SCHEDULING OF MIXTURE PREPARATION WITH MULTIPLE DEMAND USING DIGITAL MICROFLUIDICS BIOCHIPS

A DISSERTATION

Submitted in partial fulfilment of the requirements for the award of the degree

of

MASTER OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

SATENDRA KUMAR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY

ROORKEE – 247 667 (INDIA)

June 1, 2016

CANDIDATE'S DECLARATION

I hereby declare that the work presented in this dissertation "***Scheduling of Mixture Preparation with Multiple Demand Using Digital Microfluidics Biochips***" towards the fulfilment of the requirements for award of the degree of ***Master of Technology in Computer Science***, submitted to the Department of Computer Science and Engineering, ***Indian Institute of Technology-Roorkee, India*** is an authentic record of my own work carried out during June 2015 to May 2016 under the guidance of ***Dr. Sudip Roy***, Assistant Professor, Department of Computer Science and Engineering, Indian Institute of Technology Roorkee.

The content presented in this dissertation has not been submitted by me for the award of any other degree of this or any other institute.

Date:
Place: Roorkee

(Satendra Kumar)

CERTIFICATE

This is to certify that the statement made by the candidate in declaration is correct to the best of my knowledge and belief.

Date:
Place: Roorkee

(**Dr. Sudip Roy**)
Assistant Professor
Department of Computer Science and Engineering
Indian Institute of Technology, Roorkee

Abstract

Microfluidics biochips are revolutionary devices in the field of clinical diagnostics, DNA analysis and molecular biology. Biochips involve various fields of science and engineering i.e., physics, chemistry, biochemistry, nanotechnology, fabrication technology and computer science. It uses small volume of fluids on scale of micro to nano liter for automatically carrying out the reactions needed for some biochemical assays. In this report I have covered basic overview of biochips, existing algorithms for sample preparation specifically for dilution, mixing, and multiple droplet of single target. Some application requires perticular sample repeatedy at the time of asssy execution. So fast and efficient sample preparation process is require to generate multiple droplets of sample. Three approaches are presented in this report to reduce the time to generate stream of droplets for bioassay having high demand of sample at the time of assay execution. Simulation results show that new approach is quite promising as compare to existing MMS and SRS algorithms to reduce the time taken to prepare sample for such high demand.

First algorithm proposed, called KMS or K-Mixer Scheduling, utilizes K-Droplet Mixer[3] to schedule mixing tree for multiple demand of single target generation. This algorithm reduces total mix-split steps, by 74.6% than MMS and SRS but compromises with storage requirement by 18.5% than SRS but still good by 25.1% than MMS. To reduce the storage requirement modified KMS (m-KMS) schedules mixing tree more then once with fraction f of total required demand D, fraction f used to balance the total storage requirement U and total mixing operation T_{ms}. This new modified algorithm reduces total mix-split steps, by 79% and storage requirement by 80.2%, 67.7% than MMS, SRS respectively. And finally we presented KMS for mixing graph which used to schedule mixing graph. This algorithm reduces total mix-split steps by 76% than MMS and SRS, storage requirement by 79.6%, 66.5% than MMS and SRS respectively.

Dedication

To mum and dad

Acknowledgements

I would never have been able to complete my dissertation without the guidance of my supervisor, and support from my family and loved ones.

I would like to express a thankful note to my supervisor, **Dr. Sudip Roy**, for his guidance. I am also grateful to the Department of Computer Science of IIT-Roorkee for providing valuable resources to aid my research.

I would like to thank all the **lab colleagues** who were good friends, and were always willing to help and give their best suggestions.

A hearty thanks to **my parents and siblings** for encouraging me, in good times and bad.

Contents

List of Figures

List of Abbreviation

Abbreviation	**Description**
CMF	Continuous Flow Microfluidics
DMF	Digital Microfluidics
EWOD	Electrowetting On Dielectric
RMA	Ratio-ed Mixing Algorithm
MTCS	Mixing Tree with Common Subtree
CoDOS	Common Dilution Operation Sharing
KMS	K-Mixer Scheduling
KMS-m	K-Mixer Scheduling Modified
KMS-g	K-Mixer Scheduling for Graph

List of Symbols

Symbol	**Description**
K	Capacity of mixer
C_t	Concentration of target sample
T_{ms}	Total mix-split steps
W	Total waste droplets generated
I	Total input droplets required
U	Total storage unit required
d	Desired accuracy level of target sample
D	Total demand of droplets required
L	Ratio-sum of target sample

CHAPTER 1

Introduction

Microfluidics biochips are emerging devices in fields of microfluidics which are used to precisely control and manipulate small volume of fluids. These chips have potential application in analysis of biochemical assay(polymerase chain reaction, proteomics), biological computing, health care (Lab-on-Chip technology), high throughput DNA sequencing etc. These chips has many advantages over traditional devices such as small chip size (typically of few centimeters [4]), on chip sample preparation, small amount of fluid requirement (scale of micro to milli liter), high speed analysis of results with high accuracy, and less cost in production of chip[3].

Earlier stage of microfluidics biochips were based on continuous flow in which mixing, transportation of fluids are carried out through micro channels. In such biochips concurrent executions of assays are not possible due to limitation on numbers stationary rotatory mixture [4]. Other types of biochips which uses Electrowetting on Dielectrics (EWOD) principle for performing various operations i.e., transporting, mixing etc are called Digital Microfluidics Biochips(DMFB). Digital microfluidics (DMF) biochips uses discrete droplets and control each droplets individually which makes parallel execution possible. Parallel execution and flexible architecture makes DMFB more promising technology then others.

Various algorithm have been published in recent years to solve problem related to DMFB, involving transportation, Dilution, Mixing, Module Placement etc. Sample preparation is one crucial step in assay execution which requires dilution or mixing of reagents. Diluting a given sample to specific concentration is one of the major problem which requires repeated mixing of reagent with buffer in well defined order. Whereas Mixing process produce mixture of more then two reagent having concentration in given ratio. Many algorithm have been published to efficiently produce dilution and mixture of reagents.

Some application requires particular sample repeatedly at the time of assay execution. So fast and efficient sample preparation process is require to generate multiple droplets of sample as demanded. There are two existing algorithm MMS and SRS[17], utilizes 1: 1 mixing model, performs same mixing operation multiple times to fulfill required demand. We propose a new K-Mixer Scheduling which uses K-droplet mixer to fasten the process of droplet generation. Experimental result show that modified version of KMS improves mixing step, storage count, input requirement and wastage significantly to the previous multiple demand of single target generation algorithms i.e., SRS and MMS.

1.1 Motivation and Objective

Time of sample preparation in biochemical application is not directly depends on the volume of the sample for example time to prepare mixture of four droplets in one mixing step would be much less than the time, if droplets are produced in two mixing step each with two droplets. Existing MMS and SRS algorithm prepare larger demand of a sample by repeating two droplets mixing which consumes more time. The idea of proposed algorithm is to provide flexibility in volume of sample to be prepared using $n{:}\,n$ mixer where n is variable.

MMS and SRS algorithm proposed for droplet streaming uses 1: 1 mixing model which is capable of mixing two droplet in single mix step, to produce demand more then two it schedule internal nodes of mixing tree repeatedly. K-Mixer Scheduling (KMS) utilizes K-droplet mixer[3] as shown in Fig. 1.1 to mix up to K droplet simultaneously. It utilizes $n{:}\,n$ mixing model, which is capable of mixing $k = 2i, 1 \leq i \leq \lfloor \frac{K}{2} \rfloor$ in single mix step to reduce the no of mixing steps needed to generate larger demand. After a mixing operation mixer can serve as reservoir containing mixed sample and desired number of droplets can be dispensed by normal dispensing process.

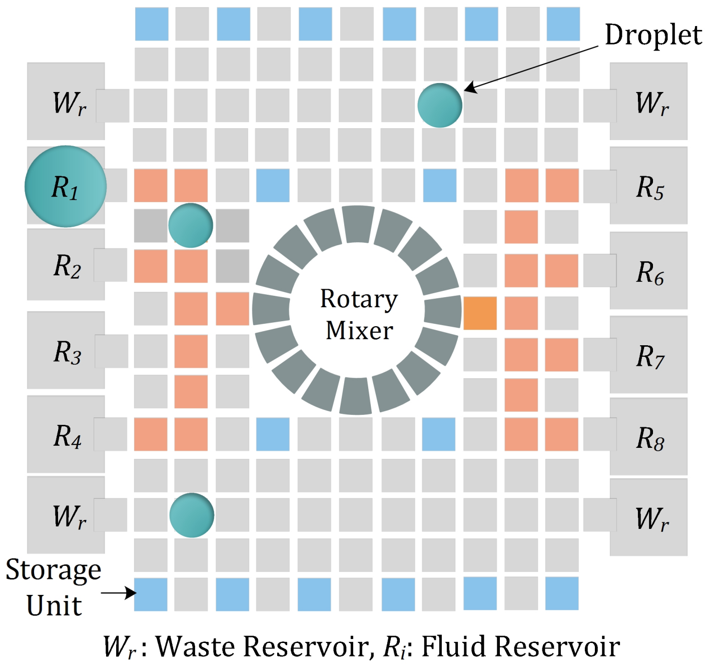


Figure 1.1: K-droplet rotary mixer.

Mixing tree produced by mixing algorithm consist of leaf node, represents input reagent and the intermediate node, represents intermediate sample. If 1: 1 mixer is used then each intermediate node require to schedule to some mixer to produce two droplets of target ratio. But if demand is more intermediate node requires to schedule more then once which results in more number of mix-split steps which increases the time of sample preparation.

K-droplet rotary mixer can be used in-place of 1: 1 mixer which can mix up to K droplets in a single mix step. For given K, a number of non homomorphic tree can be scheduled in single mix step. For example a 4-droplet mixer can schedule any of these three tree shown in fig. 1.2 in single mix step.

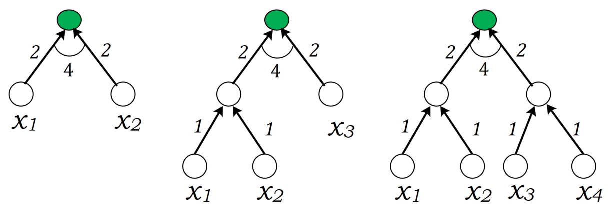


Figure 1.2: Mixing trees that can be scheduled on 4-droplet mixer.

1.2 Contribution of the Dissertation

This dissertation presented three algorithm for generating Multiple Droplet of Single Target (MDST) effeciently in term of total mix-split operation, storage utilization, input count, and wastage. Two of these algorithm are for mixing tree and the last one produced MDST for mixing graph. Following sections provide brief overview of these three algorithm.

1.2.1 KMS for Mixing Tree

The idea of *K*-Mixer Scheduling (KMS) algorithm is to provide flexibility in volume of sample to be prepared in one mix step, using $n : n$ mixer where n is variable. To reduce mix-split steps, K-Mixer Scheduling(KMS) utilizes K-droplet mixer described in section 3.1 in place of 1: 1. This algorithm scheduled mixing tree generate using existing mixing algorithm, reduces total number of mixing steps needed.

1.2.2 KMS-m for Mixing Tree

KMS for mixing tree reduces mixing steps but it compromises with storage need during sample preparation. So modified version of KMS, named KMS-m presented in this thesis to reduce storage utilization as well as total mixing step. This algorithm schedules mixing tree with a fraction of total demand repeatedly which significantly reduces storage utilization.

1.2.3 KMS-g for Mixing Graph

Above two algorithm schedules mixing tree but there are some mixing algorithm which produce mixing graph consist of sharable node pair. So KMS for mixing graph, named (KMS-g) takes graph generated by any of mixing graph producing algorithm and schedule it to produce multiple demand of target ratio.

1.3 Organization of Dissertation

This dissertation has been divided in chapters.

Chapter 1 provides introduction of microfluidics biochips with its application in different field and provides motivation for research in field of digital microfluidics.

Chapter 2 describes architecture and working principle of different biochip with detailed explanation of various concept related to DMF biochips, this chapter also presented brief explanation of various sample preparation algorithm.

Chapter 3, 4, 5 presented KMS, KMS-m, and KMS-g algorithms with details explanation, and simulation result with existing algorithm.

chapter 6 provide conclusion on all tree algorithm combined and discussed future research work.

chapter 6 provide the dissemination form dissertation.

CHAPTER 2

Background and Literature Review

Microfluidic (DMF) biochips have become more promising and emerging technology in field of health care as these device can be used as point of care diagnosis. A large amount of research have been done in design and development of these bio-chips. Beside these hardware issues, there are many software related problem required to tackle in order to effectively utilize these system, so many researchers of computer science domain got huge interest to solve these problem. Many algorithm have been proposed in recent years. These microfluidic biochips are software programmable and can be used as a device for parallel execution of bioprotocols, such as real-time bio-molecular detection, and automated drug discovery.

There are different types of Biochip available, out of those Continuous Flow Microfluidics (CMF) and Digital Microfluidics (DMF) are more popular and widely used biochips. In recent years lots of research have been done in these type of bio-chips.

