OPTIMAL SCHEDULING AND MINIMUM RESOURCE CHARACTERIZATION OF BIOCHEMICAL ANALYSES ON DIGITAL MICROFLUIDIC SYSTEMS

By

Lingzhi Luo

A Thesis Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

Major Subject: COMPUTER SCIENCE

Approved:

_____

Srinivas Akella, Thesis Adviser

Rensselaer Polytechnic Institute
Troy, New York

July 2008
(For Graduation August 2008)

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENT

This work was supported in part by National Science Foundation under Award Nos. IIS-0713517, IIS-0730817, and CNS-0709099.

I would like to thank my adviser Srinivas Akella for his support and guidance that helped me through my graduate studies and along with his belief and confidence in my abilities. I have an immense amount of gratitude for the knowledge I gained from his council and guidance. I feel fortunate to have Srinivas as my adviser.

I want to express my gratitude to Franklin Luk, Mark Goldberg, Boleslaw Szymanski, Elliot Anshelevich, David Musser, and Sanmay Das for teaching and preparing me the knowledge for my research. In addition, I would like to thank John Wen, Jeff Trinkle, and Volkan Isler for their valuable comments on my presentations.

I would like to thank Eric Griffith, Nilanjan Chakraborty and Megha Gupta for their help throughout my study in RPI whether it was a class project or research. I enjoyed our lengthy discussions where their enthusiasm helped me spawn many new ideas and insights in both my research and my life. I would also like to thank Evan Shechter, Steve Berard, Eric Meisner, Nikhil Karnad, Onur Tekdas, Binh Nguyen, Ben Roghani and Wei Yang for being the best lab mates I can imagine having had for the last two years.

I appreciate the faculty and staff in the Computer Science Department at Rensselaer Polytechnic Institute who offered me a great study and research environment. I would also like to thank my fellow students for their help in preparing for the qualifiers.

A special thanks to my mother for her belief and support of me in whatever I choose to pursue. I am grateful for her patience and understanding.

ABSTRACT

Digital microfluidic systems (DMFS) are an emerging class of lab-on-a-chip systems that manipulate individual droplets of chemicals on a planar array of electrodes. The biochemical analyses are performed by repeatedly moving, mixing, and splitting droplets on the electrodes. This thesis presents algorithms for optimized operations and design of DMFS biochips.

This thesis focuses on two issues: minimizing the completion time of biochemical analyses by exploiting the parallelism among the operations, and identifying the minimum resource requirements of biochemical analyses, towards the design of cost and space-efficient biochips. ***Minimizing the completion time***: We find the lower bound of the mixing completion time according to the tree structure of input analyses, and calculate the minimum number of mixers M_{min} required to achieve the lower bound. We present a scheduling algorithm for the case with a specified number of mixers no greater than M_{min}, and prove it is optimal to minimize the mixing completion time. These are the first results that use the analysis tree structure for optimal scheduling design of biochemical analyses on DMFS. ***Minimizing the resource requirements***: We focus on determining the minimum resources based on the tree structure of the chemical analysis and use it to design (or select) the smallest chip for a given analysis. We present an algorithm to compute, for a given number of mixers, the minimum number of storage units for an input analysis using its tree structure, and design a corresponding scheduling algorithm to perform the analysis. We define the M-depth of the analysis tree to be the minimum number of storage units with M mixers. We characterize the variation of the M-depth of a tree with M, and use it to calculate the minimum total *size* (the number of electrodes) of mixers and storage units. We prove that we can always construct the smallest chip for an arbitrary analysis using one mixer and $f(1)$ storage units where $f(1)$ is the 1-depth of the biochemical analysis tree. These are the first results on the least resource requirements of DMFS for biochemical analyses, and can be used for the design and selection of chips for arbitrary biochemical analyses.

1. Introduction

1.1 Introduction

Low-cost, portable lab-on-a-chip systems capable of rapid automated biochemical analysis can impact a wide variety of applications including biological research, genetic analysis, point-of-care diagnostics, and biochemical sensing [1–4]. *Digital microfluidic systems* (DMFS) are an emerging class of lab-on-a-chip systems that manipulate discrete droplets [5, 6]. In contrast to conventional microfluidics, these systems offer a number of advantages, including reduced reagent requirements, size reduction, power reduction, increased throughput, and increased reliability. A digital microfluidic system manipulates individual droplets of chemicals on a planar array of electrodes by using electrowetting (or dielectrophoresis). The chemical analysis is performed by repeatedly moving, mixing, and splitting droplets on the electrodes.

An important advantage of DMFS devices is their reconfigurability and flexibility in performing various biochemical analyses. We focus on microfluidic systems that manipulate droplets by electrowetting [7]. Droplets are nanoliters in volume, and have been moved at 12-25 cm/s on planar arrays of 0.15 cm wide electrodes [5,8]. The ability to control discrete droplets on a planar array enables complex analysis operations to be performed in DMFS devices (Fig. 1.1). For simple biochemical analysis operations, no special purpose devices are required aside from the array itself. Mixers and storage units are two important resources consisting of electrodes. Mixers are used to perform mixing and splitting operations, while storage units are used to store droplets which have been produced for future mixings (Fig. 1.2). The array may additionally contain cells that can perform specialized operations, such as heating or optical sensing.

This thesis focuses on two issues: 1. Minimizing the completion time of biochemical analyses by exploiting the parallelism among the operations, and 2. Determining the minimum resources based on the tree structure of the chemical analysis and using it to design (or select) the smallest chip for a given analysis.

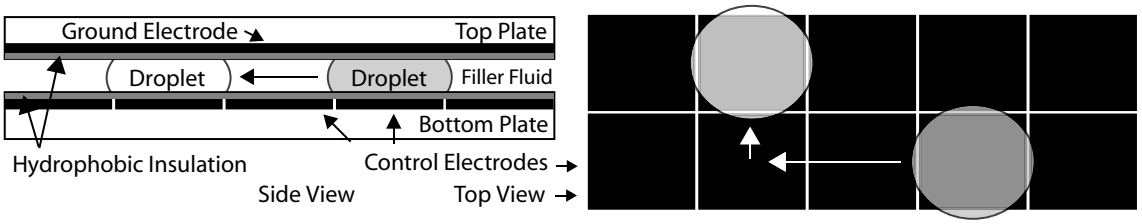
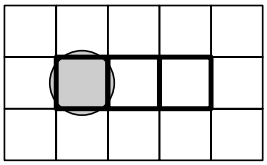


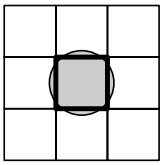
Figure 1.1: Droplets on an electrowetting array (side and top views). The droplets are in a medium (usually oil or air) between two glass plates. The gray and white droplets represent the same droplet in initial and destination positions. A droplet moves to a neighboring electrode when that electrode is activated; the electrode is turned off when the droplet has completed its motion. Based on [6].

1.2 Minimizing the Completion Time

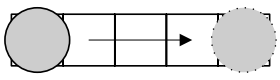
The completion time of biochemical analyses in batch mode is the time required to produce one droplet of final product. Our goal is to minimize the completion time in batch mode under resource constraints on DMFS. The completion time of a reaction depends on the reaction, the provided resources, and the algorithm to perform it. Here we exploit the tree structure of biochemical analysis to do scheduling and



(a) a mixer with 3 electrodes for droplets to move



(b) a storage unit with 1 electrode for droplets to stay



(c) transportation path for droplets

Figure 1.2: A schematic of a mixer, a storage unit and transportation paths.

minimize the completion time. A full binary tree structure is introduced to model the biochemical reactions we consider, and is clear and convenient for algorithm design and analysis. First, using pipelining we overlap the mixing operations with input and transportation operations and give the quantitative (other than just conceptional) scheduling order of different operations. Then we focus on scheduling of mixing operations to minimize the mixing completion time. We calculate the lower bound of a given reaction based on the tree structure. Also we calculate the minimum number of mixers required to achieve minimum completion time, and present a greedy scheduling algorithm to minimize the completion time under mixer constraints. With the above algorithm, parallelism is fully exploited not only among different mixing steps but also among different kinds of operations. Also, we consider two extreme cases: with just one mixer and with zero storage units. In the first case, we prove all active scheduling (e.g. keep the mixer busy at all times) result in the same completion time, calculate the minimum number of storage units for the given analysis, and design a corresponding scheduling algorithm. In the second case we compute the minimum number of mixers required. Considering these two extreme cases we start exploring the relationship between the number of mixers and the number of storage units and this result can be used to guide how to select the chip selection or design layout with proper number of mixers and storage units. Also we provide theoretical guarantee for the results of designed algorithms.

For each section of mixer scheduling, we first analyze and design an algorithm assuming identical mixing duration and then generalize to the case with different mixing durations. When discussing the optimal scheduling with given number of mixers, we first design and analyze two algorithms which can sometimes achieve the optimal solution but are not guaranteed. Then we reduce the problem with different mixing durations to an equivalent problem with identical mixing duration based on preemption.

1.3 Minimizing the Resource Requirements

There are several classes of resources (i.e., functional components consisting of electrodes) in a DMFS: mixers, storage units, input and output units, and trans-

portation paths. Figure 1.3 illustrates their functions. We focus on performing a biochemical analysis in batch mode, where the goal is to produce one droplet of the final product.

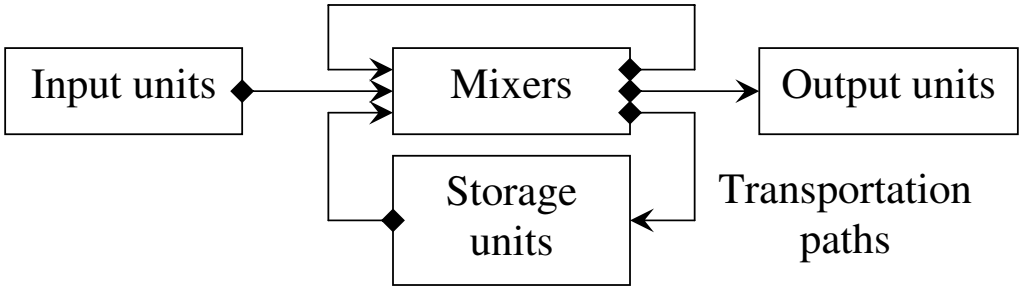


Figure 1.3: Five classes of functional resources for DMFS biochips. Transportation paths (shown as arrows starting from diamonds) are used to move droplets from one resource to another. Input units are used to input droplets. Mixers are used to mix two different droplets and split their mixture to produce two new droplets. In batch mode, one of the new droplets will be used and the other will be discarded as a waste droplet at an output unit. The new droplet may be kept idle in a storage unit for some time, or it can stay at the mixer, or be transported to a new mixer for a subsequent mixing. The droplet of final product will be transported to an output unit and collected there.

Here we explore the structure of different biochemical analyses and classify them according to their least resource requirements. We assume that the resources (e.g., mixers and storage units) are fixed at the start of the analysis and do not change over the course of the analysis. This work is motivated by the desire to answer the following practical questions: For any biochemical analysis, how do we identify a biochip with sufficient number of resources to perform it? For a given biochip, what kind of biochemical analyses can be performed on it? First, we compute the least resource requirements based on the tree structure of biochemical analyses and design corresponding algorithms to perform them. Then we define the M-depth of a tree to describe such resource requirements and characterize the variation of the M-depth with the number of mixers M. These results can be used to answer the above questions and also to design the smallest chip for any given biochemical analysis. The smaller the number of electrodes in a DMFS chip, the easier the fabrication and the lower the cost. This is the first work on DMFS to compute the least resource

requirements for biochemical analyses and it can be used to guide the selection of chips for arbitrary biochemical analysis.

1.4 Literature Review

Almost all current lab-on-a-chip systems use continuous flow microfluidics (for example, [1]). In addition to needing moving parts such as microvalves and micropumps, such devices can perform only one (or a very small number of) prespecified chemical analyses since fluid flow occurs in fixed channels and are therefore not reconfigurable. Consequently, these devices lack the ability to change analyses in response to intermediate analysis results.

Digital Microfluidic Systems: There are two primary methods of actuation in a DMFS: electrowetting [9], where the surface tension of droplets is modulated by a voltage, and dielectrophoresis [10], where a nonuniform electric field causes droplet motion. mixing strategies on the rate of droplet mixing. Gong, Fan, and Kim [11] developed a portable digital microfluidics lab-on-chip platform using electrowetting on dielectrics. They use a time-multiplexed control scheme to control droplets with limited row-column addressing [12]. Gong and Kim [13] recently developed a directly addressable 2D digital microfluidic system by using multi-layer printed circuit board technology. Jones et al. [14] demonstrated dielectrophoresis based liquid actuation and nanodroplet formation. Gascoyne et al. [15] developed a dielectrophoresis-based chip to move droplets and cells, and Manaresi et al. [16] have developed a developed a dielectrophoretic chip capable of manipulating over 10,000 living cells. Recent DMFS research has also focused on applications. Srinivasan et al. [17] demonstrate the use of a DMFS as a biosensor for glucose, lactate, glutamate and pyruvate assays, and for clinical diagnostics on human whole blood, plasma, serum, urine, saliva, sweat, and tears [18]. Pollack et al. [19] demonstrated the use of electrowetting-based microfluidics for real-time polymerase chain reaction (PCR) applications. Wheeler et al. [20] demonstrate an electrowetting-based DMFS for high throughput analysis of proteins by matrix-assisted laser desorption/ionization mass spectrometry. Detailed physical modeling of droplet motion is an active area of research [21, 22]; however this is not a focus of this thesis since we

are interested in higher-level droplet coordination and scheduling issues.

Despite these rapid advances in digital microfluidic hardware technologies, there is a lack of algorithms for automated coordination and control of droplets on DMFS chips. Some issues were explored as follows.

Resource Requirements: Su and Chakrabarty presented architecture-level synthesis and geometry synthesis of biochips [23, 24]. They used acyclic sequence graphs to represent the reactions and developed techniques in operation scheduling, resource binding, and module placement. In their papers resource constraints are given in advance by the size of chips or the number of mixers and storage units. However there is no general guideline for selecting chips with proper number of mixers and storage units for biochemical analyses. Griffith and Akella [25] explored the relationship between the number of mixing units and the highest stable input rates in the system. By simulations, it was found that increasing the number of mixing units permits the system to be stable at a higher input rate and the effectiveness of maintaining system stability by increasing the number of mixers decreases. They experimented with a variety of modifications to the resources to gauge the effects on the stability of the system [26]. The work in [25,26] is based on simulations while in this paper we are formally analyzing the resource requirements for analyses based on the underlying tree structure.

Layout Design: Layout design maps the functional units such as mixers, storage units, and routing paths to the underlying hardware. Griffith and Akella presented a semi-automated method to generate the array layout in terms of components [25]. Su and Chakrabarty developed an online reconfigurable technique to bypass the fault unit cells in the microfluidic biochips [27].

