Heterogeneous Earliest Finish Time based Scheduling for Digital Microfluidic Biochips

Rajesh Kolluri, J. V. Phani Kumar and Sumanta Pyne
Department of Computer Science and Engineering, National Institute of Technology Rourkela, Odisha 769008, India

Keywords: Lab-on-chip, Microfluidics, DMFB, MEMS, Scheduling, DMHEFT.

Abstract: One of the recent emerging technology in biochemical analysis field is Lab-on-chip (LOC) technology which uses digital microfluidics property to manipulate droplets discretely. LOC efficiently carries out all biochemical operations we do in traditional laboratories on a single reconfigurable chip called as Digital Microfluidic Biochip (DMFB). DMFBs helps to achieve parallelism and miniaturization compared to traditional laboratory methods in terms of samples and equipment used. One of the important problem in DMFB synthesis is scheduling. We present a simple method called as Heterogeneous Earliest Finish Time for digital microfluidics (DMHEFT) for scheduling DMFB. It is a greedy heuristic based list scheduling. HEFT is previously used for task scheduling where multiple heterogeneous processors are available for solving inter-dependent tasks depicted as DAG. In this paper, it is applied to Microfluidic biochips. DMHEFT uses Upward rank value to prioritize the tasks or operations and earliest finish time to assign tasks to different modules like mixers, heaters and detectors etc. Simulation results show that it produces better assay lengths and run time compared to existing algorithms.

1 INTRODUCTION

Digital Microfluidics is small part of bio-MEMS (Micro Electro Mechanical Systems) or Lab-On-Chip (Verpoorte and De Rooij, 2003). Basic agenda of Digital Microfluidic Biochips (DMFBs) is to integrate all required functions of biochemical analysis onto a single small-sized chip using microfluidics. In other words DMFBs are specially designed chips for biological and chemical analysis. Digital Microfluidics explores so many advantages compared to traditional laboratory techniques or methods. The main advantages opting for DMFB rather than traditional laboratory analysis of samples are DMFB reduces time, cost and achieves automation, miniaturization. DMFB has applications in the area of clinical diagnostics (Schulte et al., 2002), Analyzing human physiological fluids like saliva, urine, sweat, serum, plasma, blood and tears (Srinivasan et al., 2003), DNA sequence analysis and it can also be used to test some fluids related to terrorist activities using bio-samples (Su and Chakrabarty, 2008) and fluorescence detection (Bhardwaj and Jha, 2018).

DMFB works by manipulating discrete level droplets on biochip. Unlike the continuous-flow microfluidics or analog-digital microfluidics which uses pumps and valves to make fluids flow continuously and mix them with reagents (Verpoorte and De Rooij, 2003). Digital microfluidics uses Electro wetting on Dielectric (EWoD) property to make droplets move discretely and mix them with reagents on 2-Dimensional array of electrodes (Pollack et al., 2002). DMFB consists of two glass plates placed one above the other and droplets will move between these plates. Upper glass plate consists of coating of hydrophobic layer which prevents fluids stick to the surface of the plate and one ground electrode that spreads across the entire upper plate. In addition to hydrophobic layer, lower plate has $m * n$ control electrodes and a dielectric layer which works as an insulator. A droplet will move from one electrode to another electrode by activating one of the adjacent electrode where you want to move the droplet while deactivating the electrode which is holding the droplet now. DMFB is capable of performing various operations on droplets like dispensing, dilute, mixing, merging two droplets into one droplet, splitting a droplet, storage and detection of change in a droplet in terms of color, heating, volume etc. by putting a LED and photo diode on the corresponding electrode (Su and Chakrabarty, 2004). DMFB is dynamically reconfigurable which means all the operations can be

Kolluri, R., Kumar, J. and Pyne, S.
Heterogeneous Earliest Finish Time based Scheduling for Digital Microfluidic Biochips.
DOI: 10.5220/0007367101750182
In *Proceedings of the 12th International Joint Conference on Biomedical Engineering Systems and Technologies (BIOSTEC 2019)*, pages 175-182
ISBN: 978-989-758-353-7

done on-chip and there is no permanent designated place for any operation, operations can be done anywhere, anytime on the chip one at a time. Various biological, chemical assays are specified as directed acyclic graphs (DAGs) depicting the different type of operations and dependencies between them (Su and Chakrabarty, 2008).