2.1 Continuous-Flow Biochips

Continuous-flow microfluidic biochips contain μ channels through which fluids in scale of μ *liter* volume can follow. μ valves are used to guide the flow of fluids through various micro channels. To carry out some reaction, first step is to fill rotatory mixture by more than one fluids each separated by μ valves and then fluid inside the mixer is mixed by actuating the μ pumps. Permanently attached microstructures also lead to limited reconfigurability. Flow layer shown in Fig. 2.1 consist of μ channel, μ valves and mixture module[20]. Wastage module is used for throwing extra fluid remained after mixing operation. And storage cells are used to store fluids and are attached to external input as well as to mixture. There is another layer named control layer which is used to guide flow of liquid by changing μ valves state i.e., on/off and is controlled by external microcontroller.

2.2 Digital Microfluidics Biocips

DMF Biochips places discrete droplets between two plates and uses principle of electrowetting-on-dielectric effect (EWOD)[14] to drive them on chip. This effect is used to carry out operations such as dispensing, droplet transport, merge and split by applying voltage to the electrode attached below the chip plate. Since each droplet can be controlled individually DMF Biochips are capable of dynamically reconfigure droplet movement during execution of multiple bioassay concurrently. Due to recofigurability this architecture can be programmed and can be used as a device for parallel bioassay operation, which makes it better then other type of biochips. DMF Biochip provides multiple on-chip operation such as dispensing, mixing, splitting, detection, transport in order to perform assay.

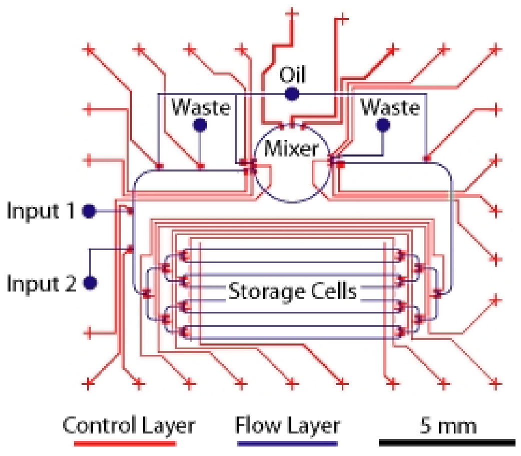


Figure 2.1: Layout of CMFB. [20]

2.2.1 Basic Layout of DMF Biochip

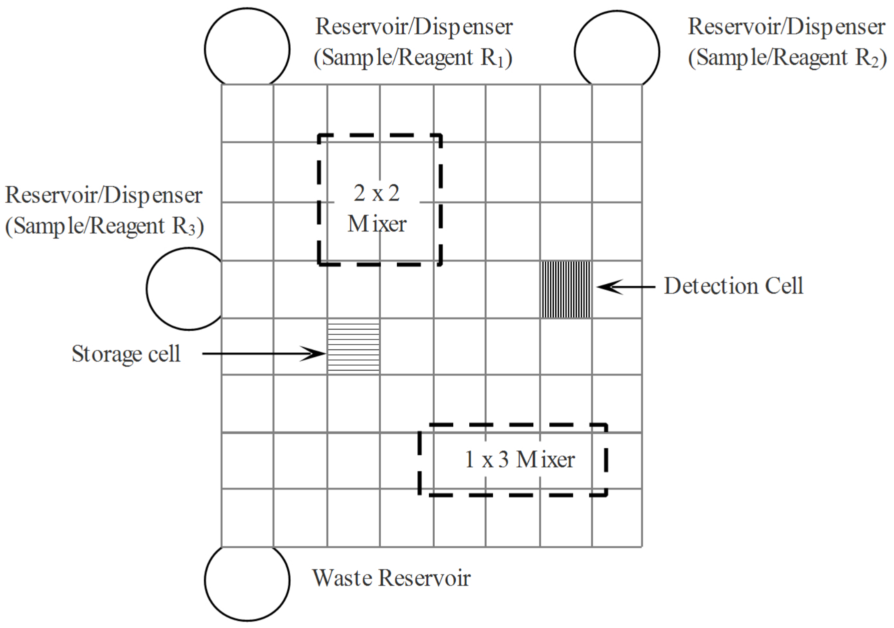

Reservoir/Dispenser (Sample/Reagent R₁)
Reservoir/Dispenser (Sample/Reagent R₂)
Reservoir/Dispenser (Sample/Reagent R₃)
2 x 2 Mixer
Detection Cell
Storage cell
1 x 3 Mixer
Waste Reservoir

Figure 2.2: Architecture of basic DMF Biochip having two 1: 1 mixer of different shape.

An basic layout of DMF biochip is shown in Fig. 2.2 where each square of two-dimensional plate is attached to an electrode which is controlled by an external microcontroller. Mixing module can be of different shape, two of such modules (1×3) and (2×2) mixer/splitters are marked by doted rectangle. The placement (locations) of the mixers can be reconfigured dynamically on chip according to availability of free electrodes with respect to time. Three reservoirs (circled electrode) are shown at the periphery of the DMF biochip. Reagent R_1, R_2 and R_2 are placed in these reservoir/dispenser. Reagent droplets can be dispensed from the respective reservoirs into the DMF chip. A waste reservoir is used to collect the discarded droplets in process of assay execution. Any free cell on chip can be used as the storage unit. A detection unit is used for result inspection. Droplet after assay execution can be transported to these detection unit for inspection.

Cross section view of DMF biochip is shown in Fig. 2.3. It consists of two plate and fluid droplet which is sandwiched between these plates. The bottom plate is attached with a two dimensional array of electrodes which is controlled individually by external microcontroller, and the top plate is attached with a ground electrode. Electrodes are made of indium tin oxide (ITO). A dielectric insulator (parylene C) coated with hydrophobic layer (Teflon AF) is placed on top and bottom plate to decrease the wetting of the surface by fluid [15].

2.2.2 Principle of Droplet Movement

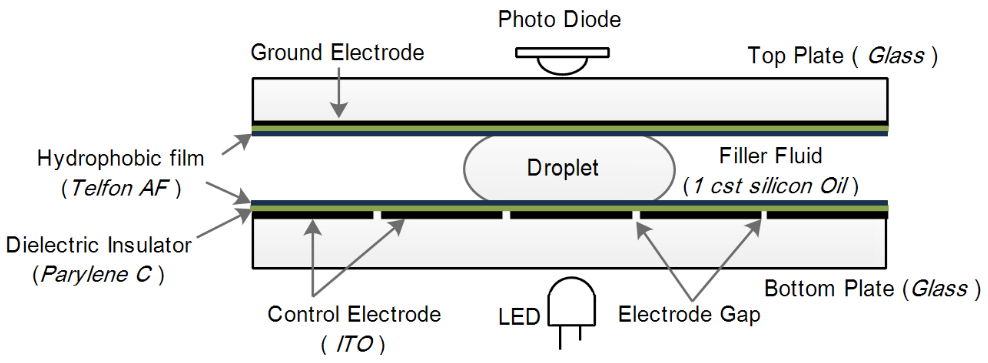


Figure 2.3: Droplet sandwiched between two plate.

The digital microfluidic biochips are based on the manipulation of nanoliter droplets using the principle of electrowetting on dielectric effect. When Droplet placed on electrode e_1 without any voltage its contact angle formed by plate, fluid surface and air remain α as shown in Fig. 2.4 and when e_1 is set on high voltage contact angle changed to β [15]. As shown in Fig. 2.4 the volume of droplet should be sufficient enough to slightly overlap the adjacent electrodes. The velocity of the droplet can be controlled by adjusting the control voltage (0 90V)[5].

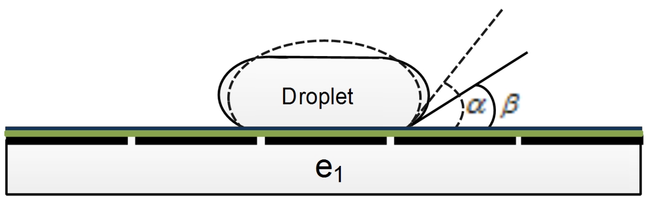


Figure 2.4: Angle changed form α to β on activating electrode e_1.

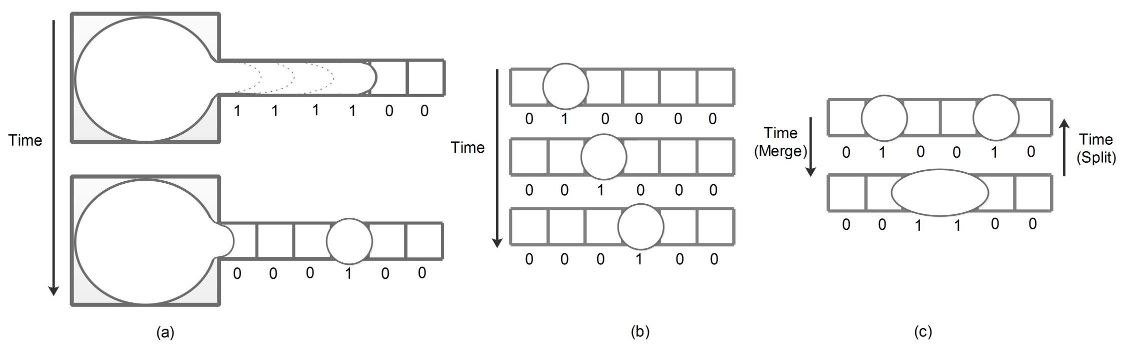


Figure 2.5: Basic operation on fluid droplets.

2.2.3 Principle of Droplet Movement

2.2.4 Basic Operation on Fluid Droplets

Droplet creation (dispensing), transportation, merges, split are the fundamental operation carried out on fluid droplets on DMF Biochip Fig. 2.5 shows a particular state of these operations.

2.2.4.1 Dispensing

Dispensing is the process of creating droplet form the reservoir. Fig. 2.5(a) show 3-electrode pinch-off technique of dispensing one droplet form reservoir. First step is to stretched out the droplet from the reservoir on the four electrode by applying actuation sequence 11110 as shown in figure and then it deactivates the middle three electrodes i.e.,(actuation sequence 00010) in next cycle. This process creates one droplet of unit volume using four electrodes.

2.2.4.2 Merging

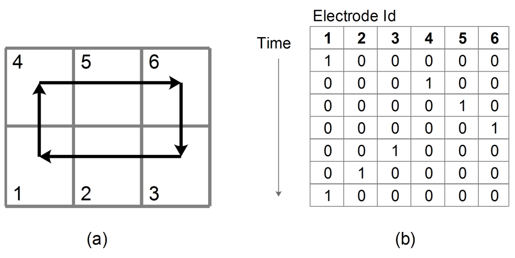
It is the operation of merging two droplet of unit volume to create new droplet of two unit volume. When two droplets moved to a single electrode it gets merged as shown in Fig. 2.5(c).

2.2.4.3 Splitting

This is the process of creating two droplets of unit volume form a droplet(mixture) of two unit volume. Fig. 2.5(c) illustrates the process of spiting on droplet of two unit volume which divides in two droplets when actuation sequence corresponding to electrodes changed to 010010 form 001100.

2.2.5 Actuation Sequence

To perform various operation corresponding to bioassay, a sequence of change in voltage level of each electrodes of chip required and this sequence is achieved by external microcontroller attached to chip. Fig. 2.6(a) show the actuation sequance corrosponding to droplet movement is shown Fig. 2.6(a).



1	2	3	4	5	6
1	0	0	0	0	0
0	0	0	1	0	0
0	0	0	0	1	0
0	0	0	0	0	1
0	0	1	0	0	0
0	1	0	0	0	0
1	0	0	0	0	0

Figure 2.6: (a) droplet movement (b) actuation sequence corresponding to droplet movement.

2.3 Automated Sample Preparation

Bioassay protocols are implemented on biochip and these protocol may require pre-processing of input reagents e.g., a reagent with particular concentration or a mixture having more than two reactants in some specified ratio. Either these samples should be prepared outside the chip and can be provided as input to biochip during assay execution or it can to be prepared on chip automatically during execution. Outside sample preparation may take time which is undesirable in most of biochemical assay such as in clinical diagnosis. So it is desirable to provide automatic sample preparation on chip.

2.3.1 Basic Concepts of Sample Preparation on DMF Biochips

2.3.1.1 Mixing Models

DMF Biochip dispenses droplets of unit volume and these droplet move on a uniform 2-dimensional array of electrodes. DMF Biochip uses any one of mixing model of $m{:}\,n$ form where m droplets of one reagent mixed with n droplet of other reagents in a single operation and generates $m+n$ unit of mixed solution. Droplet can be mixed in following three different mixing models.

1. $m = n = 1$
2. $m = n \neq 1$
3. $m \neq n$

2.3.1.2 Mixing Tree

To guide the process of assay execution a well defined steps are required and these steps are represented by tree or graph. In a mixing tree, each leaf node corresponds to a reagent and internal node represents mix-split step. Fig. 2.7 represent a mixing tree to generate sample of ratio.