Scheduling Algorithms: Scheduling algorithms optimize the system performance by properly allocating tasks to resource. Kwok and Ahmad studied the static scheduling of a program on a multiprocessor system to minimize the program completion time in parallel processing [28]. Since the general problem is NP-complete, they compared 27 heuristic scheduling algorithms. In contrast to the general problem in parallel computing, the multiple-task reactions in DMFS have certain kinds of precedence, which can be represented by a full binary tree. Ding et al. [29] and Su

and Chakrabarty [23] represent the DMFS reactions using data-flow directed graph and consider the scheduling using Integer Linear Programming (ILP). They solve the problem by general ILP solvers and heuristic algorithms without exploiting the structure of DMFS reactions.

Routing: Böhringer modeled the routing problem in DMFS as a multi-robot cooperation problem and used a prioritized A^* search algorithm to generate the optimal plan for droplets [30]. Griffith and Akella presented a general-purpose DMFS and designed routing algorithm based on Dijkstra's algorithm [25, 26]. Su, Hwang and Chakrabarty proposed a two-stage routing method to minimize the number of electrodes used for droplet routing while considering the resource constraint [31]. Gupta and Akella presented an algorithm for coordinating droplet movement in batch mode operations on ring layouts with bus-phase addressing. The algorithm is scalable to different number of reactions within a limit which depends on the size of the layout, placement of sources and number of phases used. [32].

1.5 Thesis Overview

The rest of the chapters are organized as follows. In chapter 2, we present algorithms to minimize the completion time of biochemical analyses by exploiting the parallelism among the operations. We also consider the resource constraint issue for two extreme cases of having just one mixer and having zero storage units. Then in chapter 3, we extend the resource constraint issues and focus on characterizing the resource requirements of arbitrary biochemical analyses on DMFS biochips from their tree structure. We determine the minimum resources based on the tree structure of the chemical analysis and use it to design (or select) the smallest chip for a given analysis. Chapter 4 concludes this thesis with a summary and outlines some future directions.

2. Minimizing the Completion Time

The completion time of biochemical analyses in batch mode is the time required to produce one droplet of final product. Our goal is to minimize the completion time in batch mode under resource constraints on DMFS. The completion time of a reaction depends on the reaction, the provided resources, and the algorithm to perform it. Here we exploit the tree structure of biochemical analyses to perform scheduling and minimize the completion time. A preliminary version of this work was presented in [33].

2.1 Binary Tree Model of Biochemical Analysis

The biochemical "analysis graph" provides a representation of the operations of DMFS. It is a directed graph, with an input node for each droplet type entering the system, an output node for each product droplet type leaving the system, and a mix node for each mixing operation performed in the system. The nodes are connected based on the droplet types they require and produce, and the edges represent transport operations. For example, consider the PCR analysis graph shown in Fig. 2.1. The mixing operations during the PCR analysis are described as follows.

$A + B \rightarrow C$; $C + D \rightarrow E$; $E + F \rightarrow G$;

$H + I \rightarrow J$; $J + K \rightarrow L$; $L + M \rightarrow N$; $N + G \rightarrow O$.

Here $A + B \rightarrow C$ means reagents A and B are mixed to produce C. To model the dependencies among different mixing operations, we introduce a full binary tree structure. A full binary tree is a binary tree in which every node is either a leaf node or it has two child nodes [34]. Given an analysis R, we construct a full binary tree T as follows. Every reagent in R is represented by a node in T: source reagents, which can be directly fetched from the reservoirs, are represented by leaf nodes in T, and intermediate reagents, which are produced by mixing, are represented by parent nodes of those nodes from which they can be produced. So the analysis R is represented by T, where the final product of R is represented by the root node of T. The representation of the PCR analysis of Fig. 2.1 by a full binary tree is shown

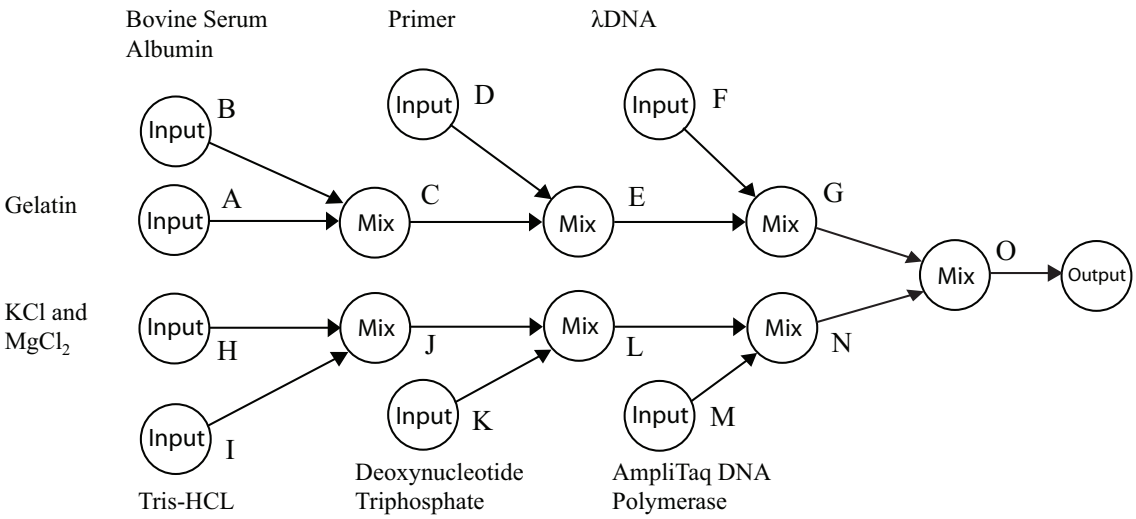


Figure 2.1: PCR analysis graph. Input nodes are labeled with the reagents they introduce and alphabet labels for convenience. Mix nodes represent the mixing operations. The output node represents the final product.

in Fig. 2.2.

2.2 Scheduling Algorithm Design

For a general biochemical analysis, there are five kinds of basic operations: *input*, *transport*, *mix* (including split), *store*, and *output*. The execution order of these operations should satisfy the following rules:

1. *Operation precedence rule*: The precedence order of related operations for a mixing $A + B \rightarrow C$ is: *Input* A and $B >$ *Transport* A and B to a mixer $>$ *Mix* A and B for $C >$ *Output*/*Store*/*Transport* C.

2. *Mixing precedence rule*: In the binary tree model, the mixing operation for a parent node should occur later than the mixing operation for its child nodes.

3. *Resource constraint rule*: At any time the utilized resources (e.g. mixers and storage units) should not exceed the resources of the biochip.

Our scheduling design is composed of two parts: pipelining and mixer scheduling. Pipelining overlaps mixing operations with other operations while satisfying

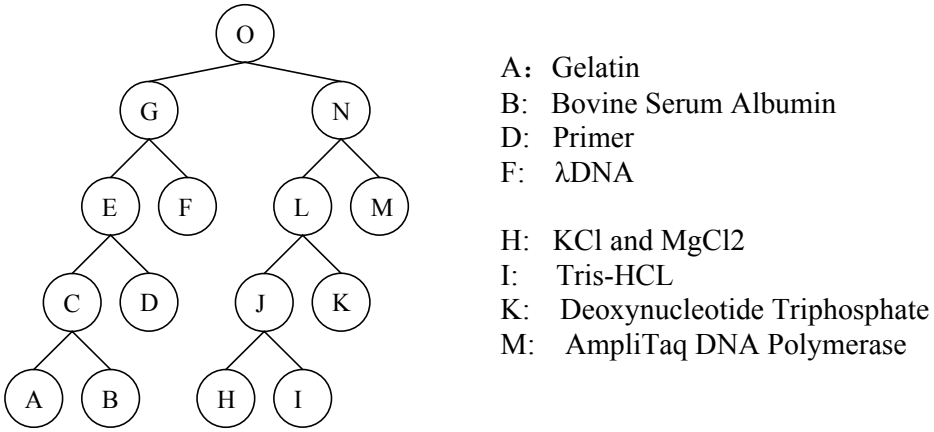


Figure 2.2: Representation of PCR analysis by a full binary tree with depth 4.

the operation precedence rule. Scheduling of mixing operations reduces the total mixing time while satisfying the mixing precedence rule. Both pipelining and mix scheduling should satisfy the resource constraint rule.

2.2.1 Pipelining

Pipelining is a scheduling method that divides a task into several subtasks and performs different subtasks of multiple tasks in different function units simultaneously to reduce the overall completion time of all tasks [35]. Using pipelining, the average completion time of a single task will depend on the subtask that takes the longest time.

For biochemical analyses on a DMFS, the procedure of obtaining a product droplet can be divided into several subtasks: input source droplets, transport droplets, mix (and split) droplets, and output waste droplets. Among these subtasks, droplet mixing takes the longest time. The mixing duration depends on the mixer size, droplet motion pattern, and the chemicals to be mixed [7]. To simplify our analysis, we initially assume all mixing operations have an identical duration. Fig. 2.3 shows an example of applying pipelining to a part of PCR analysis.

Suppose $t_{completion}$ is the completion time, T_{input}, T_{output} and T_{mix} are the durations for one input, output, and mixing operation respectively, t_{mix}^{total} is the total

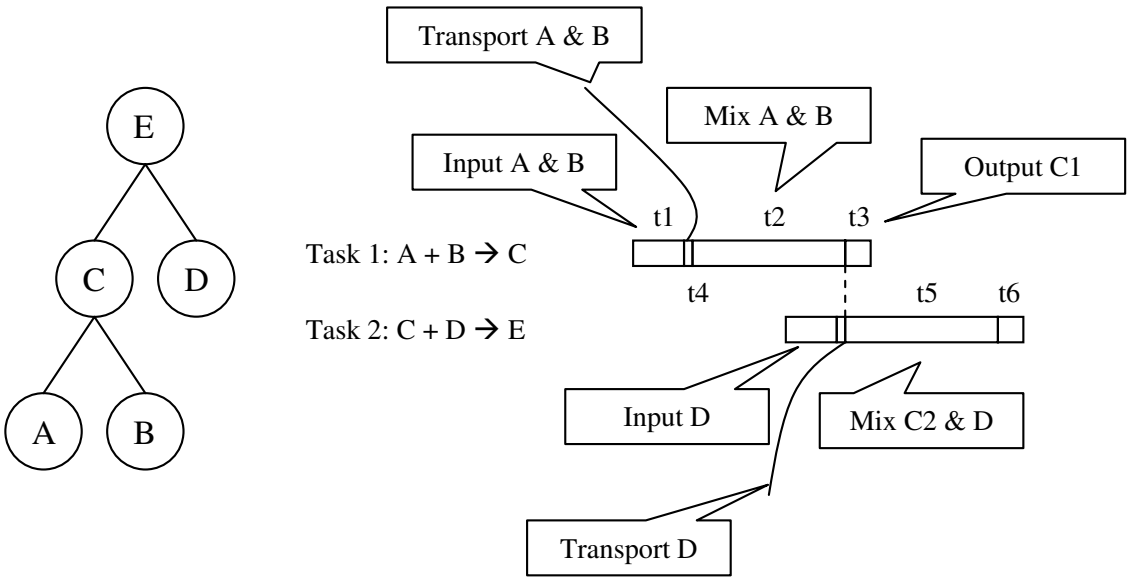


Figure 2.3: Application of pipelining to an analysis on DMFS. After mixing operations (including splitting), there will be two destination droplets produced, labeled with 1 and 2 respectively. The output operations are used to dispense one waste droplet (labeled with 1) outside the system. The second mixing task is also performed in the same mixer, so droplet C2 does not need to be transported for the second mixing task.

time for mixing, and $t_{transport}$ is the time for transportation operations.

$$t_{completion} = T_{input} + t_{mix}^{total} + t_{transport} + T_{output} \tag{2.1}$$

where t_{mix}^{total} equals $\sum T_{mix}$.

Also, since the transportation time is negligible compared to the mixing time [5,8], we ignore the transportation time in subsequent scheduling. The approximate completion time:

$$t_{completion} \approx T_{input} + t_{mix}^{total} + T_{output} \tag{2.2}$$

Since T_{input}, T_{output}, and T_{mix} are constants, we only consider optimization of mixing completion time t_{mix}^{total} below.

2.2.2 Scheduling Mixing Operations

The goal of mixer scheduling is to minimize t_{mix}^{total} using a given number of mixers. t_{mix}^{total} depends on three factors:

- The tree structure of the biochemical analysis,
- The number of mixers provided,
- The scheduling algorithm.

In the following sections, we will answer the questions below:

1. What is the lower bound of the mixing completion time for a biochemical analysis?
2. What is the minimum number of mixers to achieve the lower bound on the mixing completion time?
3. What is the optimal scheduling algorithm with a limited number of mixers for an arbitrary analysis?
4. What if we have only one mixer or zero storage units?

2.2.3 Lower Bound of Mixing Completion Time

2.2.3.1 Identical Mixing Durations

Suppose T is a full binary tree that represents the biochemical analysis R, the depth of T is K, and the number of nodes at level i is N_i, $i = 0, \ldots K$. (The root node of T is at level 0 and the deepest leaf nodes are at level K.) Here the mixing duration T_{mix} is defined as the duration from the time when the second reagent arrives at the mixer entrance to the time when the mixed product exits the mixer.

Lemma 1 *The lower bound $T_{lowerbound}$ of t_{mix}^{total} is the sum of mixing durations along the deepest branch of T.*

$$T_{lowerbound} = K \cdot T_{mix} \tag{2.3}$$

Proof. First, we show that all mixing operations can be completed in $K \cdot T_{mix}$ time with $\max_{i=1}^{K} \frac{N_i}{2}$ mixers. We perform the mixing operations from the deepest nodes to the highest nodes until the root node at level 0. Since the number of mixers $\max_{i=1}^{K} \frac{N_i}{2} \geq \frac{N_i}{2}$, we can schedule all the mixing operations on level i in a mixing duration T_{mix}. After the mixing operation on level 1, we get the root node. In this

case $t_{mix}^{total} = K \cdot T_{mix}$. So $T_{lowerbound} \leq K \cdot T_{mix}$.

Second, we show that $T_{lowerbound} \geq K \cdot T_{mix}$. Suppose we can perform all mixing operations in less time, that is, $T_{lowerbound} < K \cdot T_{mix}$. Then according to the *pigeonhole principle*, there must be at least two mixing operations along the deepest branch of the tree performed in the same time slot, which contradicts the mixing precedence rule.■

2.2.3.2 Extension to Different Mixing Durations

Let B_{dpst} denote the branch with the biggest sum of mixing time durations starting from the root node, b_i denotes the ith node along B_{dpst}, $T_{mix}^{b_i}$ denote the duration of the mixing operation to obtain b_i.

