in the middle part of interval S_1 (see Figure 4 in Section 3). (Recall that by our algorithm's invariant, $t_s \in [\frac{1}{4}, \frac{3}{4}]$. Also, note that $t_s \neq \frac{1}{4}$ since, otherwise, this would be a base case and the algorithm would use B_2 from Figure 6 instead.) Further, at the next step of the RPR process, the target concentration $t_{s+1} = 4(t_s - \frac{1}{8})$ satisfies $t_{s+1} \in (\frac{1}{2}, \frac{3}{4}]$.

We now prove the claim by contradiction, using the above observations. Assume that either $C_{6,1}^1$ or $C_{3,2}^1$ was used in the construction of G_0 .

If $C_{6,1}^1$ was used in step s of the RPR process, then at the next recursive step, for G_{s+1} , the concentration labels of the source nodes would be $\{6 : 0, 1\}$. Since $t_{s+1} > \frac{1}{2}$, this set does not have enough reactant to produce t_{s+1} —contradiction.

Next, suppose that $C_{3,2}^1$ was used in step s of the RPR process. Then, the concentration labels of the source nodes at the next recursive step are $\{3 : 0, 2 : 1\}$. This implies that the next step, $s + 1$, is guaranteed not to be a base case, since all mixing graphs used for base case concentrations contain at most three source nodes, as illustrated in Figure 6. Now, as $t_{s+1} > \frac{1}{2}$, depending on the exact value of t_{s+1} , the chosen interval for t_{s+1} must be either $S_3 = [\frac{3}{8}, \frac{5}{8}]$, $S_4 = [\frac{1}{2}, \frac{3}{4}]$, or $S_5 = [\frac{5}{8}, \frac{7}{8}]$. We now consider these three cases in which we examine what happens in step $s + 1$, when G_{s+1} is constructed.

Case 1: $t_{s+1} \in (\frac{1}{2}, \frac{9}{16}]$. Then, the chosen interval in step $s + 1$ is $S_3 = [\frac{3}{8}, \frac{5}{8}]$. The only $C_{i,j}^3$ converter with source concentration labels $\{3 : 0, 2 : 1\}$ is $C_{3,1}^3$ (see Figure 8 in Section 4.2), whose sink nodes have concentration labels $\{3 : \frac{3}{8}, \frac{5}{8}\}$ (not counting the waste droplet $\frac{1}{4}$). Therefore, the input configuration for the second-next recursive step $s + 2$ will be $\{3 : 0, 1\}$. But this set does not have enough reactant to produce $t_{s+2} = 4(t_{s+1} - \frac{3}{8}) > \frac{1}{2}$, giving us a contradiction with the choice of S_3 for t_{s+1} .

Case 2: $t_{s+1} \in (\frac{9}{16}, \frac{11}{16}]$. Then, the chosen interval is $S_4 = [\frac{1}{2}, \frac{3}{4}]$. This instance is symmetric to interval $S_2 = [\frac{1}{4}, \frac{1}{2}]$, having source concentration labels $\{2 : 0, 3 : 1\}$, instead of $\{3 : 0, 2 : 1\}$, and target concentration $t'_{s+1} = (1 - t_{s+1}) \in [\frac{5}{16}, \frac{7}{16}]$. Thus, we proceed accordingly, giving an argument for t'_{s+1} . Since every initial converter and extender in Section 4.1 has at least the same number of source nodes with concentration label 0 as source nodes with concentration label 1, no converter $C_{i,j}^2$ constructed by the algorithm will have

source concentration labels $\{2 : 0, 3 : 1\}$. Hence, we have a contradiction with the choice of S_2 for t'_{s+1} , and, thus, also with the choice of S_4 for t_{s+1} .

Case 3: $t_{s+1} \in (\frac{11}{16}, \frac{3}{4}]$. Then, the chosen interval is $S_5 = [\frac{5}{8}, \frac{7}{8}]$. The argument here is simple: to produce concentration $\frac{7}{8}$, at least three reactant droplets are needed, but the input configuration contains only two. Therefore, at the next recursive step, the algorithm will not have enough reactant droplets to construct a converter $C_{i,j}^5$ with $i, j \geq 1$, contradicting the choice of S_5 for t_{s+1} .

Summarizing, neither S_3, S_4 , nor S_5 will be chosen by our algorithm for t_{s+1} , contradicting $C_{3,2}^1$ being used in the s^{th} step of the construction of G_0 .