2.4 Literature Review

Mixing and Dilution are two fundamental operations of biological sample preparation, a well-defined algorithms are required to map these operations on DMF Biochip having some practical limitation such as balanced mixing and splitting. In last few

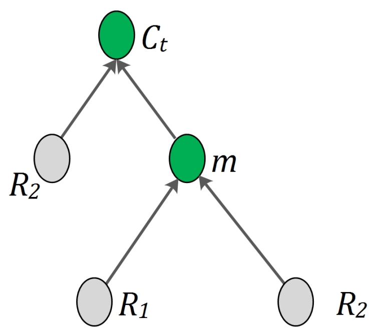


Figure 2.7: $(R_1, R_2), m, C_t$ are reagents, intermediate, and target sample respectively.

year a number of different algorithm have been proposed by researcher to automate and optimize sample preparation process. First promising algorithm for dilution and mixing proposed in 2008 2-way mix and min-mix [20] respectively. These both algorithms are based on scanning binary representation of concentration of the constituent reagents of required solution. Based on desired accuracy level, 2-way mix dilutes a given reactant to a target concentration in minimum number of mix-split operation than any other method. In spite of giving lower bound on number of mix-split count this method results in significant amount of wastage of input and intermediate reagents.

2.4.1 Bit Scanning (2-Way Mix) Algorithm for Dilution

Bit Scanning[20] approach of dilution is guided by binary string of target concentration. The dilution process of a reagent with buffer can be represented using a "dilution tree" as shown in Fig. 2.8 where leaf node of a tree represents a reagent or buffer, and each internal node represents mixture of its two children. Concentration of the mixture of two reagents (using 1:1 mixing model) can be calculated using arithmetic mean of concentration of its children. If concentrations of children are c_1 and c_2 then concentration of its parent will be $\frac{c_1+c_2}{2}$ for e.g., $n = 10, C_t = 0.59211$ find integer x such that $\frac{x}{2^n} = 0.59211$, $x = 607$ and now $C_t = \frac{607}{1024} = .1001011111_2$ dilution tree for this concentration is shown in Fig. 2.8. Process starts by scanning bits in binary string from right to left if 1 occurs intermediate sample is mixed with reagent otherwise it mixes with buffer. Process terminates in n steps.

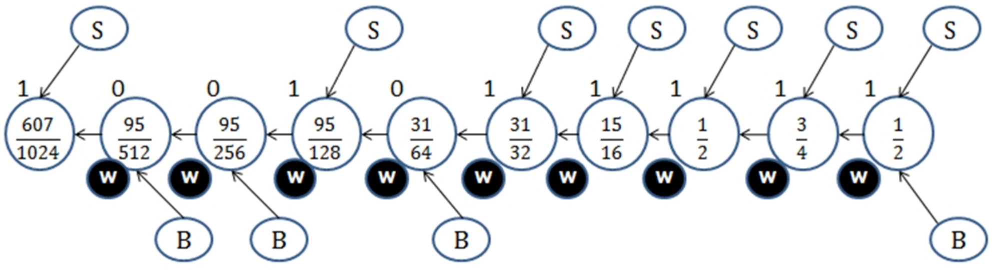


Figure 2.8: Dilution Tree for $C_t = 0.59211$.

This algorithm directly utilizes the pattern of bit string to guide the dilution process so it is attractive strategy from hardware design point of view and also it is simple to implement. No of sample and buffer droplets required for sample of concentration $C_t = 0.59211$ shown in Fig. 2.8 are 7 and 4 respectively and number of wasted droplets are 9.

2.4.2 Min-Mix Algorithm for Mixing

MinMix [20] strategy is based on scanning of the binary representation of the concentration values of each fluid. The idea of the algorithm is to place a leaf node with reagent at depth d in the mixing tree if there is 1 in its binary representation at $n - d$ where n is total length binary string. A mixing tree for 5 reagents A, B, C, D, E with ratio 5: 4: 7: 6: 7 and required accuracy of up to six precision bit in binary representation is shown in Fig.2.9 with bit representation of these ratio $\frac{5}{32}, \frac{4}{32}, \frac{7}{32}, \frac{6}{32}, \frac{7}{32}$. Starting from least significant bit, it mixes reagents having 1 in its binary representation,i.e., A, C, E, F in this example which produces two intermediate droplets which further mixed with sample having 1 in next significant bit which are C, D, E, F this process continues until single target droplets is not produced. Complete mixing tree for above mention example is shown in Fig. 2.9.

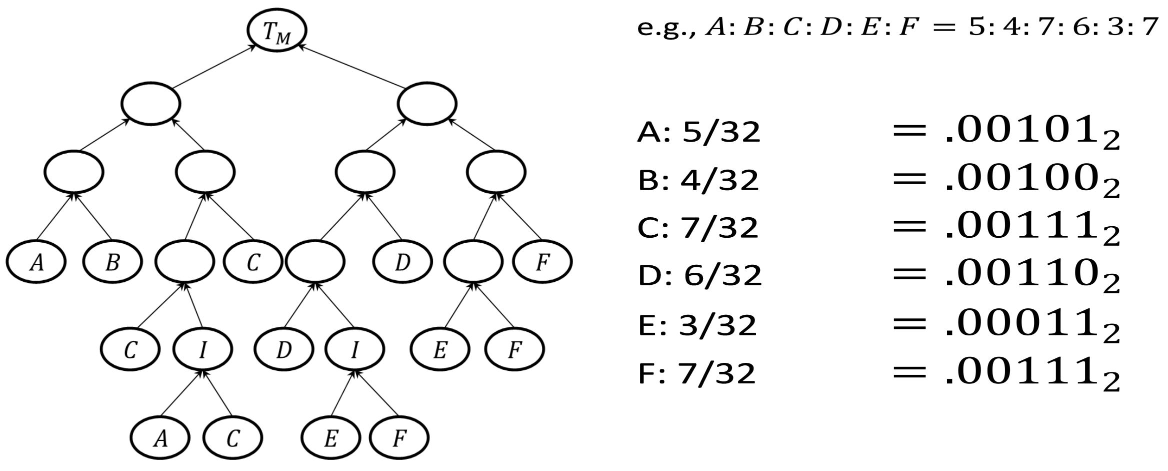


Figure 2.9: Mixing Tree for 5: 4: 7: 6: 7 and binary sequence each

2.4.3 Other Dilution Algorithms in Brief Details

1. ***Dilution and Mixing with Reduced Wastage(DMRW)***[16] algorithm provides a significant reduction in intermediate wastage. This algorithm modeled dilution problem (i.e., diluting the given input reagent to a required concentration) as searching the target concentration in the search space having lower and upper bound as the concentration of buffer and input reagent. This algorithm utilizes binary search strategy that iteratively reduces the search space by half to find target concentration and approaches to target concentration. In some cases this algorithm generates skewed mixing graph those tree results in worst performance.
2. To tackle these skewed graphs an ***Improved Dilution and Mixing Algorithm(IDMA)***[19] was suggested in 2011. With IDMA an integrated scheme was also proposed for selecting most efficient algorithm among 2-WayMix, DMRW and IDMA with a new n: n architecture to alleviate the increase in number of mixing operation in IDMA and DMRW.

Inputs (i.e., reagent and buffer) may not have same cost so reduction in wastage does not guarantees reduction in most precious input that may be either physiological sample or reagent.
3. ***REactant MInimization Algorithm (REMIA)***[10] selectively reduces the most precious reagent. REMIA generates the dilution tree in two phase named interpolated dilution phase and exponential dilution phase. Droplets generated in exponential dilution phase named as prime concentration values (PCV). This algorithm applied exponential dilution on most precious reagent to generate intermediate reagents to be used by exponential dilution phase. REMIA algorithm further reduced the amount of reactant wastage if it is used in generating multi-target sample preparation.
4. An ***Optimal Sample Preparation Algorithm***[6] based on mincost-maxflow approach was proposed in 2014. It generate dilution graph which optimizes sample and buffer uses. This algorithm formulate dilution problem as a network flow model and transform it to an Integer equal flow problem which is solved using Integer Linear Programming. This process takes the global view of the graph and optimizes cost function which represents the practical cost of sample and buffer. And finally construct optimize dilution graph to be used to generate target demand with optimal number of droplet and buffer. This algorithm provides flexibility to assign weight to sample and buffer in the cost function to optimize more precious reactant moreover by providing equal sample and buffer waste can be optimized. This algorithm can also be extends to optimize cost function in multi-target sample preparation.
5. Aforementioned sample preparation algorithms generally developed for DMF biochip having limitation of 1: 1 mixing model can also be used in CMF Biochip by limiting the segment count to two. Various mixing model can be achieved instead of having only 1: 1 mixing model by having more than two segments in CMF biochip. ***Tree Pruning and Grafting Algorithm (TPG)***[13] proposed in 2015 is the first algorithm dedicated to CMF Biochip, tries to achieve more optimization in input and wastage by utilization the various mixing model. This algorithm takes tree generated from any tree based algorithm (i.e., either from 2-WayMix or from REMIA) as input and applies various tree transformation operation and generates new dilution graph having various type of mixing operation provided by n segment in mixer. In first it applies tree pruning to input tree to generated blended tree having all leaf nodes as PCVs, which is well pruned tree utilizes all mixing model provided by architecture. In second step tree grafting process is applied to blended tree generated by tree pruning, which generates reactant-minimal blended tree. And finally apply reactant sharing step which exploits possibility of reactant sharing. TPG outperform state-of-art method those developed mainly focused for 1:1 model.
6. A ***Volume Oriented Sample Preparation Algorithm (VOSPA)***[9] for CMF Biochip having multi segments mixer that enables segment based intermediate solution reuse for better reactant Minimization. This algorithm mixes several solutions of a reactant having different concentration each corresponding to different segment only in single mix operation. A number of mixing model can be achieved by a multi-segment mixer which is used by this algorithm as by TPG to provide better reactant minimization. It tries to achieve the target sample volume by filling segment of mixer one by one. It utilizes the CV bank to keep track of available intermediate solution those are previously generated and reuses them to reduce input

reactant consumption. This algorithm has two process first the master process tries to utilizes existing intermediate solutions and accumulate it to the segment of mixer and latter one is subsidiary process which is responsible for producing appropriate intermediate solution for the use of the master process. Performance of VOSPA with four segments outperform BS, REMIA and network flow based approaches moreover by increasing the number of segment performance can further be improved.

2.4.4 Other Mixing Algorithms in Brief Details

1. After MinMix a ***Ratio-ed Mixing Algorithm (RMA)***[18] was proposed in 2011 which is based on fractional decomposition of the algebraic expression of target ratio, which is used to build mixing tree. Fractional decomposition can be represented by a disjoint mixing tree. This algorithm provides a layout aware mixing tree which helps in assigning boundary reservoirs to input fluids in such a way that droplet crossovers and transportation distances are reduced. This method reduces the droplet transportation time from boundary reservoirs to mixers and also avoids cross-contamination in routing path of droplet due to disjoint mixing.
2. In order to minimize the amount of reactants and waste, ***Intermediate Droplet Sharing Algorithm (IDSA)*** [7] was proposed which share the intermediate droplet among multiple targets many-reactant sample preparation. It utilizes both the intermediate droplets obtained after a split operation when a pair of identical subtrees is identified under permutation of leaf nodes at the same level of tree. This algorithm reduces the total number of mix-split steps, waste droplets; however, its time complexity significantly increases when number of different types of reactant increases.
3. ***Reagent-Saving Mixing Algorithm (RSMA)*** [8] is another multi-reactant sample preparation algorithm which provides reagent-saving approach to concurrently generate multiple target concentration. This algorithm also reduces waste droplet and sample preparation time for multiple target droplets as compare to serial sample preparation algorithms such as MinMix and RMA. This algorithm first decomposes the target ratios and then construct mixing graph. The first stage generates the best possible decomposition of the target ratios, and the mixing graph generated in second stage represents the sequence of mixing steps.
4. ***Common Dilution Operation Sharing Algorithm (CoDOS)*** [12] is a mixing algorithm which explore node sharing within a single mixing tree unlike the RSMA which applies droplet sharing among a set of mixing trees to reduce input and waste consumption. If number of target ratios are limited then performance of RSMA will get reduced because of less sharing node availability. Droplet sharing for reactant minimization is achieved by finding sharable node pairs (i.e., nodes having same concentration) in a mixing graph. CoDOS try to create new sharable node by swapping pair of node in same level of tree. After applying optimization on local graph using CoDOS, a global node sharing algorithm can be applied to achieve better optimization.

2.4.5 Scheduling Algorithm

Efficient and waste reduced generation of multiple droplet of same ratio (mixing) and of same concentration (dilution) is refers to Multiple Droplets of Single Target

(MDST) generation. There are two algorithm proposed in 2014 to achieve MDST. These algorithm takes mixing tree as input from any of existing mixing algorithm and repeatedly schedules it to achieve required demand.

2.4.5.1 M-Mixer Scheduling (MMS)

MMS[17] takes mixing tree generated using any state of art mixing algorithm and generates mixing forest in-order to utilize wastage produced at intermediate nodes. Then it schedules mixing forest on given number of mixers in bottom-up manner. It identified set of mix-split (non-leaf) nodes of a mixing forest as the schedulable nodes, if they were not scheduled earlier and are now ready to be performed at the same time-cycle. The schedulable nodes can be processed concurrently, if a sufficient number of mixers are available at that time-cycle. It maintains a queue to enqueue all the schedulable nodes in a level-wise bottom-up fashion, and nodes equals to number of mixer or fewer nodes are dequeued from queue to assign the available mixers at time-cycle. When all the levels are examined and the queue is still non-empty, the remaining nodes are dequeued to assign mixers in next time-cycles without enqueuing any new nodes.