Lemma 2 *The lower bound $T_{lowerbound}$ of t_{mix}^{total} is the sum of mixing durations along the branch B_{dpst} with the biggest sum of mixing time durations.*

$$T_{lowerbound} = \sum_{b_i \in B_{dpst}} T_{mix}^{b_i} \tag{2.4}$$

Proof. The proof of this lemma is similar to the proof of Lemma 1. First, given sufficient number of mixers and storage units, mixing operations along B_{dpst} can be performed consecutively without delay from completing other branches. In that case, $t_{mix}^{total} = \sum_{b_i \in B_{dpst}} T_{mix}^{b_i}$. Second, considering the *mixing precedence rule*, the mixing operations along B_{dpst} cannot be performed in parallel. So $T_{lowerbound} = \sum_{b_i \in B_{dpst}} T_{mix}^{b_i}$.■

We design Algorithm 1 to compute $T_{lowerbound}$, and B_{dpst} can be visited by tracking the "*T.root.next*" pointers stored by the algorithm. Algorithm 1 is designed based on two steps. First, base step $T_{lowerbound} = 0$ for a leaf node. Second, induction step $T_{lb}^{T} = \max(T_{lb}^{T.left}, T_{lb}^{T.right}) + T_{mix}^{T.root}$. (Since the left and the right branches can be performed in paralel, the lower bound of completing both branches is the bigger one of their lower bounds. But due to the mixing precedence rule, the mixing operation to obtain $T.root$ should be performed after completing both left and right branches. So we can get the induction step.)

Algorithm 1 Lower-Bound

Input: T // The binary tree
Output: LB // Lower bound
if $T.root$ is a leaf node **then**
  $T.root.lb = 0$
  $T.root.next = null$
else
  $Left$=Lower-Bound($T.left$)
  $Right$=Lower-Bound($T.right$)
  $T.root.lb = \max(Left, Right) + T_{mix}^{T.root}$
  if $Left > Right$ **then**
    $T.root.next = T.left.root$
  else
    $T.root.next = T.right.root$
  end if
end if
return $T.root.lb$

2.2.4 Minimum Number of Mixers M_{min} to Achieve the Lower Bound of Mixing Completion Time

To compute the minimum number of mixers, we introduce a lemma first. If we remove the reagent nodes of performed mixing operations after each time slot, the scheduling of mixing operations can be represented by a sequence of full binary trees. Let $Tree_i$ denote the full binary tree after the ith mixing duration, T_n denote the number of mixing durations for the final product. So $Tree_0$ is the initial tree, and $Tree_{T_n}$ is the root node.

Lemma 3 *The number of mixing operations that can be performed for $Tree_i$, $i = 0, \ldots, T_n$, will change in non-increasing order.*

Proof. A mixing operation in $Tree_i$ can be performed only when both reagents for that mixing are leaf nodes. Let P denote the set of such mixing operations. Each mixing operation belongs to one of three cases, based on its parent mixing operation as shown in Fig. 2.4.

The second case will not change the size of P, while the first and third case will decrease the size of P by 1.■

Next we calculate the minimum number of mixers required to achieve the lower bound of mixing completion time. The sufficient and necessary condition is that the

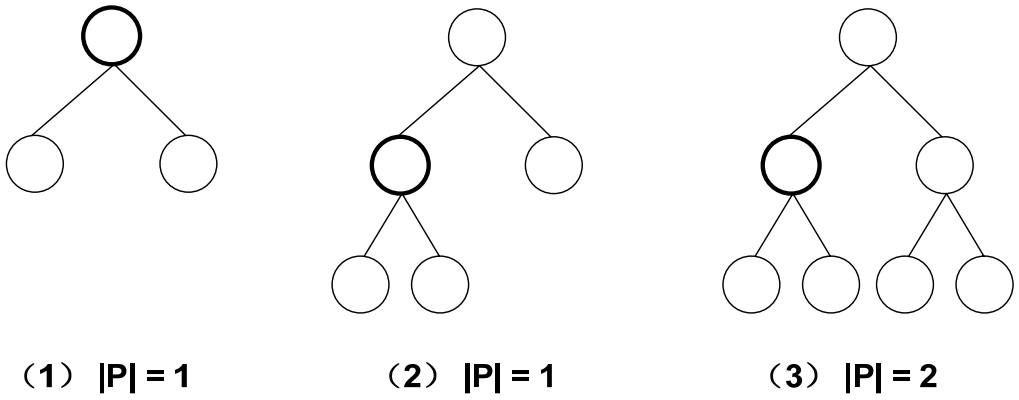


Figure 2.4: Classification of mixing operations. Case 1: It is not followed by subsequent mixing operations. Case 2: Its parent's other reagent is a leaf node. Case 3: Its parent's other reagent results from a mixing operation.

mixing operations along the deepest branch can be performed one by one in each mixing duration T_{mix}.

Let M_{min} denote the minimum number of mixers to achieve the lower bound of mixing completion time, (i, j) denote the jth node in the ith level, $M_{(i,j)}$ denote cumulative number of mixing operations from leaf nodes to (i, j), and N_i denote the number of nodes on the ith level.

Theorem 1 *To achieve the lower bound of mixing completion time, the sufficient and necessary condition is*

$$\sum_{j=1}^{N_i} M_{(i,j)} \leq M_{min} \cdot (K - i) \qquad \textit{for all } i \in \{0, 1, \cdots, K-1\} \tag{2.5}$$

where

$M_{(i,j)} =$

$$\begin{cases} 0 & \textit{if } (i,j) \textit{ is a leaf node;} \\ & \textit{if } (i+1, j_l) \textit{ and } (i+1, j_r) \\ M_{(i+1,j_l)} + M_{(i+1,j_r)} + 1 & \textit{are left and right child nodes} \\ & \textit{of } (i,j) \end{cases}$$

That is,

$$M_{min} = \max_{i=0}^{K-1} (\lceil \frac{\sum_{j=1}^{N_i} M_{(i,j)}}{K - i} \rceil) \tag{2.6}$$

Proof. When the deepest branch is processed up to the ith level, the required number of mixing operations is $\sum_{j=1}^{N_i} M_{(i,j)}$ and the number of mixing operations that can be performed by M_{min} mixers during the elapsed $K-i$ time slots is $M_{min} \cdot (K-i)$. So the necessary condition is that Equation 2.5 holds true.

Next, we show that Equation 2.5 is also the sufficient condition for $t_{mix}^{total} = T_{lowerbound}$ by proving that the number of mixers $S_z = \max_{i=z}^{K-1} (\lceil \frac{\sum_{j=1}^{N_i} M_{(i,j)}}{K-i} \rceil)$ can guarantee that nodes in level i can be produced before the end of time slot $K-i$ for all $i = K-1, \ldots, z$.

Base Step: $z = K-1$. $S_{K-1} = \sum_{j=1}^{N_{K-1}} M_{(K-1,j)}$ is the number of mixing operations to produce all nodes at level $K-1$. The conclusion holds true.

Induction Step: For some level v, suppose that all mixing operations for nodes in levels $l > v$ can be performed using $S_{v+1} = \max_{i=v+1}^{K-1} (\lceil \frac{\sum_{j=1}^{N_i} M_{(i,j)}}{K-i} \rceil)$ mixers. $S_v = \max_{i=v}^{K-1} (\lceil \frac{\sum_{j=1}^{N_i} M_{(i,j)}}{K-i} \rceil)$. $S_v \geq S_{v+1}$. So all mixing operations for nodes in levels $l > v$ can be performed using S_v mixers. If in some time slot $K-l$, there are less number of mixing operations than S_v that can be performed, according to Lemma 3, the situation will continue to time slot $K-v$. So all mixing operations for nodes in all levels will be performed. Otherwise, if in all time slots from 1 to $K-v$, there are enough mixing operations to be performed, all mixing operations can also be performed in this case since $S_v = \max_{i=v}^{K-1} (\lceil \frac{\sum_{j=1}^{N_i} M_{(i,j)}}{K-i} \rceil) \geq (\lceil \frac{\sum_{j=1}^{N_v} M_{(v,j)}}{K-v} \rceil)$.

So the number of mixers $S_z = \max_{i=z}^{K-1} (\lceil \frac{\sum_{j=1}^{N_i} M_{(i,j)}}{K-i} \rceil)$ can guarantee that the mixing operations for nodes in level i can be performed before the end of time slot $K-i$ for all $i = K-1, \ldots, z$. So the root node, which is in level 0, can be produced at the end of time slot K. So the lower bound of mixing completion time is achieved using M_{min} mixers. ■

Using Equation 2.6, we can directly compute the minimum number of mixers to achieve the lower bound of mixing completion time. However this result can only applied to the case with identical mixing duration. For the case with different mixing duration, we can use binary search for the number of mixers until the optimal mixer scheduling (see next section) achieves the lower bound of mixing completion time. But in that case we do not have an equation to directly compute the number.

2.3 Optimal Mixer Scheduling with Given Number of Mixers

2.3.1 Unique Mixing Durations

The mixer scheduling in Algorithm 2 is basically a greedy algorithm performing mixing operations from bottom to top (this scheduling is in fact the critical path (CP) rule for parallel scheduling [36]).

- First perform the mixing operations at the deepest level and then proceed upwards.
- At each stage, perform as many mixing operations as possible using the given number of mixers.

Algorithm 2 Optimal-Scheduling-with-M-Mixers

Input: T, M *// The binary tree and the number of mixers*
Output: F *// Schedule*
$i = 1$ // time slot index
$F = \emptyset$
while $T \neq$ a tree with just a root node **do**
  $C = \emptyset$ *// set of mixings that can be performed currently*
  Find pairs of leaf nodes with the same parent nodes in T *// identify mixing operations ready to be performed*
  Store them in C, ordered from deepest level upwards
  if $|C| < M$ **then**
    $F = F \bigcup \{$Schedule all mixing operations in C in the time slot $i\}$
  else
    $F = F \bigcup \{$Schedule the first M mixing operations of C in the time slot $i\}$
  end if
  Update T by removing leaf nodes involving the mixing operations just added to F
  $i = i + 1$
end while
return F

Theorem 2 *Algorithm 2 is the optimal scheduling algorithm with M mixers to minimize mixing completion time.*

Proof. Suppose algorithm O is optimal with M mixers to minimize mixing completion time and T_O, T_Q are the number of mixing durations of O and our algorithm, respectively.

First consider our algorithm described above. Let i be the time slot when our algorithm finally encounters $|C| > M$ and all operations in C are mixings of nodes belonging to the same level. Let d denote the depth of T at the beginning of time slot i. According to Lemma 3 and the bottom-to-top execution order of Algorithm 2, $|C| > M$ was always satisfied before the i^{th} time slot and no node at level higher than d was mixed. Also for every subsequent time slot, the algorithm will finish mixing operations of all nodes at subsequent level. That is, $T_Q = i + d$.

Suppose T_1 is the number of time slots spent mixing all nodes at level d or deeper by O, U is the number of mixing operations which need to be performed for nodes at level d or deeper, and T_2 is the number of time slots spent mixing all nodes at levels smaller than d by O. So $T_2 = T_O - T_1$.

At the end of time slot i, our algorithm performed fewer mixing operations than U using M mixers, so $U > M \cdot i$, $T_1 \geq U/M \geq i + 1$. According to Lemma 1, $T_2 \geq d - 1$. Thus, $T_O = T_1 + T_2 \geq i + d$.

So $T_O \geq T_Q$. That is, our algorithm is optimal with M mixers to minimize the mixing completion time. ■

2.3.2 Extension to Different Mixing Durations

We first consider extending the previous algorithm on the biochemical analysis tree, and design Algorithm 3. The basic idea is, when selecting the mixing operations from the remaining tree, we just implement the previous algorithm as if all mixing durations were the same; when performing mixing operations, we use the real mixing durations. Algorithm 3 is the direct extension of Algorithm 2 taking into account the different mixing durations. However, it is not the optimal algorithm. Consider the example in Figure 2.5. Algorithm 3 will select the mixing operation with the deepest level first as shown in Figure 2.6. But the optimal solution is shown in Figure 2.7, where the lower bound is achieved. Since the time durations are different, we do not use the concept of level for scheduling. Instead, we use the sum

Algorithm 3 Sched-Diff-Mixing-Durations-with-M-Mixers

Input: T, M *// The binary tree and the number of mixers*
Output: F *// Schedule*
$t = 0$ // present time
$F = \emptyset$
while $T \neq$ a tree with just a root node **do**
  $C = \emptyset$ *// set of mixings that can be performed currently*
  Find pairs of leaf nodes with the same parent nodes in T *// identify mixing operations ready to be performed*
  Store them in C, ordered from deepest level upwards
  if $t == 0$ **then**
    if $|C| < M$ **then**
      $F = F \bigcup$ {Schedule all mixing operations in C at time t}
    else
      $F = F \bigcup$ {Schedule the first M mixing operations of C at time t}
    end if
  end if
  Find the first produced node i
  $t = t + T^i_{mix}$
  Update T by removing leaf nodes involving the performed mixing operations
  if $C \neq \emptyset$ **then**
    $F = F \bigcup$ {Schedule the first mixing operation in C at time t}
  end if
end while
return F

of mixing durations (defined as length) along the branch from nodes to the root node.

The basic idea of Algorithm 4 is as follows:

1. First perform the mixing operations with the greatest length.

2. At each stage, perform as many mixing operations as possible using the given number of mixers.

The optimal solution shown in Figure 2.7 can be achieved using Algorithm 4. However, it is not the optimal algorithm either. Consider another example in Figure 2.8. Algorithm 4 will select the mixing operation with the greatest length first as shown in Figure 2.9. However the optimal solution which achieves the lower bound is shown in Figure 2.10.

So Algorithm 3 and Algorithm 4 can not guarantee the optimality of their

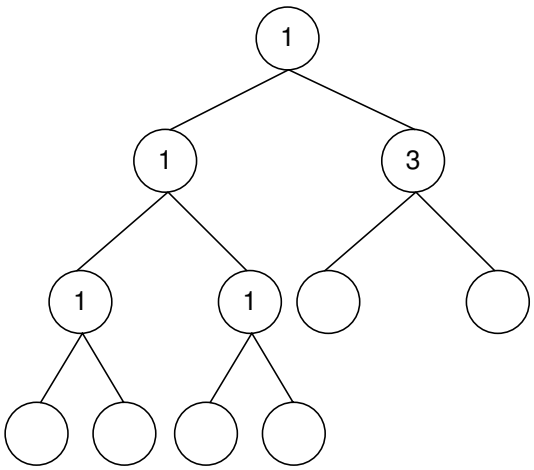


Figure 2.5: A counterexample to the optimality of Algorithm 3 with two mixers. The number inside non-leaf node i is the remaining mixing times to produce i(e.g. $T^i_{mix} - t_i$, where t_i is the time when a mixer was used to produce i). Note that there are no numbers inside leaf nodes since they have been produced.

solutions. Here we apply the concept of preemption [36] and thus get the optimal completion time schedule based on preemption. In the context of DMFS, preemptions imply that a mixing operation can be interrupted at any time and when a preempted mixing operation resumes, it can be completed in its remaining processing time. By using preemptions, problems with different mixing durations can be reduced to equivalent problems with the same mixing duration; the structure of trees will change accordingly. We can then use Algorithm 2 to get the optimal schedule for problems with different mixing durations.