This completes the proof of Claim 1 and Lemma 6.2 (thus, also completing the proof of Theorem 6.1). $\square$

Size of mixing graphs. Let $G = G'_0 \bullet C_0$ be the mixing graph computed by Algorithm RPRIS for a target concentration t ; recall that, as detailed in Section 3, C_0 is constructed by process IS while G'_0 is obtained from G_0 (constructed by process RPR) by changing its concentration labels appropriately. The target concentration of G_0 is t_0 with precision d_0 and its input set of concentrations is I_0 . We claim that the size of G satisfies $|G| = O(d^2)$, for $d = \text{prec}(t)$. To prove this, we show that $|G'_0| = |G_0| = O(d_0^2)$ and $|C_0| = O(d)$, which is sufficient because $|G| \leq |G'_0| + |C_0|$ and $d_0 \leq d$.

First, we show that the size of G_0 is $O(d_0^2)$. By the recursive construction (see Figure 5), G_0 is a disjoint union of some base graph B_q and b converters $C_1, C_2, \dots, C_b$, for $b = O(d_0)$. (Recall that the number of converters b is roughly $\frac{1}{2}d_0$.) Thus, $|G_0| \leq |B_q| + \sum_s |C_s|$. As $|B_q| = O(1)$, we can ignore the contribution of B_q to G_0 's size, and we focus on $\sum_s |C_s|$. So consider some C_s , with output set $I'_s = \{i : l, j : r\}$ of concentrations. This C_s is one of the $C_{i,j}^k$ converters constructed in Section 4. Each such $C_{i,j}^k$ is obtained by starting with an initial converter $C_{i',j'}^k$ and coupling it iteratively with $O(i+j)$ extenders. (Indeed, for each $k = 1, 2, 3$, the coefficients ϕ and ψ in Section 4 that represent the numbers of added converters satisfy $\phi + \psi = O(i+j)$.) As initial converters and extenders each have size $O(1)$, we obtain that $|C_s| = O(i+j) = O(|I'_s|)$. But the number of outputs $|I'_s|$ of C_s cannot exceed the total number of sinks in the whole graph G_0 , so, by Lemma 6.2, we have $|I'_s| \leq \frac{1}{2}d_0 + 3 = O(d_0)$. Thus, $|C_s| = O(d_0)$. Since also $b = O(d_0)$, we obtain that $|G_0| = O(bd_0) = O(d_0^2)$, as claimed.

We next show that the size of C_0 is $O(d)$. As explained at the end of Section 3, C_0 is used to produce the subset $\{j : 1/2^{\gamma-\sigma}\}$ of input concentrations I'_0 of G'_0 . C_0 consists of $\gamma - \sigma$ parallel rounds, where in round z , we add j_z nodes. It has $j = j_{\gamma-\sigma}$ sinks, $j_0 = \lceil j/2^{\gamma-\sigma} \rceil$ inputs, and for each round $z = 0, 1, \dots, \gamma - \sigma - 1$, we have $j_z \leq \frac{1}{2}(j_{z+1} + 1)$. This means that the sequence $j = j_{\gamma-\sigma}, j_{\gamma-\sigma-1}, \dots, j_1, j_0$ consists of two parts, with j_z 's geometrically decreasing in the first part, and all j_z 's equal 1 in the second part. Bounding the first part by a constant multiple of the first term j gives us that $|C_0| = \sum_z j_z = O(j + \gamma)$. Now, $|I_0|$ can not exceed the number of sink nodes in G_0 , so we have $|I_0| = O(d_0)$; therefore, since $j \leq |I'_0| = |I_0|$, $d_0 \leq d$, and $\gamma \leq d$, we obtain that $|C_0| = O(d)$.

Running time. Finally, we claim that the running time of Algorithm RPRIS is $O(|G|)$. This is straightforward, as G is composed of a set of disjoint converters and the algorithm constructs each converter incrementally by adding either a single node or an $O(1)$ -size component (initial converter or extender) in each step. For example, to construct $C_{i,j}^2$, the algorithm will compute $\phi = \lceil i/2 \rceil - 1$, $\psi = \lceil (j + \phi)/2 \rceil - 1$, and $(i', j') = (i, j) - \phi(0, 2) - \psi(2, -1)$, construct $C_{i',j'}^2$, and then append to it ϕ copies of X_1^2 and ψ copies of X_2^2 . All graphs $C_{i',j'}^2$, X_1^2 and X_2^2 are of size $O(1)$, so each individual step takes time $O(1)$. Also, $\phi + \psi = O(i+j) = O(|C_{i,j}^2|)$, so constructing $C_{i,j}^2$ takes time

$O(|C_{i,j}^2|)$. Thus, adding these times for all converters that comprise G , the overall running time will be $O(|G|)$, as claimed.