2.4.5.2 Storage Reduced Scheduling (SRS)

SRS[17] prioritizes the scheduling of non-leaf nodes in the mixing forest based on two factors: if the mixing is stalled at a node, then (i) how it affects the total storage requirement, (ii) What is its impact on the time of completion of the mixing forest. It prioritizes the internal nodes having more non leaf children over the schedulable node having less number of leaf children. It cost no storage units to stalls a node having both children as leaf node, per time-cycle. This is because a leaf node indicates direct input from the fluid reservoir and it does not require any on-chip storage unit. Hence, the priority should be given to the internal nodes whose at least one child node is an internal node over the internal nodes whose both the children nodes are leaf nodes.

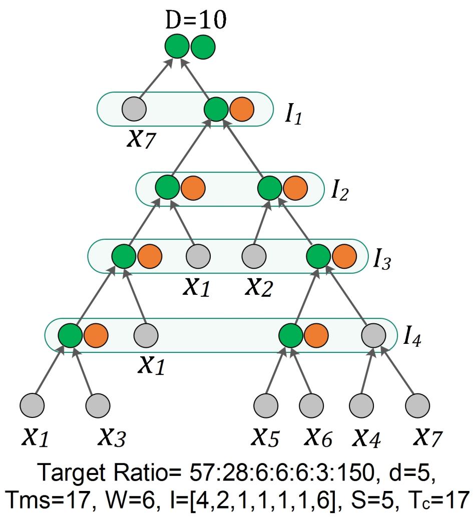

D=10
I_1
x_7
I_2
I_3
x_1 x_2
I_4
x_1
x_1 x_3 x_5 x_6 x_4 x_7
Target Ratio= 57:28:6:6:6:3:150, d=5,
Tms=17, W=6, I=[4,2,1,1,1,1,6], S=5, T_c=17

Figure 2.10: Mixing tree produced by RMA.

SRS method utilizes the extra droplet generated at intermendiate node to produce demand more then two. Extra droplet generated at intermediate node are shown by orange color in Fig. 2.10. It generates different tree to schedule extra droplets, corrosponding to intermediate level represented by I_n where n is depth of tree as shown in Fig. 2.10. Scheduling tree corrosponding to intermediate level I_n generats $2 * d_{n-1}$ target droplet where d_n number of target droplets produced when tree corrosponding to I_n scheduled. For e.g., tree corrosponding to I_2 shown by produces 2 target droplets and d_2 corrosponding to I_2 is $I_1 * 2$ i.e., 4. Example: For Minipreparation protocol [2] requires 7 reagent with ratio 57: 28: 6: 6: 6: 3: 150 scheduled using SRS on mixing tree generated by RMA shown in Fig. 2.10. When this tree scheduled by SRS for generating demand $D = 10$, consumes 16 input droplet, requires 17 mix-split steps and 5 storage unit.

CHAPTER 3

KMS for Mixing Tree

In many biological protocols, such as PCR (polymerase chain reaction), a mixture of fluids in a given ratio is required repeatedly, and hence an efficient algorithm to generate mixture in desired demand is required for assay completion. Existing MDST algorithms for digital microfluidics (DMF) biochip uses 1: 1 mixing model to generate demand of mixture by repeatedly scheduling the mixing tree which increases costly mix-split steps. The idea of KMS algorithm is to provide flexibility in volume of sample to be prepared, using $n : n$ mixer where n is variable. To reduce mix-split steps, K-Mixer Scheduling(KMS) utilizes K-droplet mixer described in section 3.1 in place of 1: 1.

3.1 K-Droplet Rotary Mixer

Time of sample preparation in biochemical application is not directly depends on the volume of the sample for example time to prepare mixture of four droplets in one mixing step would be much less than the time, if droplets are produced in two mixing step each with two droplets. SRS and MMS algorithm proposed for droplet streaming uses 1: 1 mixing model which is capable of mixing two droplet in single mix step, to produce demand more then two it schedule internal nodes of mixing tree repeatedly. K-droplet rotary mixer as shown in Fig. 3.1 can mixes up to K droplet simultaneously.

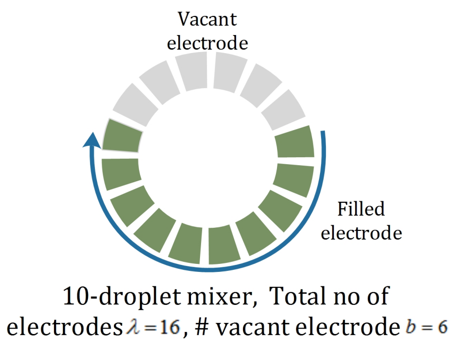

Vacant electrode
Filled electrode
10-droplet mixer, Total no of electrodes $\lambda = 16$, # vacant electrode $b = 6$

Figure 3.1: 8-droplet mixer.

It utilizes $n\colon n$ mixing model, which is capable of mixing $k = 2i, 1 \leq i \leq \lfloor \frac{K}{2} \rfloor$ in single mix step to reduce the no of mixing steps needed to generate larger demand. After a mixing operation mixer can serve as reservoir containing mixed sample and desired number of droplets can be dispensed by normal dispensing process.

For example a 4-mixer can schedule any of these three tree shown in Fig. 1.2 in single mix step.

3.2 Problem Formulation

Problem of generating multiple droplets of single target ratio using K-droplet Mixer can be formulated as below.

Inputs:

(a) A set of N different fluids, $X = x_1, x_2, ..., x_N$, $N \geq 2$, each supplied at $CF =$ 100%.

(b) A target ratio $a_1 \colon a_2 \colon \ldots \colon a_N$ of N fluids, such that the ratio-sum $L = \sum_{i=1}^{N} a_i = 2^d$, where d is the desired accuracy level in CF.

(c) Required number of target droplets, i.e., demand D

(d) Mixing capacity of mixers, K .

Output:

A schedule of mix-split steps in order to produce D droplets of the target mixture M of N fluids with the specified ratio.

Objectives:

(a) Minimize the number of mixing steps T_{ms}

(b) Minimize the number of storage units U needed on-chip.

3.3 K-Mixer Scheduling (KMS)

K-droplet mixer can prepare mixture in single mix step for different non-homomorphic tree. Due to this capability only a set of nodes of mixing tree required to be scheduled unlike in MMS and SRS where every nodes for scheduling forest need to be schedule. KMS takes mixing tree T_m produced by any of mixing algorithms [20, 18] and assign demand to its node by AssignDemand procedure presented in Algorithm 3.2. Schedulable nodes are the node which can be scheduled on K-droplet mixer. After demand assignment InputCount procedure presented in Algorithm 3.3 identifies and schedule the schedulable node.

Algorithm 3.1: $KMS(T_m, K, D)$

```
  begin
1     $AssigDemand(T_m, D)$
2     $InputCount(T_m)$
```

3.3.1 Assign Demand

Let T_m be the mixing tree obtained from any of existing mixing algorithm. D is the required demand of target droplets. *AssignDemand* procedure presented in Algorithm 3.2 takes rooted T_m and demand D, assigns D to root and recursively calls *AssignDemand* rooted at its left and right children with demand of $\lceil D/2 \rceil$ i.e., demands required at $level_l$ is half of the $level_{l+1}$. This procedure assigns the number of droplet required at each node of mixing tree to generate demand of target droplet. E.g., for Minipreparation protocol [2], having ratio 57: 28: 6: 6: 6: 3: 150 and $D = 32$ Demand required at each node is shown in Fig. 3.2

Algorithm 3.2: $AssignDemand(node, D)$

```
  begin
1     $node \cdot demand \leftarrow D$
2     if $node = leaf$ then return
3     else
4         $AssignDemand\,(node \rightarrow left, \lceil D/2 \rceil)$
5         $AssignDemand\,(node \rightarrow right, \lceil D/2 \rceil)$
```

3.3.2 Identify Schedulable Node

After assigning the demand to each node of mixing tree *InputCount* procedure presented in Algorithm 3.3 is a recursive bottom-up procedure identifies schedulable node on K-droplet mixer by counting number of reagent droplets returned from left subtree (i.e., $L = InputCount(node \rightarrow left)$) and right subtree (i.e., $R = InputCount(node \rightarrow right)$). A node is schedulable or not is determined by following two constraint.

1. $L = R$
2. $L + R \leq K$

If node fails any of above condition get scheduled with input selected using *Select-Input* procedure written in Algorithm 3.4 and demand of node returned to parent instead of returning the accumulative input count (i.e., L+R). First condition checks if inputs count at any internal node return from left $(i.e., L)$ and from right $(i.e., R)$ is unequal then schedule its left and right child only if it is non leaf, non-scheduled and demand is less then input count form its side. In case of second condition node having demand greater than K needs to schedule $\lceil demand/k \rceil$ times with $k = K$ Demand except the last one with $k = D - \lceil \frac{d}{K} \rceil - 1$.

3.3.3 Select Input

SelectInput procedure presented in Algorithm 3.4 selects the input for scheduling node from leaf or previously scheduled node. Concentration of reagent at any non-leaf node preserves only if it will select equal amount of input droplets from left and right subtree. So it recursively calls itself with the tree rooted at its left and right children dividing input requirement equally on both side and upon reaching leaf or scheduled node returns reagent with require input count. Return value is a Set of pairs $\{node, input\_count\}$ where *node* indicate reagent corresponding to leaf or scheduled node and *input_count* is amount of reagent corresponding to *node*. It used to aggregate all inputs with number of droplets required.

Algorithm 3.3: *InputsCount*(n)

```
begin
1   if $n = leaf$ then
2       return $d$
3   else
4       $d = n \cdot demand, k = K, S \leftarrow \phi, l = n \rightarrow left, r = n \rightarrow right$
5       $L \leftarrow InputsCount\,(l)$
6       $R \leftarrow InputsCount(r)$
7       if $L = R$ then
8           if $2(L + R) \leq K$ then
9               return $(L + R)$
10          else
11              go to step 24
12      else
13          if $l$ is internal & unscheduled & $L \neq \lceil d/2 \rceil$ then
14              $S \leftarrow SelectInputs(l, \lceil d/2 \rceil)$
15              schdule $l$ at time $t$ with $S$ inputs
16              $t \leftarrow t + 1$
17          if $r$ is non_leaf & $R \neq \lceil d/2 \rceil$ then
18              $S \leftarrow SelectInputs(r, \lceil d/2 \rceil)$
19              schedule $r$ at time $t$ with $S$ inputs
20              $t \leftarrow t + 1$
21          if $2d \leq K$ then
22              return $d$
23          else
24              $i \leftarrow itr \leftarrow \lceil d/K \rceil$
25              while $itr >= 1$ do
26                  if $itr = 1$ then
27                      $k = d - (i - 1)K$
28                  $S \leftarrow SelectInputs(n, k)$
29                  schedule $n$ at time $t$ with $S$ inputs
30                  $t \leftarrow t + 1$
31                  $itr \leftarrow itr - 1$
32              return $d$
```

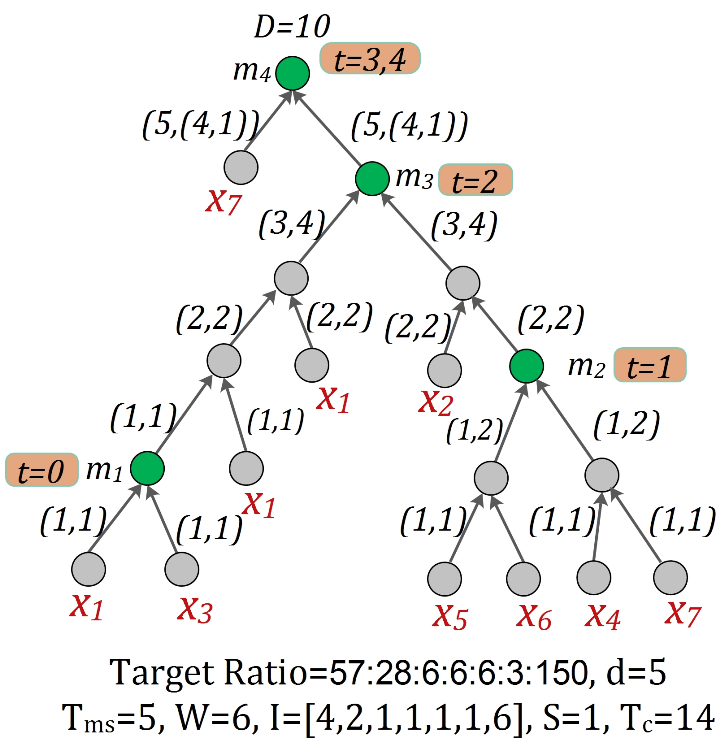


Figure 3.2: Scheduled tree using KMS with $K = 8$.