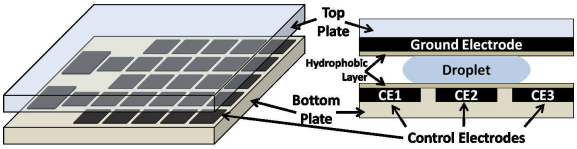

Figure 1: Digital Microfluidic Biochip structure (Grissom and Brisk, 2012b).

Synthesis of DMFB consists of three steps namely Scheduling, Placement and Routing (Grissom and Brisk, 2012b). Also called as DMFB compiler. Scheduling is allocating absolute start, stop time slots to every operation depicting that when an operation has to occur pertaining to resource constraints ensuring there are enough resources to process certain bioassay. In this paper we emphasize on scheduling. In conjunction to this placer and router algorithm can be used to finish the synthesis step. Scheduling is more important than placement and routing part of synthesis because of operations (mixing and detection) take more time compared to actual placing of operations and transportation of droplets.

2 RELATED WORK

Some of the significant scheduling algorithms are based on Heuristic based List Scheduling variants, Genetic Algorithms and Integer Linear Programming (ILP).

List Scheduling (LS) (Micheli, 1994) is a heuristics based greedy scheduling algorithm. LS generates a prioritized sequence of tasks and allocates processor to tasks in the sequence of their priority which minimizes some predefined objective function. Modified List Scheduling (MLS) (Su and Chakrabarty, 2008),(Su and Chakrabarty, 2004) algorithm is an augmentation to List Scheduling algorithm. Unlike LS, MLS has a rescheduling step which effects in achieving legal schedule under resource constraints. It is a polynomial time executed greedy based heuristic algorithm.

Force-directed List Scheduling (FDLS) (O'Neal et al., 2012) is also based on List Scheduling not on MLS. It modifies/replaces priority function (which is used to prioritize tasks for execution sequence) of List Scheduling. First, this method was used for the high level synthesis of digital signal processing. FDLS computes the priorities of tasks at every scheduling step in response to the dependencies underlying between tasks (vertices) as shown in DAG which were already scheduled in earlier calculation of priorities and also Force calculation which considers the fewest steps at which the operations can be scheduled which will be known by As Soon As Possible (ASAS) and As Late As Possible (ALAP). FDLS produces better assay lengths than MLS but runs slower than MLS because of complex rank calculation mechanism.

Path Scheduling (PS) (Grissom and Brisk, 2012b) schedules the DAG, path after path unlike node by node in List Scheduling. All operations on a path are scheduled contiguously. Path scheduling works by calculating two priorities Critical path priority and Independent path priority. By calculating node priorities, it only explores paths with the lowest priority thereby reducing the number of droplets stored in DMFB. If storage droplets are more on-chip it adversely affects the performance of chip because it is indirectly reducing functional area of chip. When a large assay has to be scheduled on small DMFB then PS works better than LS and FDLS. But, PS only works for trees/forests, failed to schedule other type of DAGs.

Two Genetic List Scheduling Algorithms GA1 (Su and Chakrabarty, 2008), GA2 (Ricketts et al., 2006) were there which takes long run time but converge to locally optimal solutions. These two also based on List scheduling. Which initially use LS to produce a legal initial schedule and randomly varying operation priorities using crossover, mutation. Genetic algorithms gives lesser bio-assay lengths but runs slower compared to LS, PS and FDLS. Some Integer Linear Programming (ILP) based scheduling algorithms (Ding et al., 2001), (Su and Chakrabarty, 2004), (Su and Chakrabarty, 2008), are there same as Genetic algorithms achieves optimal solutions but has exponential running time.

3 SCHEDULING OVERVIEW

3.1 Scheduling Preliminaries

Scheduling of a DMFB is an NP-Complete problem. Scheduling DMFB operations is actually scheduling a DAG. Scheduling determines each operation's starting execution time slot and finishing execution time slot sticking to resource constraints and operation dependences derived from DAG. We cannot say that a particular schedule that we get is legal at all times because there is a chance that resource requirements of

bio-assay may exceed the existing resource or modules of a DMFB. The goal of scheduling algorithm is to reduce the bio-assay execution time.