7 EXPERIMENTAL STUDY

In this section, we compare the performance of our algorithm with algorithms Min-Mix, REMIA, DMRW, GORMA, and ILP. We start with brief descriptions of these algorithms to give the reader some intuitions behind different approaches for constructing mixing graphs. Let $t \in (0, 1)$ be the target concentration and $d = \text{prec}(t)$ its precision. Also, let $\text{bin}(t)$ be t 's binary representation with no trailing zeros.

Min-Mix [12]: This algorithm is very simple. It starts with $\tau = 0$ and mixes it with the bits of $\text{bin}(t)$ in reverse order, ending with $\tau = t$. It runs in time $O(d)$ and produces d droplets of waste.

REMIA [7]: This algorithm consists of two phases. In the first phase, the algorithm computes a mixing graph G' whose source nodes have concentration labels that have exactly one bit 1 in their binary representation; these bits correspond to the 1's in $\text{bin}(t)$. Then, in the second phase, a mixing graph G'' , whose sink nodes are basically a superset of the source nodes in G' , is computed. The final mixing graph G for t is obtained as $G' \bullet G''$. (Although REMIA targets reactant usage, its comparison to different algorithms in terms of total waste was also reported in Ref. [7]. Thus, for the sake of completeness, we included REMIA in our study.)

DMRW [10]: This algorithm is based on binary search. Starting with pivot values $l = 0$ and $r = 1$, the algorithm repeatedly “mixes” l and r and resets one of them to their average $\frac{1}{2}(l + r)$, maintaining the invariant that $t \in [l, r]$. After d steps, we end up with $l = r = t$. Then, the algorithm gradually backtracks to determine, for each intermediate pivot value, how many times this value was used in mixing; based on this information, it computes the required number of droplets. This information is then converted into a mixing graph.

GORMA [3]: This algorithm enumerates a number of mixing graphs for a given target concentration. An initial mixing graph is constructed in a top-down manner; starting from the target concentration t (the root node), the algorithm computes two concentrations x and y (called a preceding pair) such that $t = \frac{1}{2}(x + y)$, and both x and y have smaller precision than t . These concentrations x and y become t 's children and both x and y are then processed recursively. (Note that a concentration might have many distinct preceding pairs. Each preceding pair is processed.) A droplet sharing process is then applied to every enumerated mixing graph to decrease reactant usage and waste produced. A branch-and-bound approach is adopted to reduce the (exponential) running time of the algorithm.

ILP [4]: This algorithm constructs a “universal” mixing graph that contains all mixing graphs of depth d as subgraphs. It then formulates the problem of computing a mixing graph minimizing waste as an integer linear program (a restricted flow problem), and solves this program. This universal graph has size exponential in d ; thus, the overall running time is doubly exponential in d .

We now present the results of our experiments. Each experiment consisted of generating all concentration values with precision d , for $d \in \{7, 8, 15, 20\}$, and comparing the outputs of each of the algorithms. The results for GORMA and ILP are shown only for $d \in \{7, 8\}$, since for $d \in \{15, 20\}$, the running time of both GORMA and ILP is prohibitive.

Fig. 14. The number of waste droplets of algorithms Min-Mix, REMIA, DMRW, GORMA, ILP, and our algorithm RPRIS, for all concentrations with precision 7 (top figure) and 8 (bottom figure). For better visual clarity, all graphs are smoothed using MATLAB's *smooth* function.

Figure 14 illustrates the experiments for concentrations of precision 7 and 8. Figure 15 illustrates the experiments for concentrations of precision 15 and 20. In both figures, the data was smoothed using MATLAB's *smooth* function to reduce clutter and to bring out the differences in performance between different algorithms.

As can be seen from these graphs, RPRIS significantly outperforms algorithm Min-Mix, REMIA, DMRW and GORMA: It produces on average about 50% less waste than Min-Mix (consistently with our bound of $\approx \frac{1}{2}(d + \gamma)$ on waste produced by RPRIS), and 40% less waste than REMIA. It also produces on average between 21–25% less waste than DMRW, with this percentage increasing with d . Additionally, for $d \in \{7, 8\}$, RPRIS produces on average about 17% less waste than GORMA and only about 7% more waste than ILP.

Among all of the target concentration values used in our experiments, there is not a single case where RPRIS is worse than either Min-Mix or REMIA. When compared to DMRW, RPRIS never produces more waste for precision 7 and 8. For precision 15, the percentage of concentrations where RPRIS produces more waste than DMRW is below 2%, and for precision 20, it is below 3.5%. Finally, when compared to GORMA, the percentage of concentrations where RPRIS produces more waste is below 4%.