Algorithm 3.4: *SelectInput*$(node, d)$

```
   begin
1      if node is scheduled or leaf then
2          return $(node, d)$
3      else
4          return
             $select\_input(node \rightarrow left, \lceil d/2 \rceil) \cup select\_input(node \rightarrow right, \lceil d/2 \rceil)$
```

Example: Mixing tree obtained using RMA for Miniprep protocol[2] shown in Fig. 2.10, scheduled using KMS is shown in Fig. 3.2. Directed edge from child to parent node is labeled with pair whose first element indicating demand assigned by AssignDemand procedure to child and second is a vector provides information about number of input count of child return to parent with respect to time stamp by *SelectInput* procedure. Intermediate node m_1, m_2, m_3, m_4 are scheduled at time cycle $t_0, t_1, t_2, < t_3, t_4 >$ respectively. Node m_4 requires to schedule two times, generates 8 and 4 droplets respectively. Scheduled tree represented by k-ary tree having each intermediate node scheduled is shown in Fig. 3.3

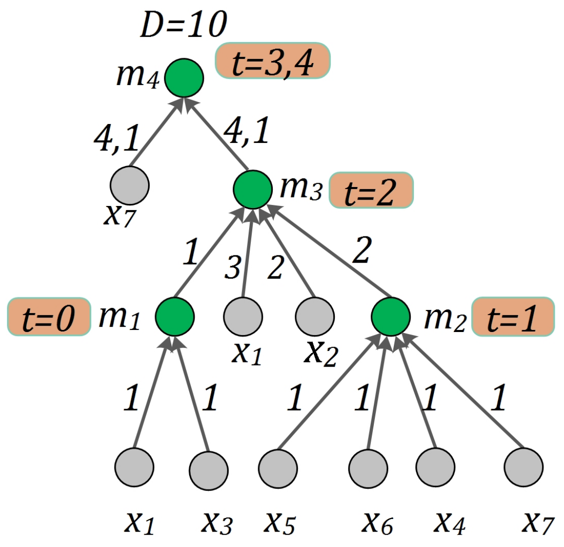
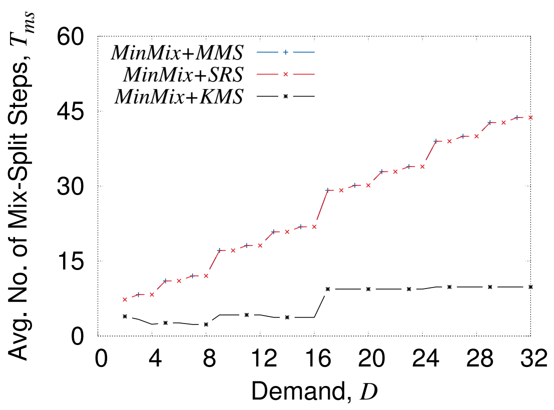


Figure 3.3: Scheduled tree represented by K-ary tree.

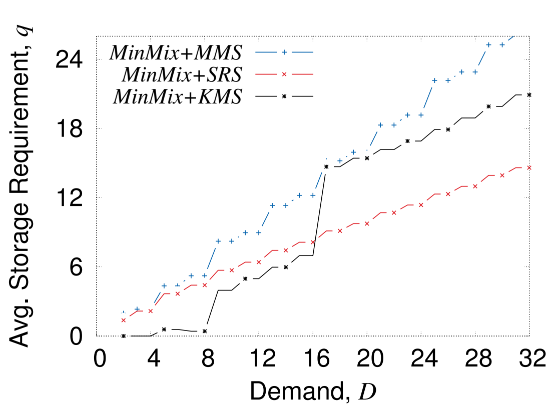
3.4 Simulation Result

Simulation performed on data set consists of 6058 target ratios of $N(2 \leq N \leq 12)$ different fluids with ratio-sum 16 and 32 and each ratio approximated in scale of 256. RMA and MinMix are used to generate mixing tree for each target ratio with demand D varies from 2 to 32. Simulation result on total mix-split T_{ms}, storage utilization U, input requirement I, and wastage count W for ratio-sum 16 and 32 is shown in Fig. 3.4 and in Appendix A.1 respectively. Simulation result for ratio-sum 16 shows that KMS improves T_{ms} by 74% than MMS and SRS, utilization U by 25% than MMS, but SRS perform better in storage utilization by 18% then KMS.There is no significance change in input requirement and wastage count as shown in figure 3.4(c) and 3.4(d).

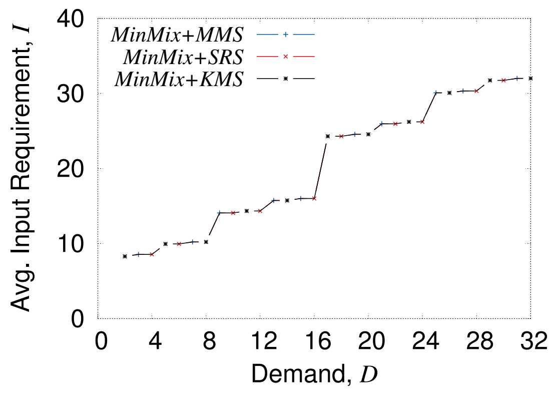
Simulation result with K=8 & ratio-sum=16 for total mix-split T_{ms}, storage requirement U, input requirement I, and wastage count W are shown in Fig. 3.4(a), 3.4(b), 3.4(c), and 3.4(d) respectively.



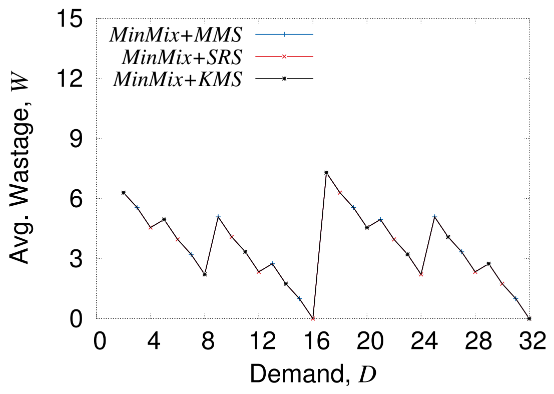
(a) Total mix-split.



(b) Storage requirement.



(c) Input requirement.



(d) Wastage count.

Figure 3.4: Simulation result for MMS, SRS, and KMS with *K=8 & ratio-sum=16*.

CHAPTER 4

Modified KMS for Mixing Tree

KMS reduces total mix-split steps significantly but compromises by increase in storage requirement. To reduce the storage requirement modified KMS (KMS-m) schedules mixing tree more then once with fraction f of total required demand D. Fraction f is used to balance the total storage requirement U and total mixing operation T_{ms}. Relation amongst T_{ms}, U and f can be represented as $f \propto \frac{1}{T_{ms}}$ and $f \propto U$ hence $f \propto \frac{U}{T_{ms}}$. Extra generated intermediate droplets are stored and utilizes in next pass.

Flow of KMS-m is same as of KMS, it first assigns demand to mixing tree generated using existing mixing algorithm [20, 18], and second step identify schedulable node using *InputCount* procedure presented in Algorithm 4.3 and finally identified nodes are scheduled with inputs selected using *SelectInput* procedure presented in Aalogorithm 4.6.

Algorithm 4.1: $KMS - m(T_m, K, D, f)$

```
begin
1   $k = f$
2   $i \leftarrow itr \leftarrow \lceil D/f \rceil$
    while $itr >= 1$ do
3       if $itr = 1$ then $k = D - (i-1)f$
4       $AssigDemand(T_m, k)$
5       $InputCount(T_m)$
```

4.1 Assign Demand

AssignDemand procedure checks for availability of reagent corresponding to node in storage before assigning demand to non-leaf nodes. Following three condition are checked and handled properly.

1. $storage[\text{node}] \geq D$
2. $0 < storage[node] < D$
3. $storage[node] = 0$

If first condition satisfied then tree below the node assigned with demand zero, and for second condition left and right child are called AssignDemand with remainder demand i.e., $\frac{D-storage[node]}{2}$, and in last condition nodes not having any storage available are assign demand half the demand of parent.

Algorithm 4.2: *AssignDemand*(*node*, *D*)

```
begin
1     $node \cdot demand \leftarrow D$
2     if $node$ is leaf then
3         return
4     if $storage[node] >= D$ then
5         $ready[node] \leftarrow D$
6         $storage[node] = storage[node] - D$
7         $AssignDemand\,(node \rightarrow left, 0)$
8         $AssignDemand\,(node \rightarrow right, 0)$
9     else
10        $ready[node] \leftarrow storage[node]$
11        $storage[node] \leftarrow 0$
12        $AssignDemand\,(node \rightarrow left, \lceil(D - ready[node])/2\rceil)$
13        $AssignDemand\,(node \rightarrow right, \lceil(D - ready[node])/2\rceil)$
```

4.2 Identify Schedulable Node

InputCount procedure is similar to the KMS except the node having storage handled differently. Two handler function are used to handle different condition.

1. $L = R$ & $storage[node] > 0$
2. $L \neq R$ &
   (a) $storage[node \rightarrow left] > 0$ *or* $storage[node \rightarrow right] > 0$
   (b) $storage[node] > 0$

$handler_1$ is used in condition 1 and 2(b) and $handler_2$ is used to handle condition 2(a).

4.3 Handler Functions

Handler function are used in InputCount procedure to handle nodes if it is available in storage. If mixture corresponding to node is available in storage then it is used and if complete demand is not satisfied then node is scheduled with remaining demand.

4.3.1 $handler_1$

$handler_1$ is used to handle node when input count return form left child i.e., L and right i.e., R are same and mixture corresponding to node is available in storage.

Algorithm 4.3: *InputsCount*(n)

```
begin
    if $n$ is leaf then return $n \cdot d$
    else
        $d = n \cdot demand, k = K, S \leftarrow \phi, l = n \rightarrow left, r = n \rightarrow right$
        $L \leftarrow InputsCount\,(l)$
        $R \leftarrow InputsCount\,(r)$
        if $L = R$ then
            if $storage[node] > 0$ & $d \neq 0$ then $handler_1(n, L)$
            else
                if $2(L + R) \leq K$ then return $(L + R)$
                else go to step 25
        else
            if $storage[l] \geq 0$ & $l \cdot demand \neq L$ then $handler_2(l)$
            else if $l$ is internal & unscheduled & $L \neq l \cdot demand$ then
                $S \leftarrow SelectInputs(l, l \cdot demand)$
                schedule $l$ at time $t$ with $S$ inputs, $t \leftarrow t + 1$
                if $S > l \cdot demand$ then $stored[l] = S - l \cdot demand$
            if $storage[r] \geq 0$ & $r \cdot demand \neq R$ then $handler_2(r)$
            else if $r$ is internal & $R \neq r \cdot demand$ then
                $S \leftarrow SelectInputs(r, r \cdot demand)$
                schedule $r$ at time $t$ with $S$ inputs, $t \leftarrow t + 1$
                if $S > r \cdot demand$ then $storage[r] = S - l \cdot demand$
            if $storage[n] > 0$ then $handler_1(n, l \cdot demand)$
            if $2 * (l \cdot demand + r \cdot demand) \leq K$ then return$(l.d + r.d)$
            else
                $i \leftarrow itr \leftarrow \lceil d/K \rceil$
                while $itr >= 1$ do
                    if $itr = 1$ then $k = d - (i - 1)K$
                    $S \leftarrow SelectInputs(n, k)$
                    schedule $n$ at time $t$ with $S$ inputs
                    $s \leftarrow s + S$ , $t \leftarrow t + 1$, $itr \leftarrow itr - 1$
                if $s > n \cdot demand$ then $storage[node] = s - n \cdot demand$
                return $d$
```

4.3.2 *handler*$_2$

handler$_2$ is used to handle node when input count return form left child i.e., L and right i.e., R are not same and mixture corresponding to scheduling child is available in storage. This handler function schedules child and return demand corresponding to child to parent.

Algorithm 4.4: *handler*$_1(n, amt)$

```
begin
    if $2(2 * amt + ready[n]) \leq K$ then
        $t \leftarrow 2 * amt + ready[n]$
        return $t$
    else
        $i \leftarrow itr \leftarrow \lceil (n \cdot d - ready[n])/K \rceil$
        while $itr >= 1$ do
            if $itr = 1$ then  $k = d - ready[n] - (i-1)K$
            $S \leftarrow SelectInputs(n, k)$
            $schedule\ n\ at\ time\ t\ with\ S\ inputs$
            $s \leftarrow s + S$ , $t \leftarrow t + 1$, $itr \leftarrow itr - 1$
        if $n \cdot demand < s$ then
            $stored[n] \leftarrow stored[n] + s - (n \cdot demand - ready[n])$
        return $d$
```

Algorithm 4.5: *handler*$_2(n)$

```
begin
    $S \leftarrow SelectInputs(n, n \cdot demand - ready[n])$
    $schedule\ n\ at\ time\ t\ with\ S\ inputs$
    $t \leftarrow t + 1$
    if $n \cdot demand - ready[n] < s$ then
        $stored[n] \leftarrow stored[n] + s - (n \cdot demand - ready[n])$
```

4.4 Select Input

SelectInput procedure presented in Algorithm 4.6 selects input for schedulable node from leaf, previously scheduled node, or storage. Concentration of reagent at any non-leaf node preserves only if it will select equal amount of input droplets from left and right subtree. So it recursively calls itself with the tree rooted at its left and right children dividing input requirement equally on both side and upon reaching leaf or scheduled node or node having storage returns reagent with require input count. Return value is a Set of pairs $\{node, input\_count\}$ where *node* indicate reagent corresponding to leaf or scheduled node or node having storage and *input_count* is amount of reagent corresponding to *node*. It used to aggregate all inputs with number of droplets required.