First, suppose the mixing durations are integers and the scheduler can preempt any mixer at integer times. (This will be relaxed later.) Due to the application of preemption, we can reduce this problem to an equivalent problem with identical mixing durations: For each mixing operation with duration T_{mix} greater than one, we can replace it with T_{mix} unit-duration mixing operations as shown in Figure 2.11. After this replacement step, the tree structure is changed and the problem with different mixing durations is reduced to an equivalent problem with identical mixing durations. So we can solve the equivalent problem with identical mixing operations using Algorithm 2 and get schedule F. Construct the optimal schedule for the

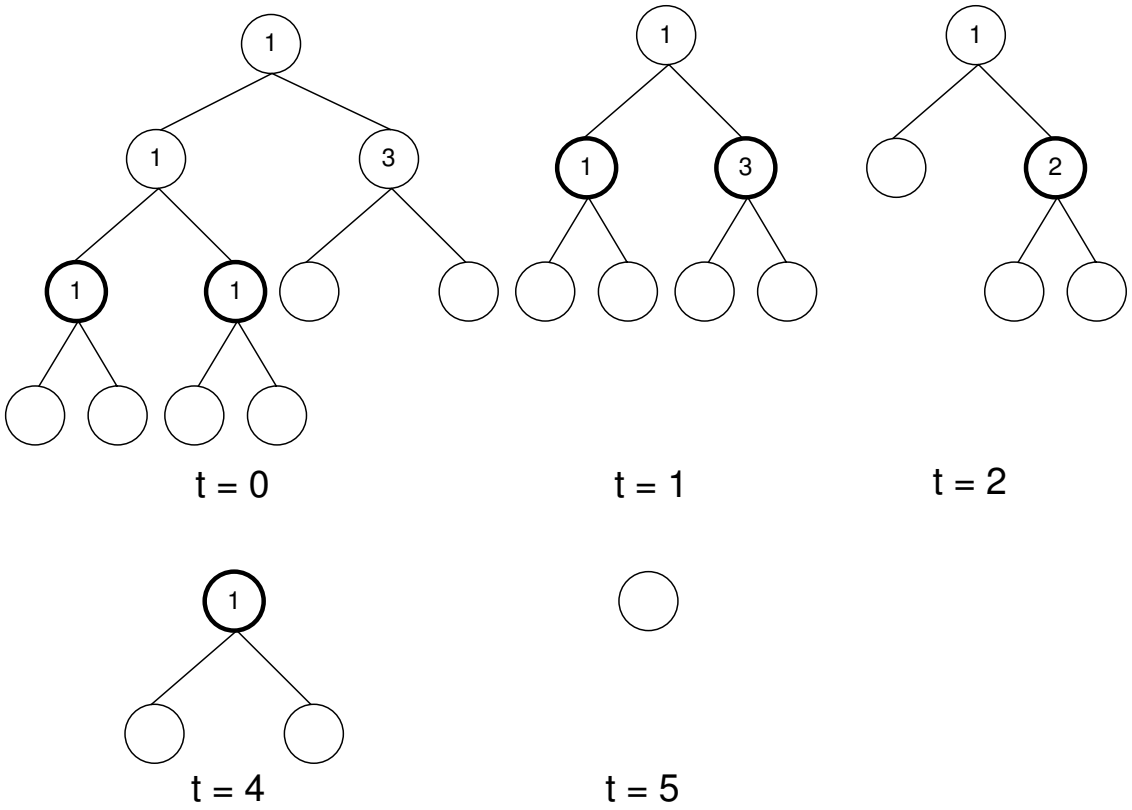


Figure 2.6: The schedule by Algorithm 3 for the chemical analysis tree of Fig 2.5. t denotes the lapsed time from the start of the mixing. A series of T updated in Algorithm 3 are shown with time labels t. The completion time is 5 using Algorithm 3.

problem with different mixing operations exactly the same as F. Note that if a storage unit is required for the new nodes introduced to a mixing operation by the replacement (e.g., nodes D and E in Figure 2.11), then the mixing operation is interrupted when performed and its mixer is preempted at that time. Also if we increase the resolution of mixing durations and reduce one unit of mixing duration to an arbitrarily small value ϵ, the problem intrinsically remains the same since the relative lengths of mixing durations do not change. Keeping decreasing ϵ will demonstrate that the above method works for continuous time as well.

Also, after replacement of mixing operations, the minimum number of mixers to achieve the lower bound can be calculated using Theorem 1.

2.4 Resource Constraint Analysis for Two Extreme Cases

The resource constraints come from two aspects: the number of mixers and the number of storage units. The former limits how many mixing operations we can perform in a time slot. The latter limits the number of droplets that can be stored for mixing after having been produced.

2.4.1 Case 1: Scheduling with Only One Mixer

2.4.1.1 Identical Mixing Durations

At least one mixer is required to be able to perform any mixing operations. Let Q denote the number of mixing operations to perform for the biochemical analysis.

Lemma 4 *With one mixer, all scheduling methods that keep the mixer busy will*

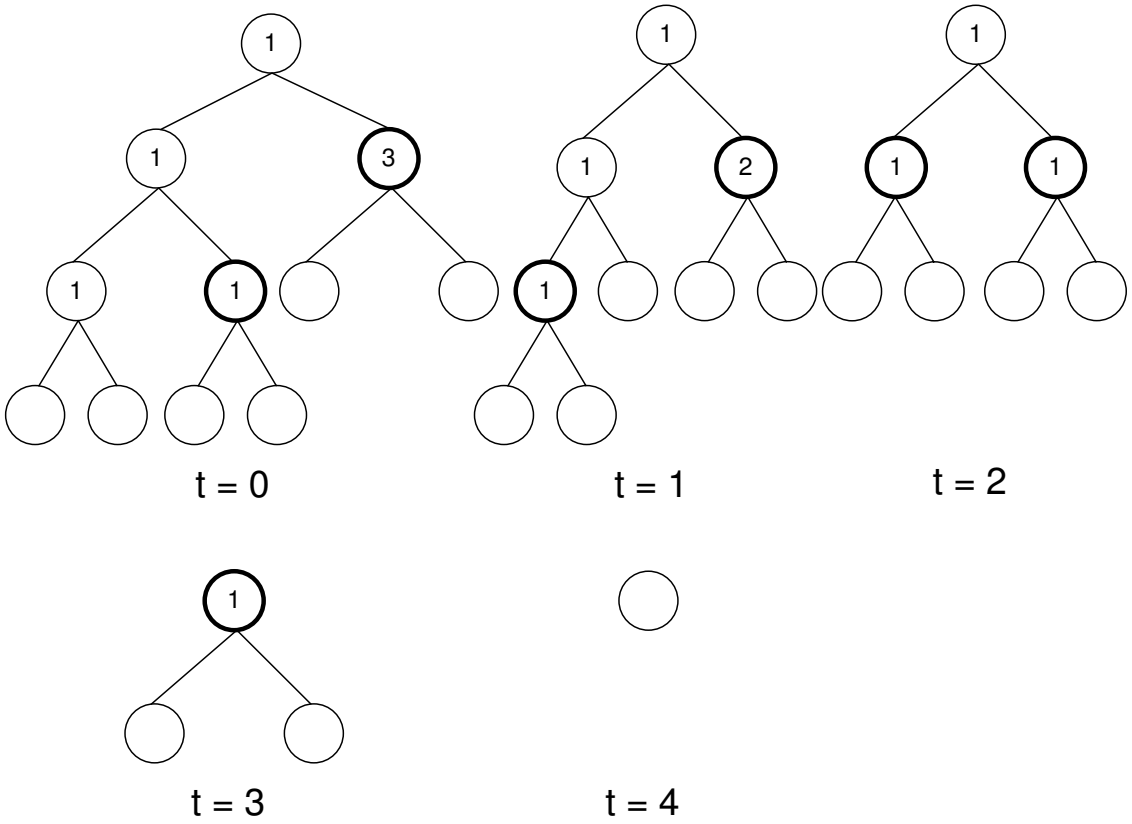


Figure 2.7: The optimal schedule for the biochemical analysis tree in Fig 2.5. A series of T updated in the optimal schedule are shown with time labels t. The completion time is 4 for the optimal schedule.

Algorithm 4 Scheduling-Diff-Mixing-Durations-Length

Input: T, M *// The binary tree and the number of mixers*
Output: F, t_{mix}^{total} *// Schedule and mixing completion time*
$t = 0$ // present time
$F = \emptyset$
Trace the tree to compute the lengths of nodes
while $T \neq$ a tree with just a root node **do**
  $C = \emptyset$ *// set of mixings that can be performed currently*
  Find pairs of leaf nodes with the same parent nodes in T *// identify mixing operations ready to be performed*
  Store them in C, in the decreasing order of the length
  if $t == 0$ **then**
    if $|C| < M$ **then**
      $F = F \bigcup$ {Schedule all mixing operations in C at time t}
    else
      $F = F \bigcup$ {Schedule the first M mixing operations of C at time t}
    end if
  end if
  Find the first produced node i
  $t = t + T_{mix}^{i}$
  Update T by removing leaf nodes involving the performed mixing operations
  if $C \neq \emptyset$ **then**
    $F = F \bigcup$ {Schedule the first mixing operation in C at time t}
  end if
end while
$t_{mix}^{total} = st + T_{mix}^{T.root}$
return F, t_{mix}^{total}

have the same t_{mix}^{total}*:*

$$t_{mix}^{total} = Q \cdot T_{mix} \tag{2.7}$$

Proof. No matter which scheduling method we use, only one mixing operation can be performed in each time slot due to the mixer resource constraint. As long as the mixer keeps busy, the mixing completion time is the sum of durations of all mixing operations. $t_{mix}^{total} = Q \cdot T_{mix}$. ■

Given a reaction tree T, we design Algorithm 8 to compute the minimum number of storage units, and Algorithm 6 to generate the corresponding schedule. Suppose $T.root$ is T's root node, $T.left$ and $T.right$ are the left and right subtrees of T.

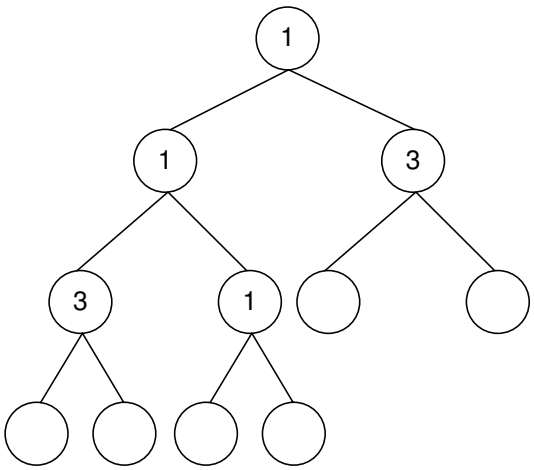


Figure 2.8: A counterexample to the optimality of Algorithm 4 with two mixers. The numbers inside non-leaf nodes are the remaining mixing times to produce the nodes. Note that there are no numbers inside leaf nodes since they have been produced.

Since the tree is a full binary tree, either $T.root$ is a leaf node, or both $T.left$ and $T.right$ are non-null.

Theorem 3 *The minimum number of storage units for a reaction using one mixer is computed by Algorithm 8; the corresponding schedule in the order of execution is output by Algorithm 6.*

Proof. Suppose $S(T)$ is the minimum number of storage units for the reaction represented by tree T using one mixer. First we prove the first half of the theorem. If $T.root$ is a leaf node or $T.value = 1$, $S(T) = 0$. (Base step)

If $T.root$ is not a leaf node, $S(T) \geq \max(S(T.left), S(T.right))$.

If $S(T.left) \neq S(T.right)$, without loss of generality assume $S(T.left) > S(T.right)$. Schedule as follows: first, perform all mixing operations in the left subtree, when we need $S(T.left)$ storage units; then perform those in the right subtree, when we need $S(T.right)$ storage units for the nodes in the right subtree and one for the $T.left$ node. So $S(T) = \max(S(T.left), S(T.right))$.

If $S(T.left) = S(T.right)$, we show that $S(T) = S(T.left) + 1$. Perform all mixing operations in the left subtree, and then the right subtree. Here we need $S(T.left) + 1$ storage units. So $S(T) \leq S(T.left) + 1$. If $S(T) < S(T.left) + 1$, then two conditions should be satisfied: first, in the time slot when $S(T.left)$ storage

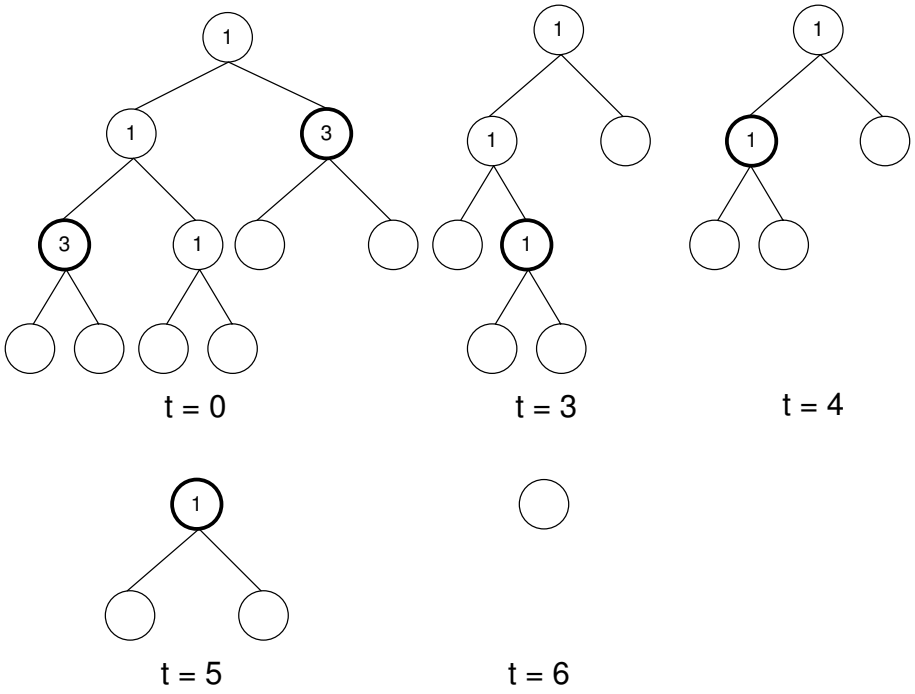


Figure 2.9: Schedule by Algorithm 4 for the biochemical analysis tree of Fig 2.8. t denotes the lapsed time from the start of the mixing. The numbers inside non-leaf nodes are the remaining mixing times to produce the nodes. Note that there are no numbers inside leaf nodes since they have been produced. A series of T updated in Algorithm 4 are shown with time labels t. The completion time is 6 using Algorithm 4.

units are used for the left subtree, no mixing operations in the right subtree have been performed; second, in the time slot when $S(T.right)$ storage units are used for the right subtree, no mixing operations in the left subtree have been performed. Obviously, they cannot be satisfied at the same time. By contradiction, we conclude $S(T) = S(T.left) + 1$. The induction step is also correct. So N returned by the algorithm equals $S(T)$, the optimal value.