DMFB consists of Different modules like General modules to execute normal operations like mix, merge, split and Specialized modules to execute special operations like detection, heating and Input ports to implement dispense operations and Output ports to yield output or to vent waste. Each input fluid will have at least one input port.

3.1.1 Input

Inputs to scheduling algorithm are

- A Bio-assay represented as a DAG G=(V,E), V represents different bio-assay operations and E represents dependencies between them.
- Module library consists of number of reconfigurable modules like mixers, heaters or detectors etc. $(M_1, M_2, ..., M_q)$ and Input/Output ports $(I_1, I_2, ..., I_e, O_1, O_2, ..., O_f)$.
- Each operation's execution time $(E_1, E_2, ..., E_v)$ where $v = |V|$.
- The number of distinct input fluids $(F_1, F_2, ..., F_x)$
- Number of droplets allowed to be stored on a module at any time-step t is (k)

3.1.2 Output

Each operation in DAG will be assigned a starting and finishing time slot. Final task's finishing time is called as assay length or assay execution time.

3.1.3 Objective

The objective of scheduling algorithm is to reduce the assay length or assay execution time under resource constraints and adhering to operation dependencies as depicted in DAG.

$$Obj = min\{max_{u \in V}(AFT(u))\} \quad (1)$$

Assay length is the Actual Finish time of the last executing operation or task. Scheduling objective is to decrease the assay length, that is to minimize the finish time of the last executing task in assay.

3.1.4 Droplet Storage

The droplet produced by operation u must be stored for time interval H(u), if any of it's successor operation not starting immediately after it's finishing time. Some work module must be available to store u during H(u).

$$H(u) = \{ \max_{v \in succ(u)} AST(v)\} - AFT(u) \quad (2)$$

3.1.5 Legality

A legal schedule must satisfy precedence constraints as specified in DAG.

$$\forall (u,v) \in E, AST(v) \geq AFT(u) \quad (3)$$

At any time-step t in the schedule, the number of operations scheduled should not exceed the number of available work modules present on the chip.

3.2 Scheduling Assumptions

In DMFB scheduling, we have to schedule the different DMFB operations to particular available modules on the chip. Reconfigurable nature of DMFB is any operation of the assay can be performed anywhere on the chip (any module of DMFB). The execution time of mixing or detecting operation depends on the size of the mixer or detector module. If the size of the module is large then execution time will be less and if the size of the module is small then execution time will be more (Su and Chakrabarty, 2004). We can conclude that execution time is inversely proportional to the size of the module. This property of DMFB causes problems, those are scheduling cannot be separated from placement and placement cannot be separated from scheduling (O'Neal et al., 2012). Suppose if we start with a placement then in the middle of execution on demand if we change the size of the module then the placement that we start with becomes illegal. Hence, in scheduling the main assumption is all the modules are of fixed size, they don't change their size in the middle of assay execution. That is the reason we will declare the module library, that is a number of modules and their sizes and their execution time before we start scheduling.

The second assumption is all the scheduling operations we perform are non-preemptive in nature unlike in digital computing. Once a droplet is moved on to chip for an operation, it cannot be taken back to reservoirs at any cost. Droplet has to wait, if any obstacle in its path or if any further move of that droplet causes deadlock.

The third assumption is unlike in parallel or distributed computing there is no communication cost between modules or operations of the chip because in distributed system processors are arranged at different places and operate together, there is a need for communication between different processors and that communication will take a reasonable amount of time. But in Microfluidic Biochips as it is a single chip and modules are virtual processors there is no need for communication between them. Hence, there will be no communication cost between modules (Su and Chakrabarty, 2004).

The fourth assumption is that single chip is used for both assay operations and storage. In distributed computing processors and memory are decoupled and treated as different resources, But in Microfluidic Biochips any part of the chip can be used for any operation like mixing, detection or storage at different times. Here we just designate some selected area of a chip to function as different modules as a mixer, detector, heater and storage of droplets pertaining to fluidic constraints for droplet routing (Su et al., 2006). Work modules are derived from chip size. Means based on chip size we deterministically decide what are the optimal number of work modules we can put in biochip (Micheli, 1994).