Example: Mixing tree obtained using RMA for Miniprep protocol[2] shown in Fig. 2.10, scheduled using KMS-m is shown in Fig. 4.1. To generate demand D=32, with d=8 mixing tree need to scheduled four time, tree corresponding to each pass are shown separately with scheduled node represented by green color. Directed edge from child to parent node is labeled with pair whose first element indicating demand assigned by *AssignDemand* procedure to child and second is a vector provides information about number of input count of child return to parent with respect to time stamp by *SelectInput* procedure. Table below each tree show the storage informa-

Algorithm 4.6: *SelectInput*(n, amt)

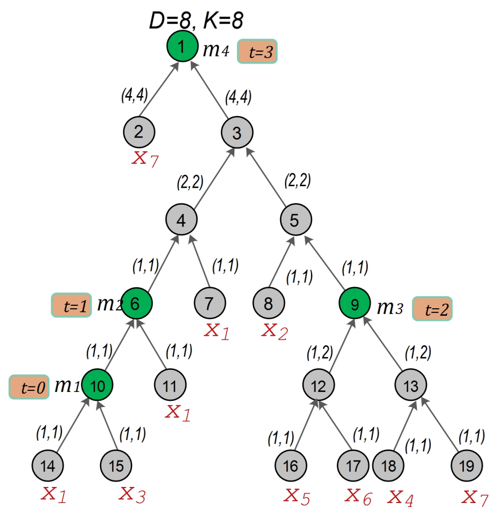
```
begin
1   if node is scheduled or leaf or storage[node] > 0 then
2       return amt
3   else
4       return SelectInput(node → left, ⌈amt/2⌉) + SelectInput(node →
        right, ⌈amt/2⌉)
```

tion after tree get scheduled. Extra generated droplet in a pass stored in storage and used in subsequent passes. For e.g., after execution of first pass node 9 generated 4 droplet and used one hence 3 droplets stored and are used in subsequent three pass one in each.

4.5 Simulation Result for KMS-m

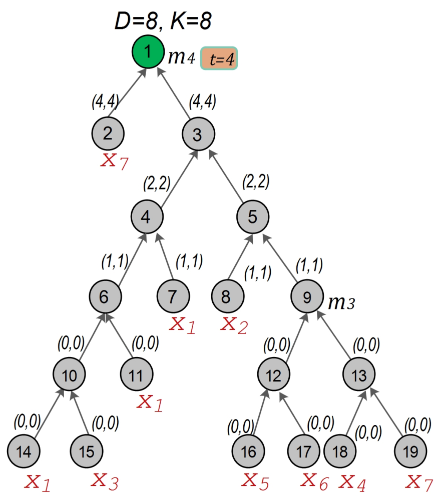
Simulation performed on data set consists of 6058 target ratios of $N(2 \leq N \leq 12)$ different fluids with ratio-sum L 16 and 32 and each ratio approximated in scale of 256. RMA and MinMix are used to generate mixing tree for each target ratio with demand D varies from 2 to 32. Simulation result on total mix-split T_{ms}, storage utilization U, input requirement I, and wastage count W for ratio-sum 16 and 32 is shown in Fig. 3.4 and in Appendix A.2 respectively. Simulation result show that KMS-m improves T_{ms} by 79% than MMS and SRS, U by 80.2%, 67% than MMS and SRS respectively.

Simulation result with *K=8, ratio-sum=16 & f=8* for total mix-split T_{ms}, storage requirement U, input requirement I, and wastage count W are shown in Fig. 4.2(a), 4.2(b), 4.2(c), and 4.2(d) respectively.



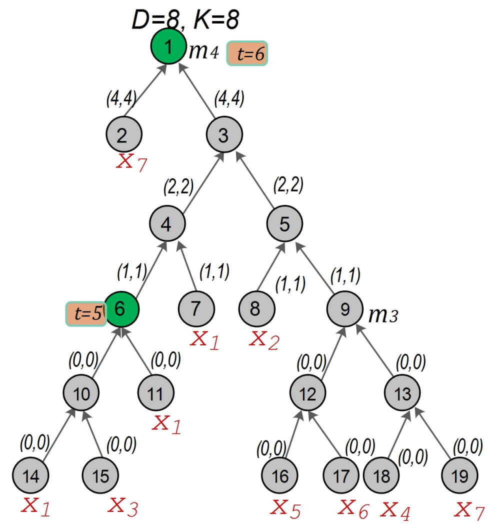
node id	amount
10	1
6	1
9	3

(a) Scheduled tree after pass one.



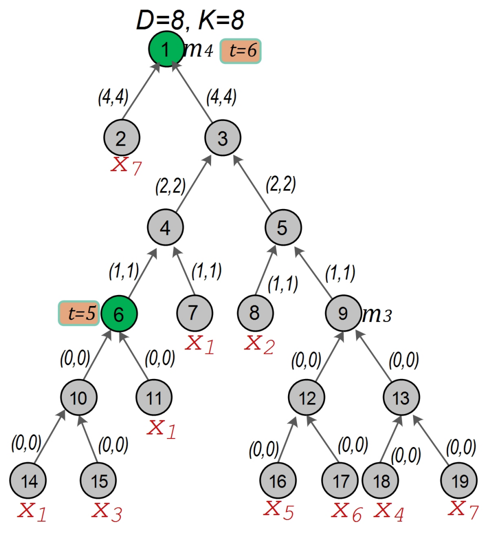
node id	amount
10	1
9	2

(b) Scheduled tree after pass two.



node id	amount
6	1
9	1

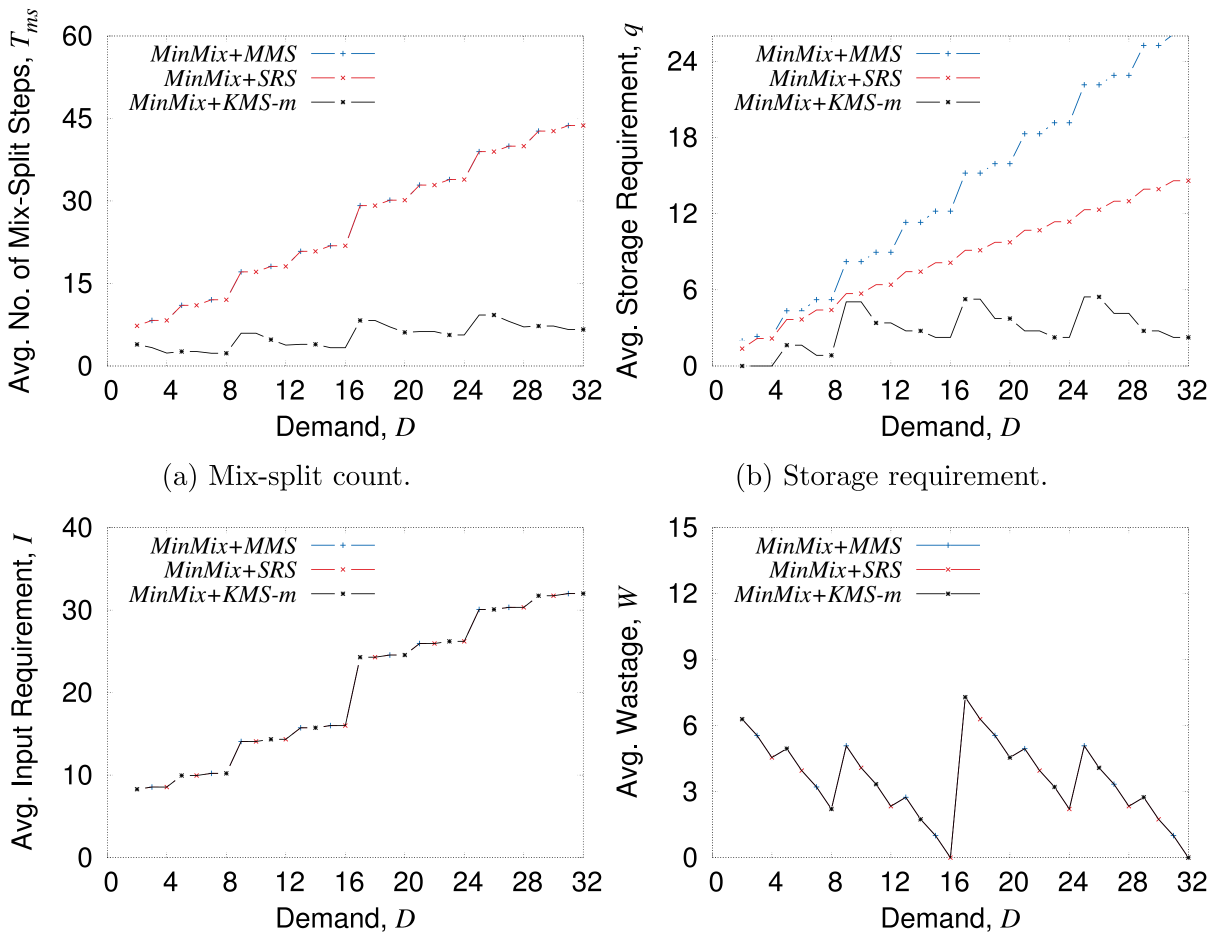
(c) Scheduled tree after pass three.



node id	amount
6	1
9	1

(d) Scheduled tree after pass four.

Figure 4.1: Scheduled tree for Miniprep protocol by m-KMS with $D = 32, f = 8$.



(a) Mix-split count.

(b) Storage requirement.

(c) Input requirement.

(d) Waste count.

Figure 4.2: Simulation result for MMS, SRS, and m-KMS with *K=8, ratio-sum=16 & f=8.*

CHAPTER 5

KMS for Mixing Graph

Some mixing algorithm generates mixing graph which contains sharable node pair. These sharable node pair shares common ratio hence tree rooted at these nodes are same. To generate demand of two droplets existing algorithms mixes one of the paired node and extra produced droplets at this node is used by other paired node. MDST may require demand of paired node more then one hence KMS for mixing graph (KMS-g) produces combined demand of sharable node pair at one of the paired node which is utilized by both node.

KMS-g takes mixing graph G_m produced by any of mixing algorithms [8, 12, 11] and assign demand to its node by AssignDemand procedure presented in Algorithm 5.2. Schedulable nodes are the node which can be scheduled on K-droplet mixer. After demand assignment InputCount procedure presented in Algorithm 5.4 identifies and schedule the schedulable node.

Algorithm 5.1: $KMS - g(G_m, K, D, f)$

```
  begin
1     $k = f$
2     $i \leftarrow itr \leftarrow \lceil D/f \rceil$
      while $itr >= 1$ do
3         if $itr = 1$ then $k = D - (i-1)f$
4         $AssigDemand(G_m, k)$
5         $InputCount(G_m)$
```

5.1 Assign Demand

Let G_m be the mixing graph obtained from any of existing mixing algorithm. D is the required demand of target droplets. *AssignDemand* procedure presented in Algorithm 5.2 takes rooted G_m and demand D, assigns D to root and recursively calls *AssignDemand* rooted at its left and right children with demand of $\lceil D/2 \rceil$ i.e., demands required at $level_l$ is half of the $level_{l+1}$. Demand of paired node produced by one node which is called donor node and other is called receiver node. This procedure assigns the number of droplet required at each node of mixing tree to generate demand of target droplet.

Algorithm 5.2: *AssignDemand*(*node*, *D*)

```
begin
1    $node \cdot demand = D$
2    if $if\ node\ is\ donor$ then
3        if $storage[node] \geq D$ then
4            $ready[node] \leftarrow D$
5            $storage[node] \leftarrow storage[node] - D$
6        else
7            $ready[node] \leftarrow storage[node]$
8            $storage[node] \leftarrow 0$
9    else
10       if $storage[node] \geq D$ then
11           $ready[node] \leftarrow D$
12           $storage[node] \leftarrow storage[node] - D$
13           $AssignDemand(node \cdot left, 0)$
14           $AssignDemand(node \cdot right, 0)$
15       else
16           $ready[node] = storage[node]$
17           $storage[node] = 0$
18           $x = \lceil \frac{node \cdot demand - ready[node]}{2} \rceil$
19           $AssignDemand(node \cdot left, x)$
20           $AssignDemand(node \cdot right, x)$
```

5.2 Identify Schedulable Node

After assigning the demand to each node *InputCount* procedure presented in Algorithm 5.4 is a recursive bottom-up procedure identifies schedulable node on K-droplet mixer by counting number of reagent droplets returned from left subtree (i.e., $L = InputCount(node \rightarrow left)$) and right subtree (i.e., $R = InputCount(node \rightarrow right)$). This procedure are same as Algorithm 4.3 except the way donor and receiver node handle. If node is receiver it is treated as leaf node and demand is returned to the parent. If node is donor and if L and R are same then it schedules node with aggregate demand of both paired node, otherwise it check for schedulablity of left and right child and schedule them accordingly.