We now show that the second half of the theorem is correct. Since Algorithm 6 outputs the mixing operations in child-node-first order, the mixing precedence rule is satisfied. Also, the algorithm traces back through $T.value$ for all subtrees using the recurrence in Algorithm 8, so it outputs the corresponding scheduling results. ■

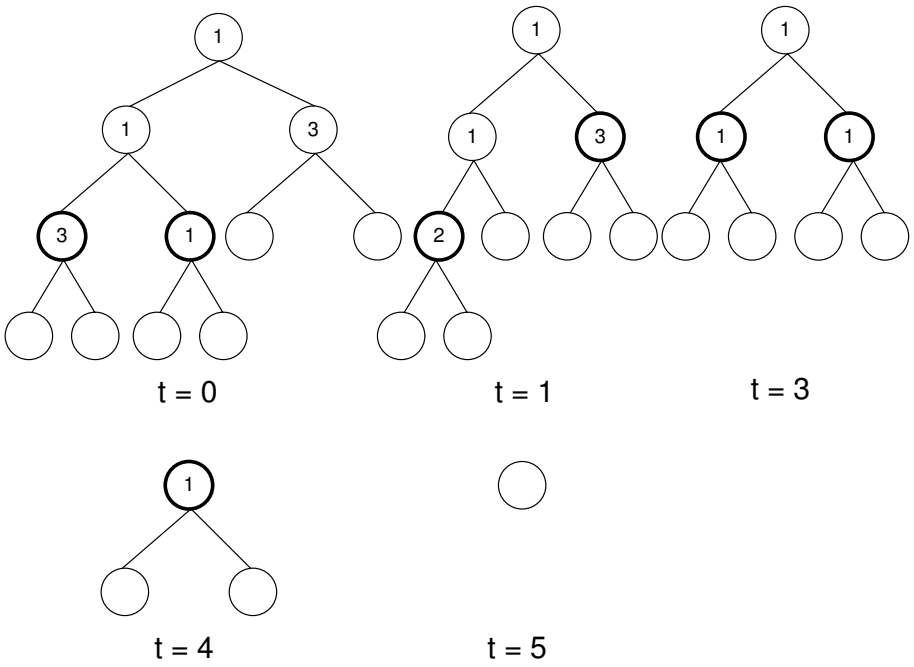


Figure 2.10: The optimal schedule for the biochemical analysis tree in Fig 2.8. A series of T updated in the optimal schedule are shown with time labels. The completion time is 5 for the optimal schedule.

2.4.1.2 Extension to Different Mixing Durations

The two conclusions above still hold true: 1. with one mixer, the total completion time is still the sum of all mixing durations as long as we keep the mixer busy all the time. (The proof of this conclusion is exactly the same as the case with an identical time duration). 2. Algorithm 8 still computes the minimum number of storage units and Algorithm 6 still returns the corresponding schedule. With one mixer, the structure of the tree decides the number of required storage units. Since the structure is independent of the mixing durations, the minimum number of storage units and corresponding schedule will not change.

2.4.2 Case 2: Scheduling with Zero Storage Units

2.4.2.1 Unique Mixing Durations

Theorem 4 *The sufficient and necessary condition of performing all the mixing operations without storage units is to use* $\max_{i=0}^{K} \frac{N_i}{2}$ *mixers, and the completion time in this case is the lower bound* $K \cdot T_{mix}$.

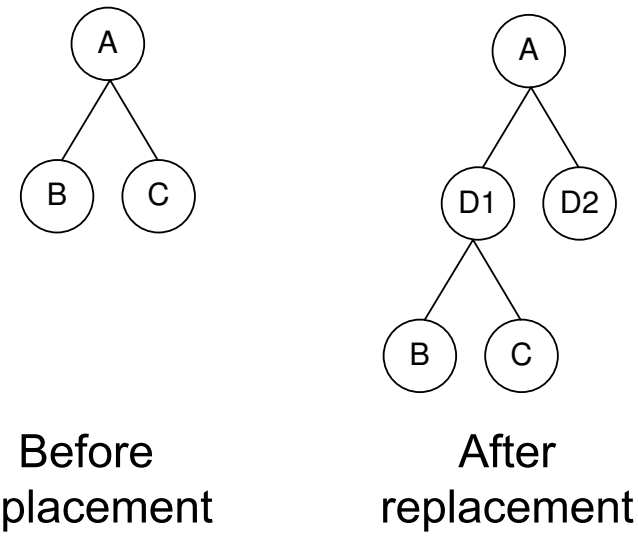


Figure 2.11: Reduce the problem with different mixing durations to the problem with identical mixing durations when allowing preemptions in discrete time. $T^A_{mix} = 2$. Using preemptions, a tree with different mixing durations for its mixing operations can be replaced by another tree with unit mixing duration for all mixing operations ($T^A_{mix} = T^{D_1}_{mix} = 1$). Nodes D_1 and D_2 are intermediate nodes of mixing and splitting nodes B and C, whose mixing operation to produce A is preempted. The connections of nodes A, B, C to other outer nodes should be maintained as before.

Proof. Actually, we have considered the sufficient condition when discussing the lower bound of mixing completion time in the proof of Lemma 1. There, using $\max_{i=0}^{K} \frac{N_i}{2}$ mixers, we provide a scheduling algorithm, which achieves the lower bound of the completion time and needs zero storage units. So we only need to prove that it is also the necessary condition for performing all mixing operations with zero storage units.

First we show that all mixing operations for non-leaf nodes at the same level must be performed in the same time slot if there are zero storage units.

Base Step: There is only one mixing operation for the root node at level 0, so the conclusion is correct obviously.

Induction Step: For some level i, suppose the conclusion holds true for all levels $j < i$. If the conclusion does not hold true for level i, there must be at least two non-leaf nodes A and B in level i, and say A is produced before B. According to the induction hypothesis, A and B are consumed for other non-leaf nodes at level $i-1$ in

Algorithm 5 Min-Storage-Units

Input: T // the binary tree
Output: N // the number of required storage units
if $T.root$ is a leaf node **then**
  $T.value = 0$
else
  $Left$=Min-Storage-Units($T.left$)
  $Right$=Min-Storage-Units($T.right$)
  if $Left = Right$ **then**
    $T.value = Left + 1$
  else
    $T.value = \max(Left, Right)$
  end if
end if
if T is the initial input tree & $T.value \neq 0$ **then**
  $N = T.value - 1$
else
  $N = T.value$
end if
return N

the same time slot. Obviously, one storage unit is required for A. By contradiction, we get the conclusion.

Since all mixing operations for non-leaf nodes at the same level must be performed in the same time slot, the number of mixers is $\max_{i=0}^{K} \frac{N_i}{2}$ as desired. ■

2.4.2.2 Extension to Different Mixing Durations

In Lemma 2, we have computed the lower bound in the case of different mixing durations. However, since different mixing operations in the same level may have different durations, the number of mixers required for a zero-storage-unit schedule can not be derived from the number of mixing operations in the same level. Instead we design Algorithm 7 to compute the number of mixers and schedule for the case of zero storage units below. The basic idea of scheduling in Algorithm 7 is

- First perform the mixing operations with the biggest length and then proceed upwards.
- Whenever the lower bound minus the lapsed time equals to the length of a

Algorithm 6 One-Mixer-Scheduling

Input: T, i *// the binary tree*
// time slot index
Output: F *// Schedule*
$i = 1$
$F = \emptyset$
if $T.value = 0$ **then**
  return
else
  if $T.left.value > T.right.value$ **then**
    One-Mixer-Scheduling($T.left$, i)
    One-Mixer-Scheduling($T.right$, i)
  else
    One-Mixer-Scheduling($T.right$, i)
    One-Mixer-Scheduling($T.left$, i)
  end if
end if
$F = F \bigcup$ {Schedule mixing to get $T.root$ in time slot i}
$i = i + 1$
return F

mixing operation, perform that mixing operation.

Based on the schedule, each time a droplet is produced, a mixer will be released and the number of mixers will decrease by 1; each time a mixing operation starts, a mixer will be taken up and the number of mixers will increase by 1. We can record the largest number of mixers during the schedule and it is the number of mixers required for the case with zero storage units.

2.5 Example

In this section, we apply our scheduling analysis and algorithms to the PCR analysis. Let $t_{transport}$ be the transportation time from one electrode to its neighboring electrode. The parameter values we use are: $T_{mix} = 5$ *sec*, $T_{input} = T_{output} = 1$ *sec*, $t_{transport} = 0.01$ *sec/electrode*. (In the experiments of [5] and [8], $t_{transport} = 0.006 - 0.013$ *sec/electrode*.) The depth of binary tree for PCR analysis is $K = 4$.

The results are shown in Table 2.1. Mixing time takes up most of the completion time since pipelining is used to overlap other operations with mixing. Exact

Algorithm 7 Mixer-Number-For-Zero-Storage-Units

Input: T // the binary tree
Output: M,F // Number of mixers and schedule
$F = \emptyset$, $M = 0$
// Compute i.time, the length from the root node to i
for each node i in T **do**
  if $i = T.root$ **then**
    $i.time = 0$
  else
    $i.time = j.time + T^j_{mix}$
    // Suppose j is i's parent node
  end if
end for
for each non-leaf node i **do**
  $s_i = t_{lb} - j.time$, $e_i = t_{lb} - i.time$ *// start, end time*
  // Suppose j is i's left child node
  // t_{lb} can be computed using Algorithm 1
end for
Sort all s_i,e_i in decreasing order and if $s_m = e_n$, e_n should be before s_m
$n = 0$
while visit all s_i, e_i in order after sorting **do**
  if the value is s_i **then**
    $n = n + 1$
    $F = F \bigcup \{$Schedule the mixing operation at time s_i to achieve node $i\}$
    if $n > M$ **then**
      $M = n$
    end if
  else
    $n = n - 1$
  end if
end while
return M, F

Table 2.1: Results of PCR analysis using one and two mixers. All times are in seconds. Approximate completion time does not consider transportation time.

Num of Mixers	Num of Storage Units	Total Mixing Time	Completion Time	
			Approx.	Exact
1	1	35	37	37.8
2	0	20	22	22.54

completion time (calculated by Equation 2.1) is almost the same as approximate completion time (calculated by Equation 2.2) since the transportation speed is very fast. (During one mixing duration, droplets can be transported over $\frac{5}{0.01} = 500$ electrodes.) The mixing time using two mixers reduces to 57% of that using one mixer due to parallel mixing operations. (The percentage is larger than 50% since one mixer is idle when the last mixing is performed.) By Theorem 1, $T_{lowerbound} = 20$ *sec*, which is the mixing time shown in Table 2.1 using two mixers.

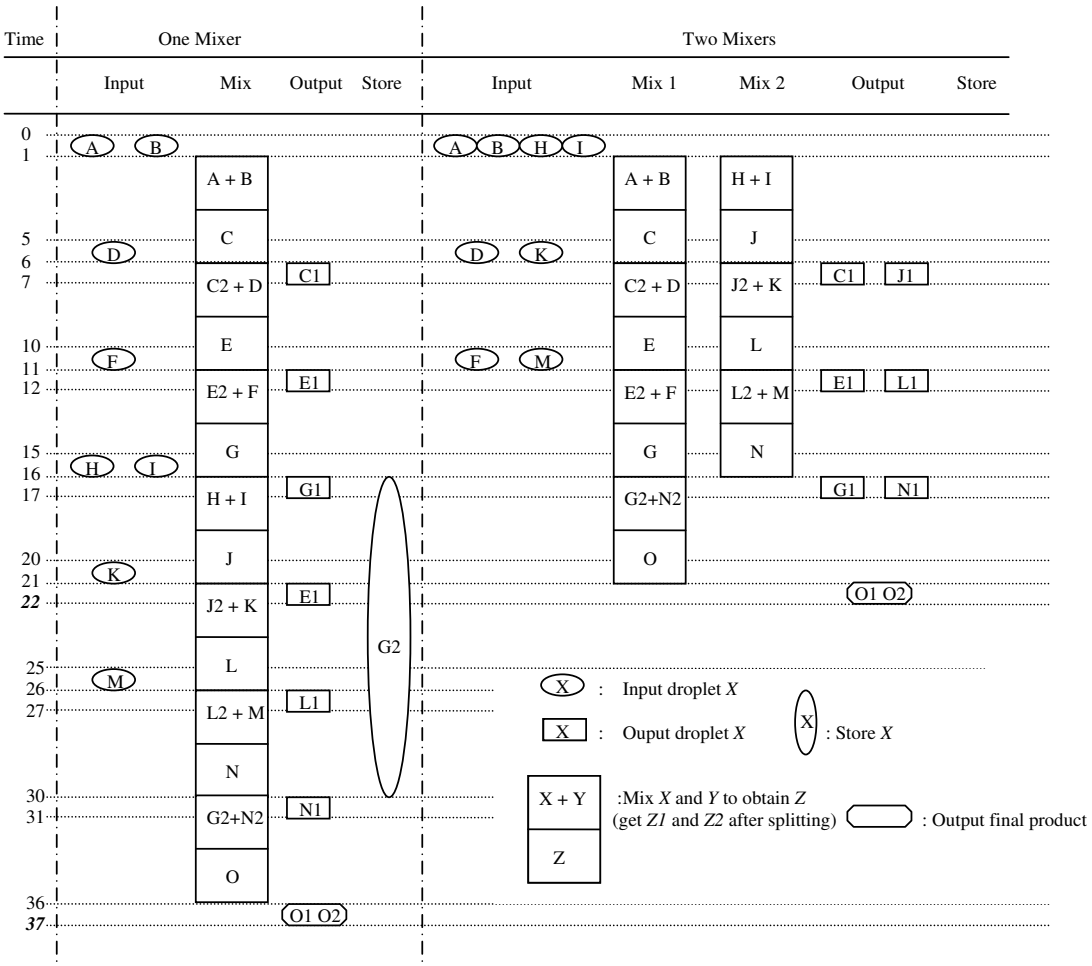


Figure 2.12: Scheduling results of performing PCR analysis on DMFS. The left side shows results with one mixer, while the right side shows result with two mixers. Transportation time is ignored here. Accurate completion time considering transportation time is shown in Table 2.1. All times are in seconds.

We illustrate the scheduling results when one and two mixers are used in

Fig. 2.12. We can see that the pipelining technique is used to overlap different kinds of operations such as *input*, *transport*, *mix*, *store* and *output*. Comparing the results with one and two mixers, we see that more mixers can exploit the parallelism inside the mixing operations of PCR analysis. By Theorem 1, the lower bound of completion time has been achieved, so more than two mixers cannot decrease the completion time any more.