4 HEFT BACKGROUND

Heterogeneous Earliest Finish Time (HEFT) is a scheduling algorithm used for achieving high performance with low complexity in heterogeneous distributed systems where there are different heterogeneous processors available for solving tasks which are interdependent. Operations and the interdependencies between operations are shown using a DAG. So, HEFT schedules those tasks on the available processors in such a way that Actual Finish Time of the exit tasks is minimized as low as possible using the available processor set. HEFT follows insertion based scheduling policy which means that if two or more tasks are scheduled on a processor p_j then when we are scheduling another task on that processor, then we will search for a gap between the two scheduled processes on that processor if that gap is adequate enough to accommodate the execution time requirement of that process then that process will be inserted between those already scheduled processes. For example, a processor p_j is been assigned to task t_i then processor p_j will look for idle slots in its own schedule such that it will calculate the difference between EFT and EST of every two consecutive tasks. If that difference is greater than or equal to the execution time w_i then that task t_i will be inserted in that idle slot or gap on processor p_j. If task t_i cannot find any idle slot on processor p_j then it will be inserted after the last scheduled task on p_j. When first assigning a processor to a task, HEFT will look for a processor which will take least execution time for that process or in other terms look for a processor which will give the earliest finish time for that task (Topcuoglu et al., 2002).

4.1 Implementation Differences

DMHEFT modifies HEFT to work for DMFB environments by modifying DAG attributes and calculations according to DMFB specifications. DMHEFT don't use insertion based scheduling policy as HEFT does. This reduces overhead of checking all the idle gaps on already scheduled processor. DMHEFT uses least recently used policy when allocating modules to operations to avoid deadlocks. In HEFT scheduling sequence will be generated once, but in DMHEFT we use Candidate list and it is modified after every iteration. Other modifications and detailed algorithm is described in Section 5.

5 HEFT FOR DIGITAL MICROFLUIDICS

Heterogeneous Earliest Finish Time for Digital Microfluidics (DMHEFT) is a scheduling algorithm which is a heuristic based list scheduling method to produce better assay execution times. As it is a list scheduling heuristic the whole process divides into two phases operation prioritization and module allocation to a particular operation based on operation sequence prioritized for scheduling obtained in the first phase. To do the above mentioned two phases we need some heuristics which are mentioned below and based on attributes used in (Topcuoglu et al., 2002).

5.1 Graph Attributes used in DMHEFT

A process of any biochemical analysis in digital microfluidics is represented by a DAG G=(V,E) where V represents a set of different operations and E represents a set of dependencies between operations like operation v_3 cannot be completed before operation v_1 and operation v_2. Let us say that there are v operations namely $v_1, v_2, v_3, ..., v_j$ and q mixer modules are there in DMFB namely $p_1, p_2, p_3, ..., p_q$. Any operation with no parent (predecessor) is called as an entry operation and operation with no child (successor) is called as an exit operation. Some distributed systems require single entry single exit operation system but in digital microfluidics we can have multiple entry and multiple exit DAGs. As the first phase is task prioritization, to do that we use upward rank of a node. Upward rank of a node in a graph is the length of the longest path from that particular node to exit task including execution time of that particular operation. Upward rank of a node in the graph is calculated by summing up execution time of that operation and rank of the successor which is having the maximum

rank value in all it's successor set. Upward rank of a node is calculated recursively.

$$rank_u(v_i) = w_i + \max_{v_j \in succ(v_i)} (rank_u(v_j)) \quad (4)$$

Where $succ(v_i)$ is the set of successor nodes of operation v_i, w_i is the execution time of operation i. As the exit tasks don't have any successor, upward rank for that nodes is only the execution time of that particular operation.

$$rank_u(v_{exit}) = w_{exit} \quad (5)$$

Before we can get final schedule (Actual Start Time $AST(v_i, p_j)$ and Actual Finish Time $AFT(v_i, p_j)$) we take help of partial schedule: (Earliest Start Time $EST(v_i, p_j)$ and Earliest Finish Time $EFT(v_i, p_j)$). $AST(v_i, p_j)$, $AFT(v_i, p_j)$ are Actual start and finish time of operation v_i on processor p_j. $EST(v_i, p_j)$, $EFT(v_i, p_j)$ are earliest possible execution start and finish times of operation v_i on processor p_j. EST for first tasks we execute on chip is zero because there are no other operations executing on chip and they can start immediately whenever they arrive.