Algorithm 5.4: $InputCount(node)$

```
begin
1   if  if n is leaf or receiver then  return $n \cdot d$
2   else
3       $d = n \cdot demand, k = K, S \leftarrow \phi, l = n \rightarrow left, r = n \rightarrow right$
4       $L \leftarrow InputsCount\,(l)$
5       $R \leftarrow InputsCount\,(r)$
6       if node is donor then
7           if $L = R$ then  goto line 19
8           else
9               if $storage[l] \geq 0$ & $l \cdot demand \neq L$ then $handler_2(l)$
10              else if l is internal & unscheduled & $L \neq l \cdot demand$ then
11                  $S \leftarrow SelectInputs(l, l \cdot demand)$
12                  schedule l at time t with S inputs, $t \leftarrow t + 1$
13                  if $S > l \cdot demand$ then  $stored[l] = S - l \cdot demand$
14              if $storage[r] \geq 0$ & $r \cdot demand \neq R$ then $handler_2(r)$
15              else if r is internal & $R \neq r \cdot demand$ then
16                  $S \leftarrow SelectInputs(r, r \cdot demand)$
17                  schedule r at time t with S inputs, $t \leftarrow t + 1$
18                  if $S > r \cdot demand$ then $storage[r] = S - l \cdot demand$
19              $i \leftarrow itr \leftarrow \lceil (d + p.demand - ready[n])/K \rceil$
20              while $itr >= 1$ do
21                  if $itr = 1$ then $k = d + p.demand - ready[node] - (i - 1)K$
22                  $S \leftarrow SelectInputs(n, k)$, schedule n at time t with S inputs
23                  $s \leftarrow s + S$ , $t \leftarrow t + 1$, $itr \leftarrow itr - 1$
24              if $d + p.demand - ready[node] < s$ then
25                  $storage[node] = s - (d + p.demand - ready[node])$
26              return d
27      else
28          proceed as algorithm 4.3 form line 6
```

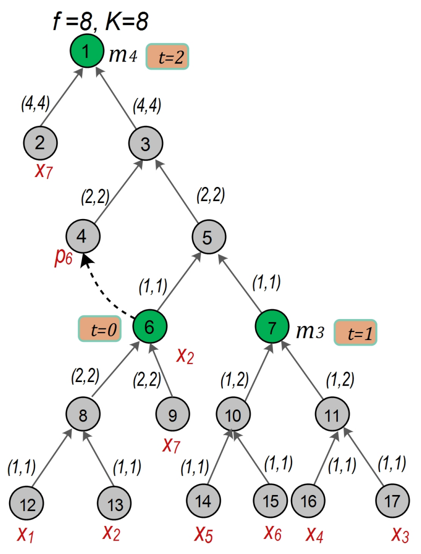
5.3 Select Input

SelectInput procedure presented in Algorithm 5.5 selects input for scheduling node from leaf, previously scheduled node, or storage. Concentration of reagent at any non-leaf node preserves only if it will select same amount of input droplets from left and right subtree. So it recursively calls itself with the tree rooted at its left and right children dividing input requirement equally on both side and upon reaching leaf or scheduled node or node having storage returns reagent with require input count. Return value is a Set of pairs $\{node, input\_count\}$ where *node* indicate reagent corresponding to leaf or scheduled node or node having storage and *input_count* is amount of reagent corresponding to *node*. It used to aggregate all inputs with number of droplets required. Example: Mixing graph obtained using MTCS for ratio 26: 21: 2: 2: 3: 3: 199 the PCR master-mix used for DNA amplification [1];shown

Algorithm 5.5: $SelectInput(n, amt)$

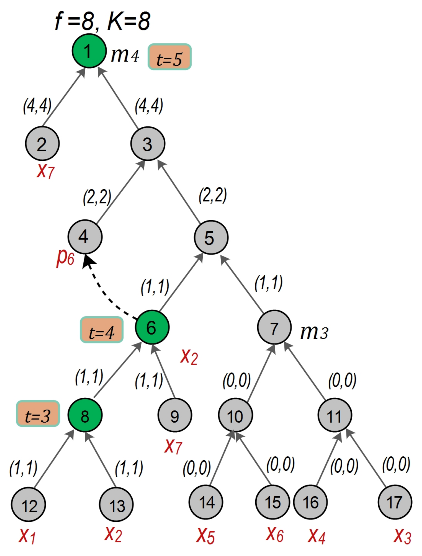
```
begin
    if node is scheduled or leaf or stored[node] > 0 then
        return amt
    else
        return  SelectInput(node → left, ⌈amt/2⌉) + SelectInput(node →
          right, ⌈amt/2⌉)
```

in Fig. 5.1(a), scheduled using KMS-g is shown in Fig. 5.1. To generate demand $D = 32$, with $f = 8$ mixing tree need to scheduled four time, tree corresponding to each pass are shown separately with scheduled node represented by green color. Directed edge from child to parent node is labeled with pair whose first element indicating demand assigned by *AssignDemand* procedure to child and second is a vector provides information about number of input count of child return to parent with respect to time stamp by *SelectInput* procedure. Table below each tree show the storage information after tree get scheduled. Extra generated droplet in a pass stored in storage and used in subsequent passes. For e.g., after execution of first pass node 7 generated 4 droplet and used one hence 3 droplets stored and are used in subsequent three pass one in each.



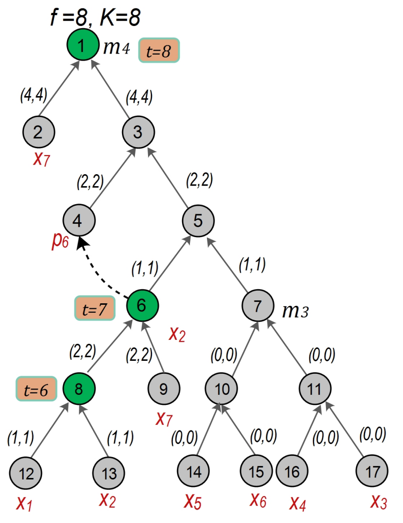
node id	amount
6	1
7	3

(a) Scheduled tree after pass one.



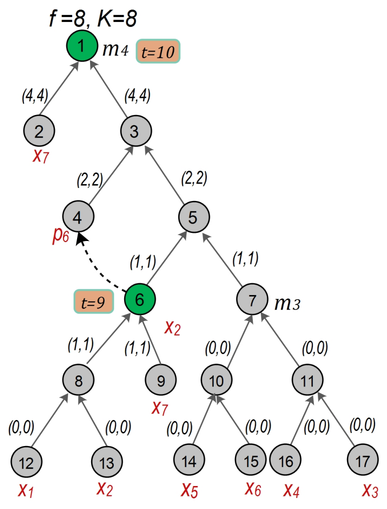
node id	amount
8	1
7	2

(b) Scheduled tree after pass two.



node id	amount
6	1
7	1
8	1

(c) Scheduled tree after pass three.

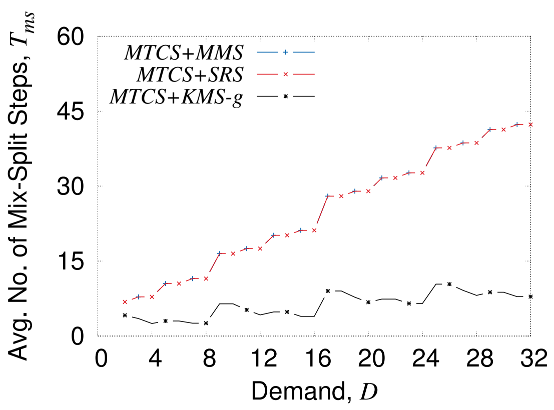


(d) Scheduled tree after pass four.

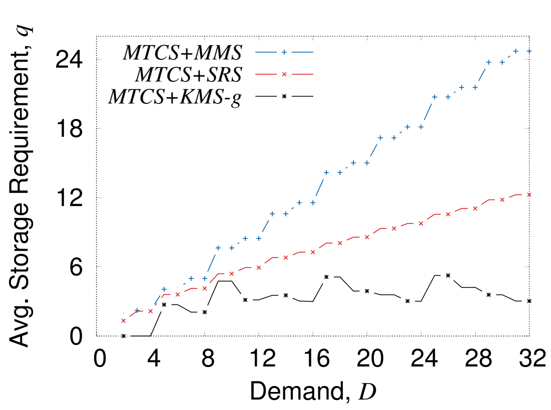
Figure 5.1: Scheduled tree for Miniprep protocol by m-KMS with $D = 32, d = 8$.

5.4 Simulation Result for KMS-g

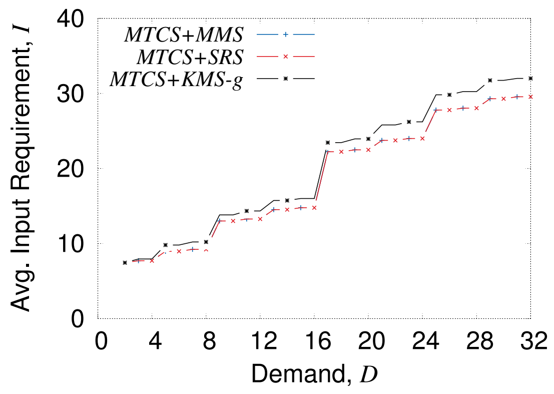
Simulation result with *K=8, ratio-sum=16 &f =8* on same data set used in previous chapter with mixing algorithm MTCS for total mix-split T_{ms}, storage requirement U, input requirement I, and wastage count W are shown in Fig. 5.2(a), 5.2(b), 5.2(c), and 5.2(d) respectively. Total mix-split step T_{ms} improves by 76.16% than MMS and SRS and by 79.6%, 66.5% than MMS, SRS respectively.



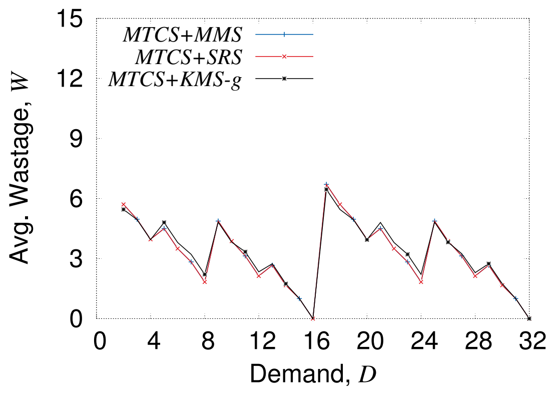
(a) Mix-Split count.



(b) Storage requirement.



(c) Input requirement.



(d) Wastage count.

Figure 5.2: Simulation result for MMS, SRS, and KMS-g with *K=8, ratio-sum=16 & d=8*.

CHAPTER 6

Conclusion and Future Work

The focus of dissertation was to generate multiple demand of single target efficiently. To reduce mix-split T_{ms}, storage U, waste W, and input requirement I, KMS, KMS-m, and KMS-g uses K-droplet rotary mixer which reduces the number of schedulable node in the mixing tree, therefore it reduces total mixing step as compare to existing scheduling algorithm MMS, and SRS. Further reduction in storage utilization is achieved by modified version on KMS, named KMS-m, which repeatedly schedules the mixing tree. To achieve MDST for mixing graph KMS-g was proposed which schedule mixing graph and perform better then MMS, and SRS in number of mixing step performed.

Modified KMS uses static balancing factor f to balance mix-split T_{ms} and storage utilization U, which is not best optimization of objective. As future work an optimization algorithm can be applied to find value of f to balance U and T_{ms} in most optimal way.

Bibliography

[1] Pcr master mix calculator. url = http://www.mutationdiscovery.com, 2006.

[2] Preparation of plasmid dna by alkaline lysis with sds: Minipreparation, cold spring harb protocols., 2006.

[3] Rotory mixer. url = http://microfluidics.ee.duke.edu/, 2006.

[4] K ChakrabartyandT. Xu, digital microfluidic biochips: Design and optimization, 2010.

[5] Sung Kwon Cho, Hyejin Moon, and Chang-Jin Kim. Creating, transporting, cutting, and merging liquid droplets by electrowetting-based actuation for digital microfluidic circuits. *Microelectromechanical Systems, Journal of*, 12(1):70–80, 2003.

[6] Trung Anh Dinh, Shinji Yamashita, and Tsung-Yi Ho. A network-flow-based optimal sample preparation algorithm for digital microfluidic biochips. In *Design Automation Conference (ASP-DAC), 2014 19th Asia and South Pacific*, pages 225–230. IEEE, 2014.

[7] Yi-Ling Hsieh, Tsung-Yi Ho, and Krishnendu Chakrabarty. On-chip biochemical sample preparation using digital microfluidics. In *Biomedical Circuits and Systems Conference (BioCAS), 2011 IEEE*, pages 297–300. IEEE, 2011.

[8] Yi-Ling Hsieh, Tsung-Yi Ho, and Krishnendu Chakrabarty. A reagent-saving mixing algorithm for preparing multiple-target biochemical samples using digital microfluidics. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 31(11):1656–1669, 2012.