Fig. 15. The number of waste droplets of algorithms Min-Mix, DMRW, REMIA, and our algorithm RPRIS, for all concentrations with precision 15 (top figure) and 20 (bottom figure). For better visual clarity, all graphs are smoothed using MATLAB's *smooth* function.

Other performance measures. Our work focuses strictly on minimizing waste. But it may also be interesting to see how our algorithm compares to the state-of-the-art in terms of other performance measures. The number of mixing operations was analyzed in Section 6, where we show that it can be bounded by $O(d^2)$. This, in fact, is a tight worst-case bound, as it can be shown that for some concentrations with precision d , our algorithm will produce a mixing graph with $\Omega(d^2)$ micro-mixers. Not surprisingly, this is asymptotically worse than Algorithm Min-Mix that generates a mixing graph of optimal size d . To test the performance in practice, we run the experiment where we computed the average generated graph size for all concentrations of precision $d = 15$. Obviously, the average size for Algorithm Min-Mix is 15. Our Algorithm RPRIS produced graphs of average size 30. The corresponding values for REMIA and DMRW were 18 and 27, respectively. (We did not test algorithms ILP and GORMA, since they are too slow for concentrations with precision 15.) There appears to be a tradeoff between waste and the number of mixing operations, and we discuss this tradeoff in Section 8. However, our experimental results indicate that, in spite of this tradeoff, for most concentrations, the graphs produced by Algorithm RPRIS are of reasonable size.

In the same experiment, we also did a comparison of the reactant usage. For concentration $d = 15$, the average reactant usage for Algorithm RPRIS was 4, while for Min-Mix, REMIA, and

DMRW, the average values were 8, 3, and 5, respectively. REMIA is designed specifically to minimize reactant usage, so its better performance is expected.

8 FINAL COMMENTS

In this article, we proposed Algorithm RPRIS for efficiently designing mixing graphs for generating a single droplet with a specified target concentration, and we showed that our algorithm outperforms standard waste minimization algorithms Min-Mix and DMRW in experimental comparison. We also proved that its worst-case bound on waste is significantly better than for the other two algorithms.

Many questions about mixing graphs remain open. We suspect that our bound on waste can be significantly improved. It is not clear whether waste linear in d is needed for concentrations not too close to 0 or 1, say, in the interval $[\frac{1}{4}, \frac{3}{4}]$. In fact, we are not aware of even a *super-constant* (in terms of d) lower bound on waste for concentrations in this range.

One direction that may have promise for improving the bound on waste is to exploit further our RPR technique. (The approach we outline here applies only to very large values of d , so it's mostly of theoretical interest.) To convey the main idea, let's focus on concentrations of precision d in the interval $[\frac{1}{4}, \frac{3}{4}]$, where the inevitable waste $\gamma = 0$. As explained in Section 3, in each recursive step, RPR reduces the precision by 2 while increasing the waste by 1. This 2 : 1 proportion guarantees the total produced waste for such concentrations to be $\approx \frac{1}{2}d$. A more general approach would be to reduce, in each step, the precision by some δ while increasing the waste by some ω , yielding a bound of $(\omega/\delta)d$ on the total waste. This would require the existence of appropriate converters, and we don't believe that such converters exist for $\omega = 1$ and $\delta = 3$; thus, the next possible values could be $\omega = 2$ and $\delta = 5$, which would give the total waste bound of $\frac{2}{5}d$. The required converters will be quite large, complex, and many, so that argument would likely involve a computer-assisted proof or an existence proof that circumvents the need for an explicit construction.

For single-droplet targets, it is not known whether minimum-waste mixing graphs can be effectively computed. The most fascinating open question, in our view, is whether it is decidable to determine if a given multiple-droplet target set can be produced without any waste. (As mentioned in Section 1, the ILP-based algorithm from Ref. [4] does not always produce an optimum solution.)

Another interesting problem is about designing mixing graphs for producing multiple droplets of the same concentration. Using perfect-mixing graphs from Ref. [6], it is not difficult to prove that if the number of droplets exceeds a certain threshold, then such target sets can be produced with at most one waste droplet. However, this threshold value is very large and the resulting algorithm very complicated. Such target sets are of practical significance, so a simple algorithm with good performance would be of interest.

Our focus is on minimizing the total waste, but, as mentioned in the introduction, in some applications, other performance measures for mixing graphs could be more appropriate. One such measure is the reactant usage. The fundamental idea of our RPR approach may be applicable to reactant minimization, but formalizing this approach would require better understanding of the bounds on reactant waste, namely defining the analogue of evitable waste. Then, each step of the process would reduce the "evitable reactant waste" value by some amount δ while producing ω actual reactant waste, where ω/δ is as small as possible. (In this case, δ and ω may not be integers.)