$$EST(v_{entry}, p_j) = 0 \quad (6)$$

EST for other nodes excluding entry nodes will be calculated recursively. EST is maximum value between processor available time and maximum AFT of all predecessor nodes of node v_i. avail[j] is the time at which mixer will be ready for execution of another task if it is executing any task or any tasks are queued on that mixer for execution. In other words if operation v_k is already executing on processor p_j then avail[j] is the AFT of v_k. The other inner max block in (7) refers to maximum AFT among all predecessor of node v_i. $pred(v_i)$ refers to set of all predecessor nodes of particular node v_m. If you want to calculate EST of particular node v_m then all it's predecessor nodes should have been already scheduled.

$$EST(v_i, p_j) = \max\{\max_{v_m \in pred(v_i)} (AFT(v_m)), avail[j]\} \quad (7)$$

$EFT(v_i, p_j)$ is Earliest execution Finish Time of operation v_i on mixer p_j. It is summation of $EST(v_i, p_j)$ and execution time of operation v_i on mixer p_j. EFT also calculated recursively and if EFT of particular node has to be calculated all its immediate predecessor operations should have already been scheduled.

$$EFT(v_i, p_j) = w_{i,j} + EST(v_i, p_j) \quad (8)$$

After a operation is scheduled on a mixer then we can get AST and AFT of that task. Main objective of scheduling is to reduce the AFT of exit task or the last task in the prioritized scheduling sequence generated using upward rank values.

$$AssayLength = \max\{AFT(n_{exit})\} \quad (9)$$

5.2 Proposed Algorithm HEFT for Digital Microfluidic Biochips

DMHEFT is an operation scheduling algorithm based on algorithm proposed in (Topcuoglu et al., 2002) for task scheduling on heterogeneous processors. DMHEFT is a heuristic based list scheduling algorithm. As it is list scheduling algorithm it divides the process into two main phases: *Operation prioritizing phase* to prioritize operations based on upward rank values calculated and *Module selection phase* for selecting a best available module for a particular operation in scheduling sequence.

Operation Prioritizing Phase: In DMFB Scheduling we assume that all operations of a bio-assay are available before we start scheduling, there are no operations coming in between the intermediate stages of assay execution. As all operations are available beforehand we will prioritize operations so as to let the system know which operation sequence it has to execute. To prioritize operations we first calculate upward rank of each operation starting from nodes in last level of a DAG to first level of DAG according to (4),(5). Now priority of a node is the respective upward rank of that node. Now as we have priority of all nodes we sort the operations based on their priorities (upward rank values) in a decreasing manner, this will be scheduling sequence that has to be executed by DMFB. When two or more nodes have same priority then tie-breaking will be done randomly. The prioritized scheduling sequence is a topological order of operations which abides all precedence constraints in DAG.

Module Selection Phase: From the prioritized scheduling sequence of operations select operation one by one and allocate an available module which has minimum mixing time/detection time for that particular operation. DMHEFT uses a different strategy for selecting a module for operations unlike original HEFT which uses insertion based scheduling. Generally as mixer area (number of cells) increases, mixing time decreases and vice versa. So, every time a mixing operation comes it selects a larger mixer available. Normally in HEFT for multiple processors with intercommunication between them uses insertion based scheduling policy for selecting a processor for a task. But in Microfluidic Biochips there is no communication between modules needed, because they all are placed on a single chip. As there is no communication between modules needed, there will not be any gaps between two scheduled operations on a module except for precedence constraints (an operation cannot

start until an operation finishes). An operation may have better EST on module p_i compared to module p_j but, we have to consider only EFT because module p_i with lower EST for operation v_x may be a smaller module and module p_j with higher EST for operation v_x is a larger module and even if starts late also it will take less time to execute. Another case is that if two operations v_x, v_y have same EFT on module p_j then the module that is not used recently will be selected because if scheduler keep on assigning a module because it is having less mixing time, droplets may get congested and may lead to deadlocks and stalling of droplets which causes overhead in routing. Algorithm of DMHEFT for digital microfluidic biochips is given in Algorithm 1.