2.6 Conclusion

In this chapter we presented scheduling algorithms to optimize the completion time of biochemical analyses on a DMFS by exploiting the tree structure of these reactions. We considered a binary tree representation of chemical analyses to schedule operations. Using pipelining, we overlapped mixing operations with input and transportation operations. We found the lower bound of the mixing completion time according to the tree structure of input analyses, and calculated the minimum number of mixers M_{min} required to achieve the lower bound. We presented a scheduling algorithm for the case with a specified number of mixers no greater than M_{min}, and proved it is optimal to minimize the mixing completion time. We also analyzed resource constraint issues for two extreme cases. For the case with just one mixer, we proved that all schedules result in the same mixing completion time as long as the mixer was kept busy at all times and then designed a scheduling algorithm to minimize the number of storage units. For the case with zero storage units, we found the minimum number of mixers required. Both the case with identical mixing duration and that of different mixing durations were analyzed and solved in this chapter. Finally, we demonstrated the benefits of our scheduling methods on an example of DNA polymerase chain reaction (PCR) analysis. Other cases of much more complicated biochemical analysis can also be handled in a similar fashion. The analysis and algorithms can easily be extended to perform several independent biochemical analyses in parallel.

3. Minimizing the Resource Requirements

In Chapter 2 we focused on algorithms to minimize time based on the tree structure of the chemical analysis. Here we focus on minimizing spatial resources based on the tree structure of the chemical analysis and using it to design (or select) the smallest chip for a given analysis. There are several classes of resources (i.e., functional components consisting of electrodes) in a DMFS: mixers, storage units, input and output units, and transportation paths. We focus on performing a biochemical analysis in batch mode, where the goal is to produce one droplet of the final product. We assume that the resources (e.g., mixers and storage units) are fixed at the start of the analysis and do not change over the course of the analysis.

3.1 Problem Formulation

The problems we want to solve in this chapter are: *Given a biochemical analysis, what are the minimum requirements of resources (e.g., the number and size of mixers, storage units, input/output units and transportation paths) to perform it? How can we design the scheduling algorithm so that the analysis can be performed with the minimum resources?* The number of input and output units is determined by the given biochemical analysis. Transportation paths are constructed so that other function units (the input units, output units, mixers and storage units) are connected. We will first focus on the resource requirements of mixers and storage units, and then consider the requirements of other resources as well in Section 3.3.

When discussing the minimum resource requirements, we can just focus on the mixer scheduling.

3.1.1 Mixers and Storage Units

The number of mixers limits how many mixing operations we can perform in a time slot. The number of storage units gives a constraint for droplets that have been produced but cannot immediately be mixed. Figure 3.1 shows how mixers and storage units are used during the progress of a biochemical analysis.

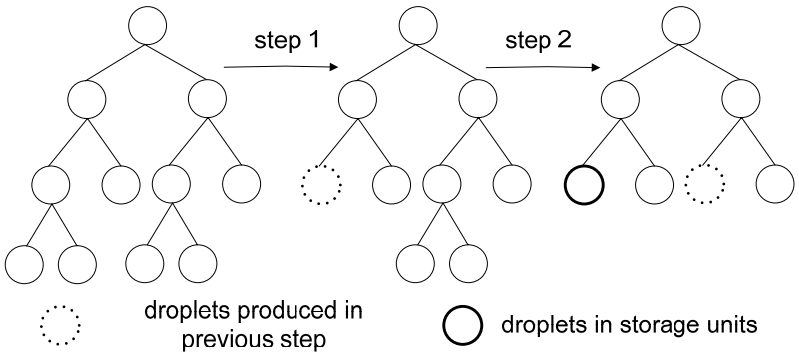


Figure 3.1: Progress of a biochemical analysis on a chip with a single mixer. The intermediate reagent produced in the first step is stored in a storage unit since it is not involved in the mixing in the second step.

3.1.2 Scheduling Representation

Each scheduling step can be regarded as the transition from one full binary tree to another where some sibling nodes are removed (see Figure 3.1). So the whole schedule *Sch* of a biochemical analysis can be represented by a series of full binary trees, starting from the initial full binary tree and ending at the root node, and the transitions between consecutive trees correspond to removing selected sibling nodes.

For a schedule *Sch*, let the number of mixers be M and the number of storage units be S. We can formulate the scheduling problem as the following two questions: *Given an initial full binary tree T, (1) What is the schedule Sch in which S is minimized given M? (2) What is the schedule Sch in which the resource requirements (i.e., the number of electrodes for M mixers, S storage units, input/output units and transportation paths) are minimized?*

3.2 Minimizing the Number of Storage Units with Given Number of Mixers

In Section 2.4 of previous chapter, we analyzed two extreme cases of resource requirements: with just one mixer and with zero storage units. In the first case, we proved all active schedules (i.e., that keep the mixer busy at all times) result in the same completion time, calculated the minimum number of storage units for a given analysis, and designed a corresponding scheduling algorithm. In the second case,

we computed the minimum number of mixers required, and proved the following conclusion in Theorem 4. *The sufficient and necessary condition for performing all the mixing operations without storage units is to use* $\max_{i=0}^{K} \frac{N_i}{2}$ *mixers, where* K *is the depth of the tree,* N_i *is the number of nodes at the* i*th level.*

In this chapter, we extend the scheduling algorithm to handle the general case with M mixers and S storage units, and characterize the relationship between the number of mixers and the number of storage units required for a biochemical analysis based on its tree structure.

Given an initial full binary tree of a biochemical analysis, M mixers, and S storage units, we need to check whether or not the given biochemical analysis can be performed under such constraints. Since two constraints exist here, there are two methods to solve this problem. First, with at most M mixers, compute the minimum number of storage units $f(M)$. If $f(M) \leq S$, it is feasible to perform the biochemical analysis under such constraints, and infeasible otherwise. Second, with at most S storage units, compute the minimum number of mixers $g(S)$. Similarly, compare $g(S)$ and M to check the feasibility. Since we may reduce the required number of one resource by increasing the number of the other resource, it is obvious that $f(M)$ and $g(S)$ are both non-increasing functions. So if we have a solution for the first method, the second one can be solved using binary search in the result of the first method, and vice versa.

In this section, first, we will compute the minimum number of storage units $f(M)$ and design the scheduling algorithm to perform the biochemical analysis using M mixers and $f(M)$ storage units; second, we will define $f(M)$ as the M-depth of the tree structure and give an equation to compute the M-depth for complete trees; third, we characterize the variation with M of the M-depth of the tree. So for any biochemical analysis, we can find out whether it can be performed using M mixers and S storage units by computing the M-depth of its tree representation. If two biochemical analyses have tree structures with the same M-depth, we say they have the same resource requirements.

3.2.1 Algorithm Design

Algorithm 8 Min-Storage-Units-M-Mixers

Input: T, M // the binary tree and the number of mixers
Output: f_M // the number of required storage units
while Scan nodes level by level from bottom up: **do**
  for any scanned node i **do**
    $i.f_M = 0$ // *storage units to produce i*
    if $i.left.f_M == 0$ and $i.right.f_M == 0$ **then**
      // *zero storage units to produce both child nodes*
      Scan the subtree rooted at i, set $i.maxnode$ as the maximum number of nodes at any level in the subtree
      if $i.maxnode > 2M$ **then**
        $i.f_M = 1$
      end if
    else
      // *at least one child node needs storage units*
      if $i.left.f_M == i.right.f_M$ **then**
        $i.f_M = i.left.f_M + 1$
      else
        $i.f_M = \max\{i.left.f_M, i.right.f_M\}$
      end if
    end if
  end for
end while
return $T.root.f_M$

Theorem 5 *Algorithm 8 returns the minimum number of storage units for a reaction using at most M mixers; Algorithm 9 outputs the corresponding schedule in the order of execution.*

Proof. Suppose $f_M(T)$ is the minimum number of storage units for the reaction represented by tree T using at most M mixers. First we prove the first part of the theorem. According to Theorem 4, if the maximum number of nodes at the same level is greater than $2M$, at least one storage unit is required. Since the algorithm scans nodes from bottom up, the nodes first satisfying this condition need one storage unit. (Base step.)

If $T.root$ is not a leaf node, obviously, $f_M(T) \geq \max(f_M(T.left), f_M(T.right))$.

If $f_M(T.left) \neq f_M(T.right)$, without loss of generality assume $f_M(T.left) > f_M(T.right)$. Schedule as follows: First, perform all mixing operations in the left subtree, when we need $f_M(T.left)$ storage units. Then perform those in the right

Algorithm 9 M-Mixer-Scheduling

```
Input: T, i // the binary tree and the time slot index (1 for the initial tree)
Output: F // Schedule, global variable initialized as ∅
if T has only a root node then
  return
else
  if T.root.f_M = 0 then
    // finish mixing operations at one level each time slot
    for all levels of T from bottom up do
      F = F ⋃ {Schedule all mixing operations at the same level in time slot i}
      i = i + 1
    end for
  else
    // first produce the child node requiring more storage units
    if T.root.left.f_M > T.root.right.f_M then
      M-Mixer-Scheduling(T.left, i)
      M-Mixer-Scheduling(T.right, i)
    else
      M-Mixer-Scheduling(T.right, i)
      M-Mixer-Scheduling(T.left, i)
    end if
    F = F ⋃ {Schedule the mixing operation to get T.root in time slot i}
    i = i + 1
  end if
end if
return F
```

subtree, when we need $f_M(T.right)$ storage units for the nodes in the right subtree and one for the $T.left$ node. So $f_M(T) = \max(f_M(T.left), f_M(T.right))$.

If $f_M(T.left) = f_M(T.right)$, we show that $f_M(T) = f_M(T.left) + 1$. Schedule all mixing operations in the left subtree, and then the right subtree. We need $f_M(T.right)+1$ storage units as discussed above, which is equal to $f_M(T.left)+1$. So $f_M(T) \leq f_M(T.left)+1$. If $f_M(T) < f_M(T.left)+1$, then two conditions should be satisfied: First, in the time slot when $f_M(T.left)$ storage units are used for the left subtree, no mixing operations in the right subtree have been performed. Second, in the time slot when $f_M(T.right)$ storage units are used for the right subtree, no mixing operations in the left subtree have been performed. Obviously, they cannot be satisfied at the same time. By contradiction, we conclude $f_M(T) = f_M(T.left) + 1$. The induction step is also correct. So f_M returned by the algorithm is $f_M(T)$, the

optimal value.

We now show that the second part of the theorem is correct. Since Algorithm 9 outputs the mixing operations in child-node-first order, the mixing precedence rule is satisfied. Also, the algorithm traces back through $T.root.f_M$ for all subtrees using the recurrence in Algorithm 8, so it outputs the corresponding scheduling results. ■

For a complete binary tree of depth 2, its 1-depth is 1 and 2-depth is 0 by Algorithm 8.

3.2.2 M-Depth of Tree Structure

From the algorithm in the last section, we see that with M mixers, the minimum number of storage units $f(M)$ for a biochemical analysis only depends on its tree structure. Thus we can define $f(M)$ as the M-depth of the tree structure corresponding to M mixers. Below we design Algorithm 10 to calculate $i.subtree$, the number of nodes at each levels in the subtree and compute $i.maxnode$, the maximum number of nodes at the same level in the subtree rooted at each node i.

In Algorithm 10, $i.subtree$ has the format of $(a_0, a_1, a_2, \ldots)$ where a_j is the number of nodes at level j of the subtree. "Append a zero to $(a_0, a_1, \ldots, a_i)$" will result in $(a_0, a_1, \ldots, a_i, 0)$, e.g., "Append a zero to (1)" will result in $(1, 0)$; the "$\oplus$" operator for $i.subtree$ works as follow: $(a_0, a_1, \ldots, a_k) \oplus (a'_0, a'_1, \ldots, a'_k) = (a_0 + a'_0, a_1 + a'_1, \ldots, a_k + a'_k)$, e.g., $(1, 0, 0) \oplus (1, 2) \oplus (1, 2) = (1, 2, 4)$. Please note, by appending zeros to subtrees, Algorithm 10 guarantees that the subtrees on two sides of "$\oplus$" have the same length. Figure 3.2 shows $i.subtree$ computed by Algorithm 10 for each node in a tree.

Using Algorithm 8 with Algorithm 10, we can compute $f(M)$ for the tree structure of any biochemical analysis. Since in Algorithm 8 we do not need the $i.subtree$ information for parent nodes of nodes with $i.maxnode > 2M$, the algorithm can be improved by skipping $i.subtree$ calculation for such nodes.

From Theorem 4, we can easily see that:

Algorithm 10 Max-Nodes-Same-Level

Input: T // the binary tree
Output: i.maxnode // for each node i in T
$Height = 0$ *// height equals depth minus level*
while Scan nodes level by level from bottom up: **do**
  for any scanned node i **do**
    $i.subtree = (1)$ *// record subtree rooted at i*
    // Record subtree for leaf nodes at the level
    $TempHeight = Height$
    while $TempHeight > 0$ **do**
      Append a zero to the end of $i.subtree$
      $TempHeight = TempHeight - 1$
    end while
    // Record subtree for non-leaf nodes at the level
    if i is not a leaf node **then**
      $i.subtree = i.subtree \oplus i.left.subtree \oplus i.right.subtree$
    end if
    set $i.maxnode$ as the maximum number in $i.subtree$
  end for
  $Height = Height + 1$
end while
return T with $i.maxnode$

Lemma 5 *Given a tree for a biochemical analysis,* $f(M) = 0$ *if and only if* $M \geq \frac{T.root.maxnode}{2}$.

For complete trees, we derive an equation to compute M-depth as follows.

Theorem 6 *For a complete tree with depth D, its M-depth can be computed:*

$$f(M) = D - (\lfloor log_2 M \rfloor + 1) \tag{3.1}$$

Proof. Suppose in a tree with depth D, we define the height of a node as D minus its level. According to Lemma 5, in a complete tree, any node i with $i.f_M = 0$ must satisfy that $i.maxnode \leq 2M$ and thus has a height at most $\lfloor log_2 2M \rfloor$. And for nodes with height more than $\lfloor log_2 2M \rfloor$, increasing height by 1 will increase $i.f_M$ by 1. So

$$f(M) = T.root.f_M = T.root.height - \lfloor log_2 2M \rfloor$$

$$= D - \lfloor log_2 2M \rfloor = D - (\lfloor log_2 M \rfloor + 1). \blacksquare$$

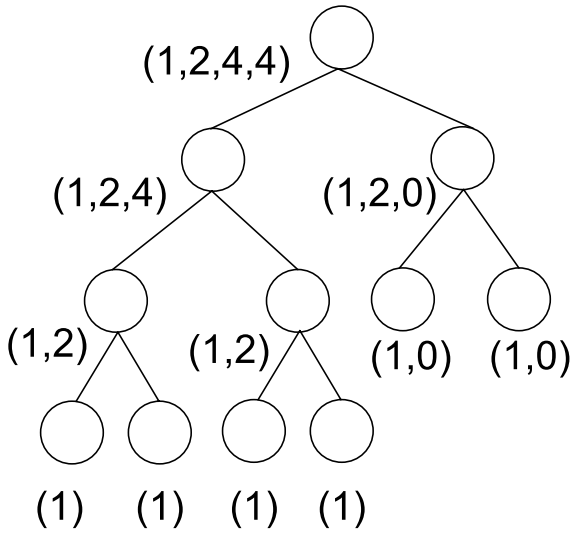


Figure 3.2: *i.subtree* **computed by Algorithm 10 for a tree.**

Corollary 1 *For a binary tree with depth D for a biochemical analysis, its M-depth must satisfy:*

$$f(M) \leq D - (\lfloor log_2 M \rfloor + 1) \tag{3.2}$$

3.2.3 Variation of M-depth with M for a Tree

We first examine the non-increasing property of the M-depth, $f(M)$. For a given biochemical analysis tree, according to Lemma 5, if $f(M) = 0$, $f(M+1) = 0$. There exist nodes where $i.f_M > 0$, but $i.f_{M+1} = 0$. Considering the same recurrence as in Algorithm 8, for each node i, $i.f_M \geq i.f_{M+1}$. So $f(M) = T.root.f_M \geq T.root.f_{M+1} = f(M+1)$. That is, with more mixers, we may reduce the minimum number of storage units required.

$$f(M+1) \leq f(M) \tag{3.3}$$

Second, there will not be much difference between $f(M)$ and $f(M+1)$.