Another well-studied performance measure is the number of mixing operations, which is the size of the mixing graph. Experimental results from the literature (and Section 7) hint at the existence of tradeoffs between these three objective functions (total waste, reactant waste, graph size), namely that optimizing one may inevitably produce graphs that are sub-optimal with respect to the other two. To our knowledge, such tradeoffs have not been yet provably established. What we have in mind here are questions of the following form: Is there a concentration c of precision d

such that any mixing graph for c with total waste at most $w(d)$ must have size at least $s(d)$? For example, if $w(d) = d/2$ and $s(d) = 2d$, this would imply a waste vs. graph size tradeoff, because we know that such c has a mixing graph with d mixers and total waste d (it can be produced by Algorithm Min-Mix from Ref. [12]).

Leveraging the ideas from Ref. [6], for multi-droplet target sets, we can establish a tradeoff between the total waste and the depth of the mixing graph, which is yet another performance measure. The argument is this: consider the target set T with one droplet of concentration 2^{-d} and $(d-1)2^d + 1$ droplets of concentration $1 - 2^{-d}$. As shown in Ref. [6], this target set has a mixing graph of depth $2d$ and no waste. But it is easy to show, just by considering the droplet with concentration 2^{-d} , that a graph of depth d that produces T must have waste at least d . (With a bit more work, an even better bound on waste can be shown.)

ACKNOWLEDGMENTS

We are grateful to the anonymous reviewers of the earlier versions of our paper for multiple insightful suggestions that helped us to improve the clarity and rigor.

REFERENCES

- [1] Sukanta Bhattacharjee, Bhargab B. Bhattacharya, and Krishnendu Chakrabarty. 2019. *Algorithms for Sample Preparation with Microfluidic Lab-on-Chip*. River Publishers.
- [2] Bhargab B. Bhattacharya, Sudip Roy, and Sukanta Bhattacharjee. 2014. Algorithmic challenges in digital microfluidic biochips: Protocols, design, and test. In *International Conference on Applied Algorithms (ICAA'14)*. 1–16.
- [3] Ting-Wei Chiang, Chia-Hung Liu, and Juinn-Dar Huang. 2013. Graph-based optimal reactant minimization for sample preparation on digital microfluidic biochips. In *2013 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*. IEEE, 1–4.
- [4] Trung Anh Dinh, Shigeru Yamashita, and Tsung-Yi Ho. 2014. A network-flow-based optimal sample preparation algorithm for digital microfluidic biochips. In *19th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 225–230.
- [5] Kang-Yi Fan, Shigeru Yamashita, and Juinn-Dar Huang. 2019. Reactant minimization for multi-target sample preparation on digital microfluidic biochips using network flow models. In *2019 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*. 1–4.
- [6] Miguel Coviello Gonzalez and Marek Chrobak. 2019. Towards a theory of mixing graphs: A characterization of perfect mixability. In *International Conference on Algorithms and Complexity*. Springer, 187–198.
- [7] Juinn-Dar Huang, Chia-Hung Liu, and Ting-Wei Chiang. 2012. Reactant minimization during sample preparation on digital microfluidic biochips using skewed mixing trees. In *International Conference on Computer-Aided Design*. ACM, 377–383.
- [8] Juinn-Dar Huang, Chia-Hung Liu, and Huei-Shan Lin. 2013. Reactant and waste minimization in multitarget sample preparation on digital microfluidic biochips. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 32, 10 (2013), 1484–1494.
- [9] Debasis Mitra, Sudip Roy, Krishnendu Chakrabarty, and Bhargab B. Bhattacharya. 2012. On-chip sample preparation with multiple dilutions using digital microfluidics. In *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 314–319.
- [10] Sudip Roy, Bhargab B. Bhattacharya, and Krishnendu Chakrabarty. 2010. Optimization of dilution and mixing of biochemical samples using digital microfluidic biochips. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 29, 11 (2010), 1696–1708.
- [11] Sudip Roy, Partha P. Chakrabarti, Krishnendu Chakrabarty, and Bhargab B. Bhattacharya. 2015. Waste-aware single-target dilution of a biochemical fluid using digital microfluidic biochips. *Integration, the VLSI Journal* 51 (2015), 194–207.
- [12] William Thies, John Paul Urbanski, Todd Thorsen, and Saman Amarasinghe. 2008. Abstraction layers for scalable microfluidic biocomputing. *Natural Computing* 7, 2 (2008), 255–275.

Received January 2020; revised May 2020; accepted June 2020