Algorithm 1: DMHEFT (HEFT for Digital Microfluidics).

```
Compute rank_u for all operations by
  traversing DAG bottom-up manner starting
  with exit tasks;
Assign child operations of nodes with EST=0
  to candidate list (CL);
while ∃u ∈ CL unscheduled do
    Sort the operations in CL in the order of
      their rank_u in decreasing manner;
    Take the current operation v_i from CL;
    for Each Available module p_k in module
      library p_k ∈ Q do
        Compute EST(v_i, p_k), EFT(v_i, p_k)
    end
    Assign operation v_i to Module p_j that
      gives minimum EFT;
    Remove operation v_i from CL;
    Add v_i's successors to CL;
    ∀(u,v) ∈ E if AST(v) > AFT(u) then
        Store droplet from Operation u in
          available module for H(u) interval
    end
end
```

5.3 An Example

Here our proposed DMHEFT method has been explained with an example on In-vitro diagnostics assay with 2 samples and 3 reagents on a 15×13 DMFB with four 3×4 modules. S_1 refers to Serum, S_2 refers to Plasma and R_1, R_2, R_3 refers to Glucose, Lactate, Pyruvate respectively. All Input/Dispense operations will take 2s, $M_1, M_2, M_3, D_1, D_2, D_3$ will take 3s and $M_4, M_5, M_6, D_4, D_5, D_6$ will take 5s and all Output operations are instantaneous.

As In-vitro2 is a DAG with 6 connected components/trees, priorities are calculated differently for each connected component. For three trees we will get priorities as 8,8,6,3,0 and for rest three trees we get 12,12,10,5,0. Operations are added to ready queue based on root nodes priority. Every tree is processed completely if it has been taken for execution because if we stop that in between, we have to store that droplet in some working module until the next operation in that tree uses that droplet. if we allow interleaving then there will be more droplets stored on chip. Operations will be executed based on the availability of input ports because we cannot schedule two operations simultaneously which uses same input port. So, if two operations with same priority which uses same input port are in ready queue then operation which uses different input port will be chosen for simultaneous execution. And DMHEFT follows greedy strategy in executing operations so always allow the highest rank valued operation when two operations with different rank values are queued. DMHEFT will not schedule dispense operations until the subsequent mix operations gets scheduled.

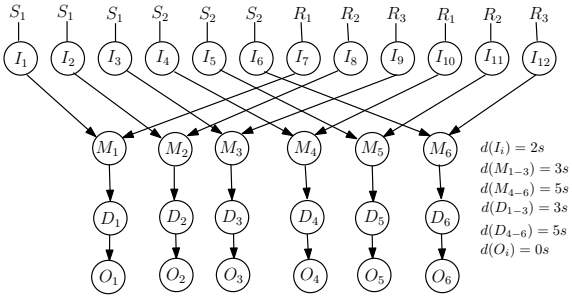

Figure 2: In-vitro Assay with 2 Samples, 3 Reagents.

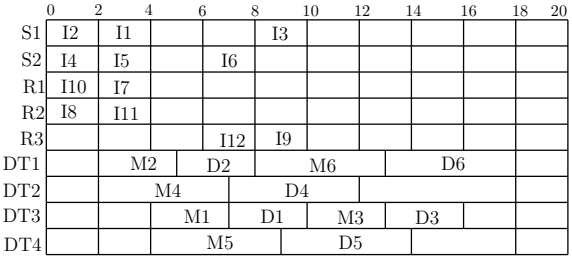

Figure 3: Gantt chart for In-vitro Assay with 2 Samples, 3 Reagents.

6 EXPERIMENTAL RESULTS

6.1 Simulation Setup

DMHEFT algorithm is implemented in C++ programming on Intel(R) Core i7 3.40 GHz processor with 4GB RAM and 500GB hard disk. DMHEFT's

Table 1: Scheduling Assay Lengths for Different Benchmarks by Various Scheduling Algorithms.

Benchmark	No.of Operations	Scheduling Assay Length (in time-steps)						
		LS	PS	FDLS	GA1	GA2	ILP	DMHEFT
PCR	15	12	12	12	12	12	12	12
In-Vitro 1	16	15	15	15	15	15	15	15
In-Vitro 2	24	21	19	18	18	19	19	19
In-Vitro 3	36	25	27	25	23	23	23	25
In-Vitro 4	48	31	33	31	29	29	29	31
In-Vitro 5	64	45	45	41	39	39	39	41
Protein	103	198	187	182	179	194	180	198