Theorem 7

$$f(M+1) \geq f(M) - 1 \tag{3.4}$$

Proof. Use mathematical induction. Base step is trivial: for a leaf node, $f(M+1) = f(M) = 0$.

For a node i with left and right child nodes, suppose $f(M)$ is M-depth of the tree rooted at i, $f_L(M)$ and $f_R(M)$ are M-depths of its left subtree and right subtree, respectively. According to the induction hypothesis,

$$f_L(M+1) \geq f_L(M) - 1, f_R(M+1) \geq f_R(M) - 1$$

There are three possible cases below:

1. $f_L(M) \neq f_R(M)$: Without loss of generality, assume $f_L(M) > f_R(M)$. Then $f(M) = f_L(M)$. Also $f(M+1) \geq f_L(M+1) \geq f(M) - 1$, so $f(M+1) \geq f(M) - 1$.

2. $f_L(M) = f_R(M) = 0$: $f(M) \leq 1$. $f(M+1) \geq 0 \geq f(M) - 1$.

3. $f_L(M) = f_R(M) \neq 0$: $f(M) = f_L(M) + 1$.

   - $f_L(M) = f_R(M) \geq 2$: $f_L(M+1) \geq f_L(M) - 1 \geq 1$, $f_R(M+1) \geq f_R(M) - 1 \geq 1$. If $f_L(M+1) = f_R(M+1)$, $f(M+1) = f_L(M+1) + 1 \geq f_L(M)$. So $f(M+1) \geq f(M) - 1$. If $f_L(M+1) \neq f_R(M+1)$, (assume $f_L(M+1) > f_R(M+1)$) then $f_L(M+1) > f_R(M) - 1$, so $f_L(M+1) \geq f_L(M)$. So $f(M+1) \geq f_L(M+1) \geq f_L(M) = f(M) - 1$.

   - $f_L(M) = f_R(M) = 1$: $f(M) = 2$. From Lemma 5, $i.left.maxnode > 2M$ and $i.right.maxnode > 2M$. So $i.maxnode > 2(M+1)$ (At least two nodes of right (or left) subtree will be at the same level with more than $2M$ nodes in left (or right) subtree). By Lemma 5 again, $f(M+1) > 0$. So $f(M+1) \geq 1 = f(M) - 1$.

So the induction step is also correct. Thus $f(M+1) \geq f(M) - 1$.■

Considering Inequalities 3.3 and 3.4, we see that for a full binary tree, $f(M) \geq f(M+1) \geq f(M) - 1$. This means that using one additional mixer to perform a biochemical analysis either keeps the minimum number of storage units the same or reduces it by one. So we get the conclusion below.

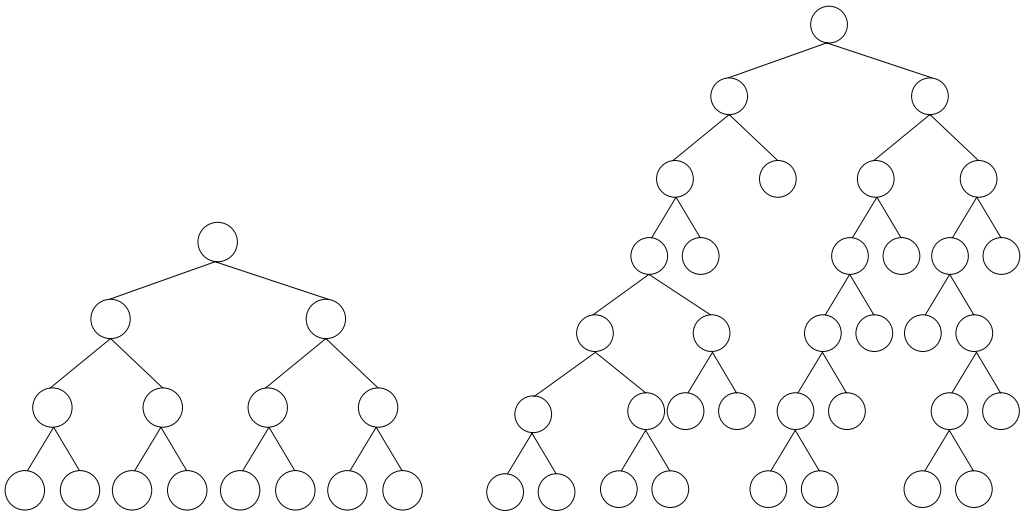
Corollary 2 *The total number of mixers and storage units $F(M) = M + f(M)$ is a non-decreasing function of M.*

Table 3.1: The total size of mixers and storage units for different M and $f(M)$.

Num of Mixers	Minimum Num of Storage Units	Total Size
1	2	33
2	1	39
3	1	54
4	0	60

3.2.4 Example

We apply the above algorithms and conclusions to characterize biochemical analyses based on resource requirements. The two trees in Figure 3.3 have different structures, but share the same resource requirements, as illustrated by their identical characteristic resource curve $f(M)$ in Figure 3.4. Suppose the size of one storage unit is 9 and the size of one mixer is 15, and we use the mixer and storage unit as shown in Figure 1.2 to construct the biochip. Table 3.1 shows the total size of mixers and storage units for different M and $f(M)$.



(a) $D = 3$ (b) $D = 6$

Figure 3.3: Two analysis trees that have the same least resource requirements.

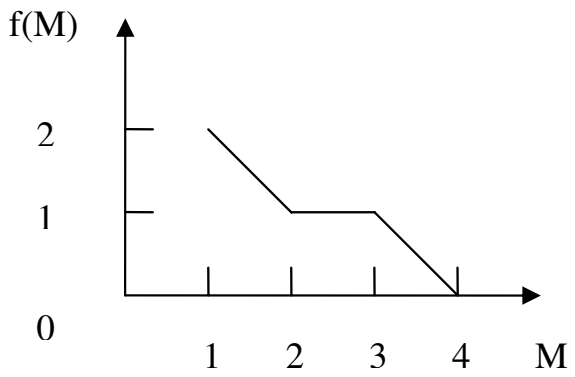


Figure 3.4: Characteristic resource curve of $f(M)$ shared by the two trees in Figure 3.3.

3.3 Towards Smallest Chip Design

We discussed mixers and storage units in Section 3.2, without considering their geometry and other resources such as input and output units and transportation paths. In this section, we will consider all resource requirements towards the design of the smallest DMFS chip for biochemical analyses. The smaller the number of electrodes in a DMFS chip, the easier the fabrication and the lower the cost.

3.3.1 Mixers and Storage Units

Mixers and storage units are the two primary functional components on a DMFS biochip, where mixers are used to mix and split droplets while storage units are used to store droplets on chip for future usage. They should be large enough to prevent droplets inside them inadvertently mixing with other droplets outside them. As shown in Figure 3.5, a storage unit needs only one electrode to hold one droplet, while a mixer needs more electrodes to move the mixed droplet inside it and thus needs more surrounding electrodes to keep the droplet inside separate from other droplets outside. It follows that the *size* of one mixer S_{mix} (the number of electrodes in one mixer) is bigger than the size of a storage unit S_{store}.

From the above observation and Corollary 2, we obtain the following result for a full binary tree whose M-depth is $f(M)$.

Theorem 8 *The total size of M mixers and $f(M)$ storage units will monotonically increase with the number of mixers.*

(a) (b)

Figure 3.5: The sizes of a mixer and a storage unit. (a) A mixer with 3 electrodes for droplets to move. $S_{mix} = 15$. (b) A storage unit with 1 electrode for droplets to stay. $S_{store} = 9$.

Proof. We need prove that $M_1 \cdot S_{mix} + f(M_1) \cdot S_{store} < M_2 \cdot S_{mix} + f(M_2) \cdot S_{store}$ when $M_1 < M_2$.

$M_1 \cdot S_{mix} + f(M_1) \cdot S_{store} = (M_1 + f(M_1)) \cdot S_{store} + M_1 \cdot (S_{mix} - S_{store})$
$\leq (M_2 + f(M_2)) \cdot S_{store} + M_1 \cdot (S_{mix} - S_{store}) < (M_2 + f(M_2)) \cdot S_{store} + M_2 \cdot (S_{mix} - S_{store})$
$= M_2 \cdot S_{mix} + f(M_2) \cdot S_{store}$ ■

3.3.2 Input and Output Units

Input and output units should be connected to the perimeter electrodes of the chip, which we call connection electrodes, as shown in Figure 3.6. For the tree in

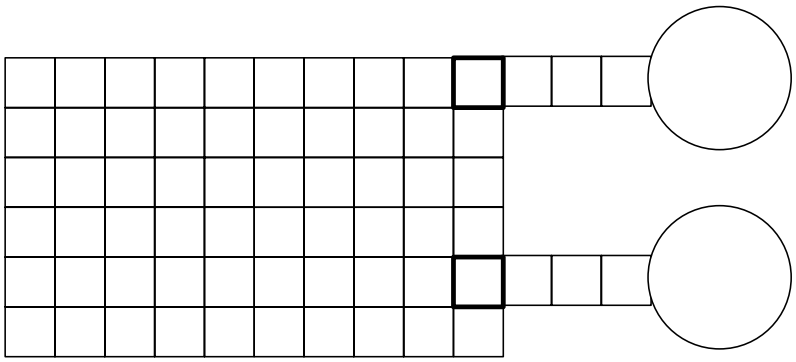


Figure 3.6: Input and output units are connected to the chip through connection electrodes (shown bold) on the edge of the chip.

Figure 3.7(a), we need just one mixer and zero storage units for the reaction. If we directly connect the input units to the chip as in Figure 3.7(b), the number of corresponding connection electrodes will be the number of leaf nodes of the tree in Figure 3.7(a). Since the connection electrodes should be on the perimeter of the

chip, the total number of electrodes on the chip will be proportional to the square of the number of connection electrodes, which is much bigger than the size of one mixer. (This assumes the chip is a rectangular array of electrodes.)

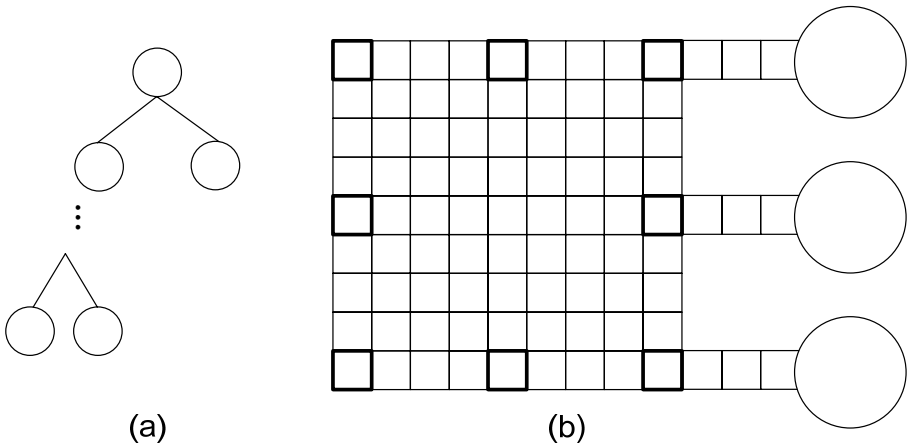


Figure 3.7: The size of chip for an analysis tree when directly connecting the input units to the chip. (a) A full binary tree of depth 6, where the right child of each node is a leaf node or a null node. (b) The chip containing sufficient connection electrodes, shown bold on the chip perimeter. (A subset of input units are shown.)

An alternative approach is to connect the input and output units to a ring of electrodes (as in [4]) and then connect the ring to a separate working zone of mixers and storage units as shown in Figure 3.8. The size of the working zone can be selected to have the required functionality, and the working zone can even be in the interior of the ring if it is sufficiently small.

3.3.3 Transportation Paths

Theorem 9 *Even considering the transportation paths, the smallest chip is constructed using one mixer and* $f(1)$ *storage units.*

Proof. Proof by contradiction. First, the smallest chip can be constructed using M mixers and $f(M)$ storage units since we can always convert extra storage units (if more than $f(M)$) to transport electrodes without increasing the chip size. Second, assume $M > 1$ in the smallest chip. The corresponding transportation paths should connect all mixers and storage units so that they are reachable from each other.

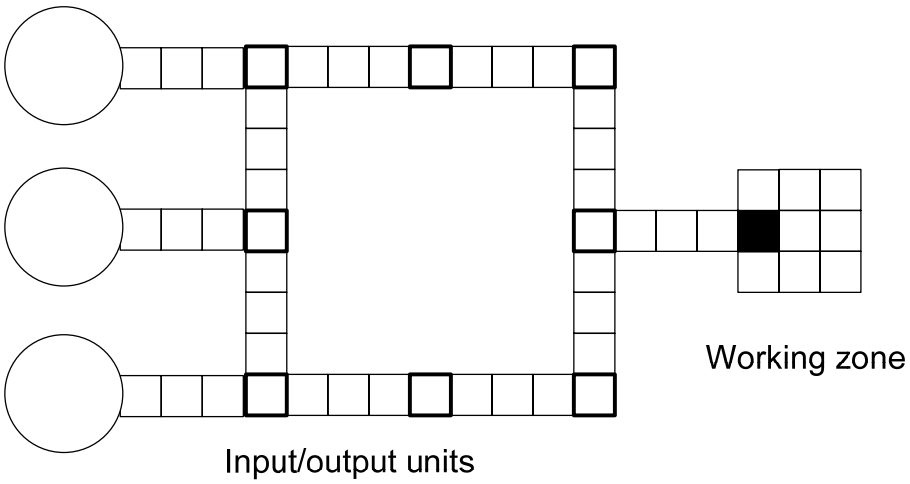


Figure 3.8: Connect the input and output units to the working zone of mixers and storage units through a ring so that all droplets to the working zone are input from and output to one electrode (filled in black). Here the working zone is a 3×3 mixer.