[9] Chi-Mei Huang, Chia-Hung Liu, and Juinn-Dar Huang. Volume-oriented sample preparation for reactant minimization on flow-based microfluidic biochips with multi-segment mixers. In *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, pages 1114–1119. EDA Consortium, 2015.

[10] Juinn-Dar Huang, Chia-Hung Liu, and Ting-Wei Chiang. Reactant minimization during sample preparation on digital microfluidic biochips using skewed mixing trees. In *Proceedings of the International Conference on Computer-Aided Design*, pages 377–383. ACM, 2012.

[11] Sudhakar Kumar, Sandip Roy, Partha Pratim Chakrabarti, Bhargab B Bhattacharya, and Krishnendu Chakrabarty. Efficient mixture preparation on digital

microfluidic biochips. In *Design and Diagnostics of Electronic Circuits & Systems (DDECS), 2013 IEEE 16th International Symposium on*, pages 205–210. IEEE, 2013.

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

[13] Chia-Hung Liu, Kuo-Cheng Shen, and Juinn-Dar Huang. Reactant minimization for sample preparation on microfluidic biochips with various mixing models. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 34(12):1918–1927, 2015.

[14] Michael G Pollack, Richard B Fair, and Alexander D Shenderov. Electrowetting-based actuation of liquid droplets for microfluidic applications. *Applied Physics Letters*, 77(11):1725–1726, 2000.

[15] Michael George Pollack. *Electrowetting-based microactuation of droplets for digital microfluidics*. PhD thesis, Duke University, 2001.

[16] Sandip Roy, Bhargab B Bhattacharya, and Krishnendu Chakrabarty. Optimization of dilution and mixing of biochemical samples using digital microfluidic biochips. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 29(11):1696–1708, 2010.

[17] Sandip Roy, Sudhakar Kumar, Partha Pratim Chakrabarti, Bhargab B Bhattacharya, and Krishnendu Chakrabarty. Demand-driven mixture preparation and droplet streaming using digital microfluidic biochips. In *Design Automation Conference (DAC), 2014 51st ACM/EDAC/IEEE*, pages 1–6. IEEE, 2014.

[18] Sudip Roy, Bhargab B Bhattacharya, Partha P Chakrabarti, and Krishnendu Chakrabarty. Layout-aware solution preparation for biochemical analysis on a digital microfluidic biochip. In *VLSI Design (VLSI Design), 2011 24th International Conference on*, pages 171–176. IEEE, 2011.

[19] Sudip Roy, Bhargab B Bhattacharya, and Krishnendu Chakrabarty. Waste-aware dilution and mixing of biochemical samples with digital microfluidic biochips. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011*, pages 1–6. IEEE, 2011.

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

Dissemination from this Dissertation

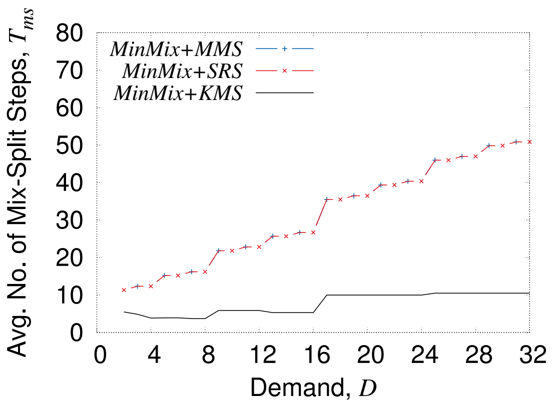
The following paper have been submitted and in process due to the work related to this thesis.

1. *Satendra Kumar*, Ankur Gupta, Sudip Roy and Bhargab Bhattacharya, "Design Automation of Multiple-Demand Mixture Preparation using a k-Array Rotary Mixer on Digital Microfluidic Biochip", submitted to the 34th IEEE International Conference on Computer Design (ICCD), 2016, Phoenix, USA.

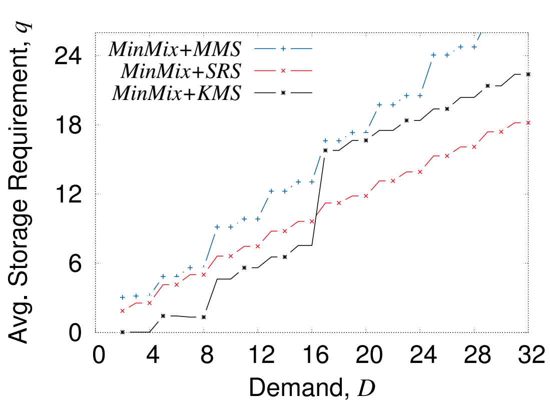
Appendix

A.1

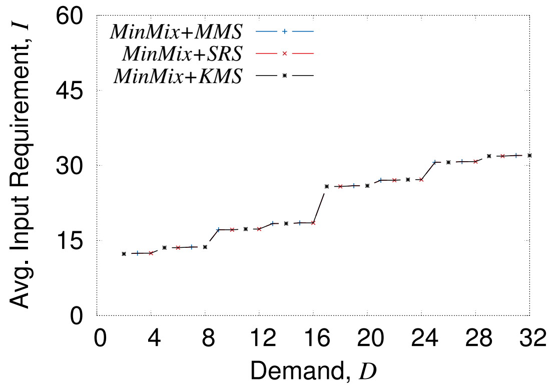
KMS Simulation result with *K=8 & ratio-sum=32*



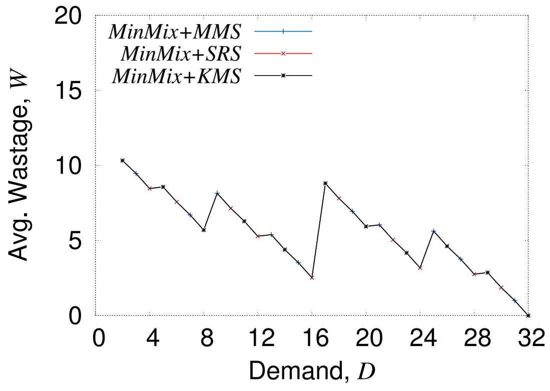
(a) Total mix-split.



(b) Storage requirement.



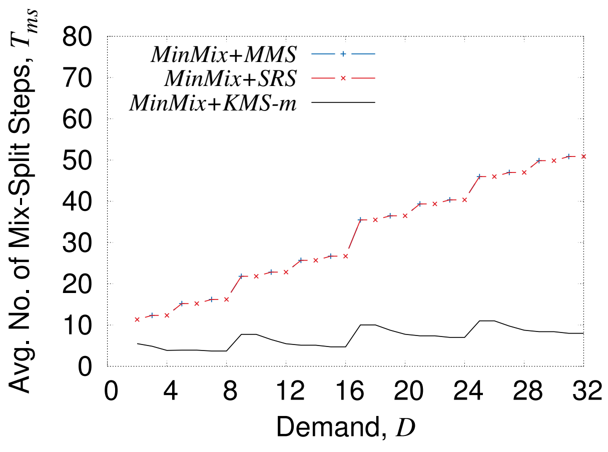
(c) Input requirement.



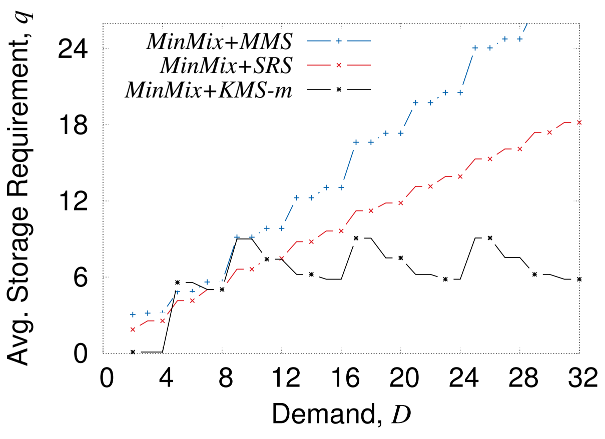
(d) Wastage count.

A.2

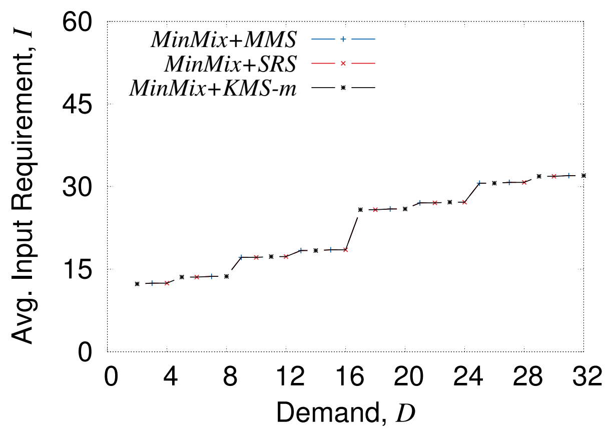
KMS-m Simulation result for *K=8, ratio-sum=32 & f* = 8



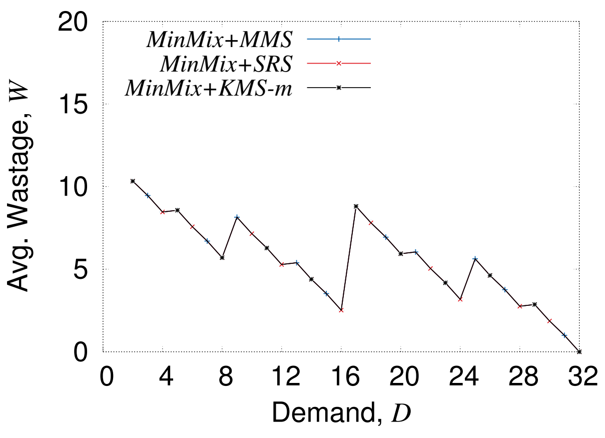
(a) Mix-split count.



(b) Storage requirement.



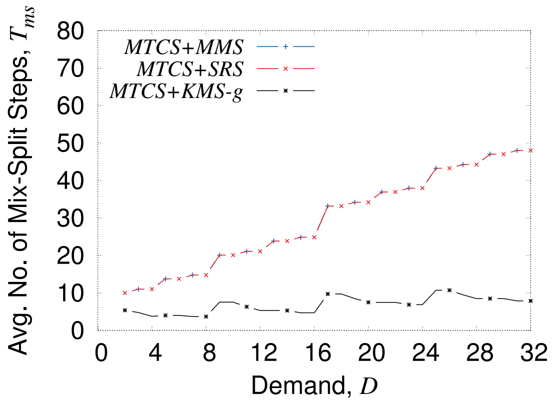
(c) Input requirement.



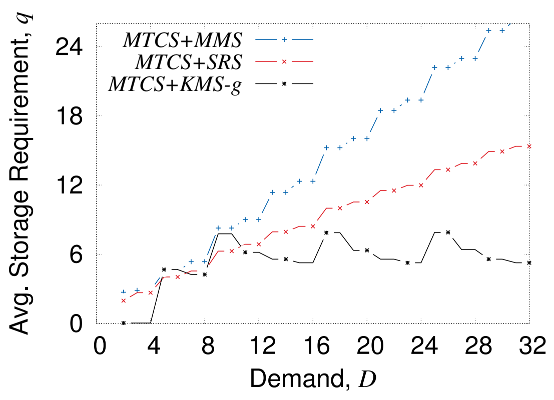
(d) Waste count.

A.3

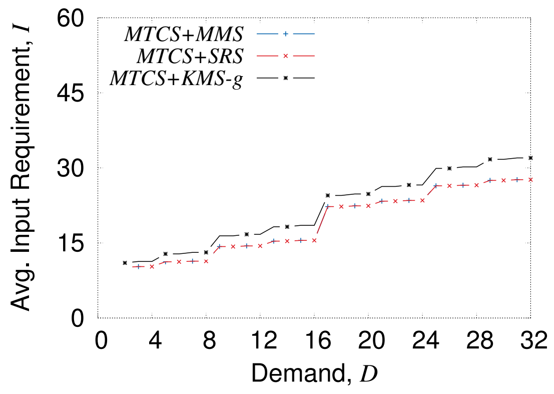
KMS-g Simulation result with *K=8, ratio-sum=32 & f* = 8



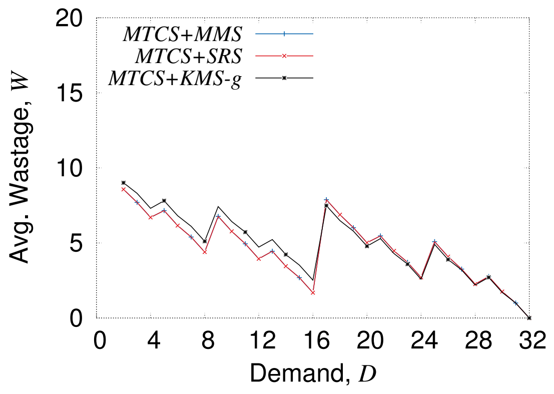
(a) Mix-split count.



(b) Storage requirement.



(c) Input requirement.



(d) Wastage count.