The size of mixers and storage units is $M \cdot S_{mix} + f(M) \cdot S_{store}$; we assume the number of electrodes for input/output units are determined by the analysis tree and therefore constant. Since the size of a mixer is bigger than that of a storage unit, we can retain one mixer and change all other mixers to be storage units. The transportation paths can still connect all mixers and storage units, and the size of mixers and storage units is now reduced to $1 \cdot S_{mix} + (M - 1 + f(M)) \cdot S_{store}$. The size of one mixer and $f(1)$ storage units is $1 \cdot S_{mix} + f(1) \cdot S_{store}$. From Corollary 2, $M + f(M) \geq 1 + f(1)$, which implies the chip with M mixers is larger than the chip with one mixer, leading to a contradiction. We conclude that even considering the transportation paths, the smallest chip is constructed using one mixer and $f(1)$ storage units. ■

3.3.4 Example

Consider a complete tree of depth 4. By Theorem 6, $f(1)$ for the complete tree is 3. Suppose that mixing and splitting droplets is performed in a mixer with 3 electrodes for droplets to move as shown in Figure 3.5 (a). Since we can place the mixer and storage units along the perimeter of the chip and overlap a subset of the surrounding electrodes, the size of a mixer and a storage unit can be smaller than

that shown in Figure 3.5. Figure 3.9 shows the layout of the smallest chip, without showing all input and output units.

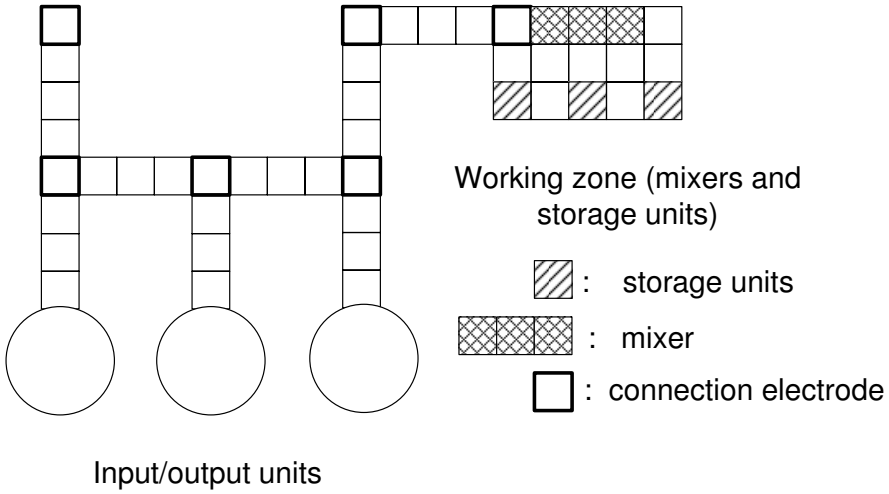


Figure 3.9: The partial layout of the smallest chip for a complete tree of depth 4.

3.4 Conclusion

In this chapter, we considered the problem of identifying the minimal resource requirements of biochemical analyses, towards the design of space-efficient biochips. We presented an algorithm to compute, for a given number of mixers, the minimum number of storage units for an input analysis using its tree structure, and designed a corresponding scheduling algorithm to perform the analysis. We defined the M-depth of the analysis tree to be the minimum number of storage units with M mixers. We characterized the variation of the M-depth of a tree with M, and used it to calculate the minimum total *size* (the number of electrodes) of mixers and storage units. We proved that we could always construct the smallest chip for an arbitrary analysis using one mixer and $f(1)$ storage units where $f(1)$ is the 1-depth of the biochemical analysis tree. Finally, we demonstrated our results on two example biochemical analyses and designed the smallest chip for a biochemical analysis with a complete tree structure of depth 4. These are the first results on the least resource requirements of DMFS for biochemical analyses, and can be used for the design and selection of chips for arbitrary biochemical analyses.

4. Conclusion

This thesis presents algorithms for an optimizing time and spatial resources on a digital microfluidic system. It achieves this by exploiting the tree structure of biochemical analyses. In this thesis, we focus on performing biochemical analyses in batch mode, where the goal is to produce one droplet of the final product.

4.1 Summary

In the first part of the thesis, we presented scheduling algorithms to optimize the completion time of biochemical analyses on a DMFS by exploiting the tree structure of these reactions. Using pipelining techniques, most input and output operations can be overlapped with mixing operations. So our goal is to complete the mixing operations as soon as possible with the given number of mixers while overlapping other operations with mixing. We introduced a full binary tree structure to model the biochemical analyses. We calculated the lower bound on the mixing completion time of a biochemical analysis and computed the minimum number of mixers to achieve this lower bound. We also designed an algorithm to schedule mixing operations for a given number of mixers, and proved it to be optimal in the measure of completion time. Finally, the resource constraint issue was also considered for two extreme cases of having just one mixer and having zero storage units. Both the case with identical mixing duration and that of different mixing durations were analyzed and corresponding algorithms were designed. These are the first results that use the analysis tree structure for optimal scheduling of biochemical analyses on DMFS.

In the second part of the thesis, we extended the resource constraint issues and focused on characterizing the resource requirements of arbitrary biochemical analyses on DMFS biochips from their tree structure. We designed a corresponding scheduling algorithm to perform the biochemical analyses using the least resource requirements. We also defined the M-depth of an analysis tree to describe such resource requirements and infer the variation in the number and size of resources

as a function of the tree structure and the number of mixers M. We can use these results to design the smallest biochip for any biochemical analysis and also determine whether a particular biochip can be used for a biochemical analysis. These are the first results on the least resource requirements of DMFS for biochemical analyses, and can be used for the design and selection of chips for arbitrary biochemical analyses.

The results in this thesis represent our initial steps in optimizing the automated operation and design of digital microfluidic biochips.

4.2 Future Work

Combining the scheduling algorithm with layout design and routing algorithms to form an integrated system is the next logical step. In our scheduling algorithms, we assume that the transportation time can be ignored due to the relatively high speed of droplet transportation. But when the algorithm is applied to a large-scale biochips, the long routing paths may significantly increase the transportation time. In that case, we need to consider optimization of the routing path to minimize the transportation time. In our future work, we will also use these results to explore the tradeoff between resource requirements and the completion time of biochemical analyses. Other extensions are to produce multiple droplets of final product and perform multiple simultaneous reactions in one chip. In the latter case, we need to consider several binary tree simultaneously, which means that greater parallelism may be exploited by carefully designing the scheduling algorithm. In this thesis, we only considered the batch mode operation of DMFS. In continuous mode, a large volume of final droplets are produced continuously, where a new objective (e.g., throughput) should be optimized instead of completion time. Also droplet transportation deadlocks, which can occur more easily since there will be many more droplets on the chip simultaneously, should be prevented by the routing algorithm. The ultimate goal of our future work is to develop efficient algorithms and a robust software platform for automated control of DMFS biochips.

LITERATURE CITED

[1] P. Dittrich and A. Manz, "Lab-on-a-chip: microfluidics in drug discovery," *Nature Reviews Drug Discovery*, vol. 5, no. 3, pp. 210–218, Mar. 2006.

[2] B. Zheng, C. Gerdts, and R. F. Ismagilov, "Using nanoliter plugs in microfluidics to facilitate and understand protein crystallization," *Current Opinion in Structural Biology*, vol. 15, pp. 548–555, 2005.

[3] X. X. Chen, H. K. Wu, C. D. Mao, and G. M. Whitesides, "A prototype two-dimensional capillary electrophoresis system fabricated in poly(dimenthylsiloxane)," *Anal. Chem.*, vol. 74, pp. 1772–1778, 2002.

[4] V. Srinivasan, V. K. Pamula, and R. B. Fair, "An integrated digital microfluidic lab-on-a-chip for clinical diagnostics on human physiological fluids," *Lab on a Chip*, vol. 5, no. 3, pp. 310–315, Mar. 2004.

[5] S. K. Cho, H. Moon, and C. Kim, "Creating, transporting, cutting, and merging liquid droplets by electrowetting-based actuation for digital microfluidic circuits," *J. Microelectromech. Syst.*, vol. 12, no. 1, pp. 70–80, Feb. 2003.

[6] M. G. Pollack, R. B. Fair, and A. D. Shenderov, "Electrowetting-based actuation of liquid droplets for microfluidic applications," *Appl. Phys. Lett.*, vol. 77, no. 11, pp. 1725–1726, Sept. 2000.

[7] P. Paik, V. K. Pamula, and R. B. Fair, "Rapid droplet mixers for digital microfluidic systems," *Lab on a chip*, vol. 3, pp. 253–259, Sept. 2003.

[8] R. B. Fair, V. Srinivasan, H. Ren, P. Paik, V. Pamula, and M. G. Pollack, "Electrowetting-based on-chip sample processing for integrated microfluidics," in *IEEE International Electron Devices Meeting (IEDM)*, 2003, pp. 779–782.

[9] M. G. Pollack, R. B. Fair, and A. D. Shenderov, "Electrowetting-based actuation of liquid droplets for microfluidic applications," *Applied Physics Letters*, vol. 77, pp. 1725–1726, 2000.

[10] T. B. Jones, "Liquid dielectrophoresis on the microscale," *Journal of Electrostatics*, vol. 51/52, pp. 290–299, 2001.

[11] J. Gong, S.-K. Fan, and C.-J. Kim, "Portable digital microfluidics platform with active but disposable lab-on-chip," in *Tech. Digest of 17th IEEE International Conference on Micro Electro Mechanical Systems (MEMS'04)*, Maastricht, The Netherlands, Jan. 2004, pp. 355–358.

[12] S.-K. Fan, C. Hashi, and C.-J. Kim, "Manipulation of multiple droplets on NxM grid by cross-reference EWOD driving scheme and pressure-contact packaging," in *IEEE Conference on MEMS*, Kyoto, Japan, Jan. 2003, pp. 694–697.

[13] J. Gong and C.-J. Kim, "Two-dimensional digital microfluidic system by multi-layer printed circuit board," in *Proceedings of IEEE MEMS 2005*, Miami, Florida, Jan. 2005, pp. 726–729.

[14] T. B. Jones, M. Gunji, M. Washizu, and M. J. Feldman, "Dielectrophoretic liquid actuation and nanodroplet formation," *Journal of Applied Physics*, vol. 89, pp. 1441–1448, Jan. 2001.

[15] P. Gascoyne, J. Schwartz, T. Anderson, D. Vykoukal, K. W. Current, C. McConaghy, F. Becker, and C. Andrews, "Dielectrophoresis-based programmable fluidic processors," *Lab on a Chip*, vol. 4, pp. 299–309, 2004.

[16] N. Manaresi, A. Romani, G. Medoro, L. Altomare, A. Leonardi, M. Tartagni, and R. Guerrieri, "A CMOS chip for individual cell manipulation and detection," *IEEE Journal of Solid-State Circuits*, vol. 38, no. 12, pp. 2297–2305, Dec. 2003.

[17] V. Srinivasan, V. Pamula, M. Pollack, and R. Fair, "A digital microfluidic

biosensor for multianalyte detection," in *IEEE 16th Annual International Conference on Micro Electro Mechanical Systems*, 2003, pp. 327–330.

[18] V. Srinivasan, V. K. Pamula, and R. B. Fair, "An integrated digital microfluidic lab-on-a-chip for clinical diagnostics on human physiological fluids," *Lab on a Chip*, vol. 4, pp. 310–315, 2004.

[19] M. G. Pollack, P. Y. Paik, A. D. Shenderov, V. K. Pamula, F. S. Dietrich, and R. B. Fair, "Investigation of electrowetting-based microfluidics for real-time PCR applications," in *Seventh International Conference on Miniaturized Chemical and Biochemical Analysis Systems (MicroTAS '03)*, Squaw Valley, CA, Oct. 2003.

[20] A. R. Wheeler, H. Moon, C.-J. C. Kim, J. A. Loo, and R. L. Garrell, "Electrowetting-based microfluidics for analysis of peptides and proteins by matrix-assisted laser desorption/ionization mass spectrometry," *Analytical Chemistry*, vol. 76, no. 16, pp. 4833–4838, Aug. 2004.

[21] J. Zeng, "Modeling and simulation of electrified droplets and its application to computer-aided design of digital microfluidics," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 2, pp. 224–233, Feb. 2006.

[22] J. Lienemann, A. Greiner, and J. G. Korvink, "Modeling, simulation, and optimization of electrowetting," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 2, pp. 234–247, Feb. 2006.

[23] F. Su and K. Chakrabarty, "Architectural-level synthesis of digital microfluidics-based biochips," in *Proc. IEEE International Conference on CAD*, 2004, pp. 223–228.

[24] ——, "Unified high-level synthesis and module placement for defect-tolerant microfluidic biochips," in *Proc. IEEE/ACM Design Automation Conference*, 2005, pp. 825–830.

[25] E. J. Griffith and S. Akella, "Coordinating multiple droplets in planar array digital microfluidic systems," *International Journal of Robotics Research*, vol. 24, no. 11, pp. 933–949, Nov. 2005.

[26] E. J. Griffith, S. Akella, and M. K. Goldberg, "Performance characterization of a reconfigurable planar array digital microfluidic system," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 2, pp. 340–352, Feb. 2006.

[27] F. Su and K. Chakrabarty, "Module placement for fault-tolerant microfluidics-based biochips," *ACM Transactions on Design Automation of Electronic Systems*, pp. 682–710, 2006.

[28] Y. Kwok and I. Ahmad, "Static scheduling algorithms for allocating directed task graphs to multiprocessors," *ACM Computing Surveys*, pp. 406–471, 1999.

[29] J. Ding, K. Chakrabarty, and R. B. Fair, "Scheduling of microfluidic operations for reconfigurable two-dimensional electrowetting arrays," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 2, pp. 329 – 339, Feb. 2006.

[30] K.-F. Böhringer, "Modeling and controlling parallel tasks in droplet-based microfluidic systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 12, pp. 1463–1468, Dec. 2001.

[31] F. Su, W. Hwang, and K. Chakrabarty, "Droplet routing in the synthesis of digital microfluidic biochips," in *Design, Automation and Test in Europe (DATE) Conference*, Munich, Germany, Mar. 2006, pp. 323–328.

[32] M. Gupta and S. Akella, "A scheduling and routing algorithm for digital microfluidic ring layouts with bus-phase addressing," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2007, pp. 3144–3150.

[33] L. Luo and S. Akella, "Optimal scheduling of biochemical analyses on digital

microfluidic systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Diego, CA, Oct. 2007, pp. 3151–3157.

[34] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. MIT Press and McGraw-Hill, 2001.

[35] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 3rd ed. Morgan Kaufmann, 2002.

[36] M. Pinedo, *Scheduling: Theory, Algorithms, and Systems.* Englewood Cliffs, NJ: Prentice-Hall, 1995.