scheduling assay lengths are compared with 6 existing algorithms, those are ILP based formulation (Ricketts et al., 2006), LS (Grissom and Brisk, 2012a), PS (Grissom and Brisk, 2012b), FDLS (O'Neal et al., 2012), GA1 (Ricketts et al., 2006), GA2 (Su and Chakrabarty, 2008). All these scheduling algorithms were already implemented in UCR DMFB Static Simulator (Grissom et al., 2012), (Grissom et al., 2015) and made publicly available. We executed all of these scheduling algorithms on three famous benchmarks for DMFB synthesis as provided in (Su and Chakrabarty, 2006) namely PCR, In-Vitro Diagnostics with 5 different variants and Protein synthesis assay. In-Vitro Diagnostics assays have 5 different assays based on number of samples which varies from 2 to 4 and number of reagents in reactions which also varies from 2 to 4. All of these assays are represented in DAG form and readily available in UCR DMFB Static Simulator. Each bioassay requires different number of I/O ports and has different number of operations like mixing, storage, split, heating or detection.

All of the scheduling algorithms are implemented on a 15×13 DMFB with different configuration (I/O ports and Module library) based on assay. PCR assay has 8 input ports and 1 output ports and In-Virto assay has 12 input ports and 1 output port and Protein Synthesis assay has 5 input ports and 1 output port and all of the three assays have four 3×4 reconfigurable modules placed on 15×13 DMFB. Modules are placed according to virtual topology presented in (Grissom and Brisk, 2012a), (Grissom and Brisk, 2014) so that in future no placement and routing failures should occur. Number of droplets allowed to be stored on each module(k) are 4.

6.2 Time Complexity

DMHEFT runs with complexity $O(v*q)$ where v is number of bioassay operations and q is number of modules in the chip. Where $O(v)$ is for calculating priority of every vertex/operation in DAG and $O(v*q)$ is for selecting best available module p_j from a module library Q for operation v_i.

6.3 Assay Length Results

Table 1 reports the scheduling assay lengths in time steps (1 time step= 1 s) of different scheduling algorithms including DMHEFT on each of the above mentioned benchmarks. All In-vitro assays got same assay length for k=2 and k=4 but, Protein assay got lesser assay lengths when k=4 than k=2. And all assays gets lower assay lengths if we increase the number of modules. DMHEFT performed well on PCR, In-Vitro assays but not so compromising over Protein assay. DMHEFT generates better schedules than LS in two cases and produces schedules no longer than LS in all other cases. DMHEFT runtime complexity is also polynomial as LS. Compared to PS, DMHEFT produces shorter schedules in three cases and in other three cases same length schedules and in one case slightly larger schedule than PS. PS only works on trees/forests but, DMHEFT can be applied to any kind of DAG. Compared to FDLS, DMHEFT produces five equal length schedules and two longer schedules. But, DMHEFT rank calculation mechanism is simpler than FDLS and that is why it runs faster than FDLS. And Compared to GA1, GA2 and ILP DMHEFT produces equal length schedules in few cases and longer schedules in most of the cases because obviously they run for more amount of time and verify nearly 2000 random generated schedules and pick local optimal solution and ILP also runs for a time limit of four hours to get these assay lengths. But DMHEFT within much shorter time gives schedules with little difference in the assay length compared to GA1, GA2, ILP.

7 CONCLUSION

We proposed a simple scheduling algorithm DMHEFT for digital microfluidics which is based

on HEFT proposed by (Topcuoglu et al., 2002). DMHEFT is a list scheduling based greedy heuristic works in two phases, operation prioritization for generating scheduling sequence and module allocation. DMHEFT schedules each vertex only once unlike ILP and iterative improvement methods. It uses upward rank value for prioritizing the operations and determining scheduling sequence. Module with earliest finish time is chosen for operations in the queue. DMHEFT uses Least Recently Used policy when scheduling operations on modules to avoid deadlocks in future steps. DMHEFT can schedule all types of DAGs unlike PS which only works on trees/forests. Produces shorter or equal length schedules than LS in all cases. Gives shorter or equal length schedules than PS except in one case. DMHEFT's rank calculation mechanism is simpler than FDLS's, that is why it runs faster than FDLS. Although DMHEFT produces longer schedules than GA1, GA2 and ILP it runs much faster than them with small difference in assay lengths.

REFERENCES

Bhardwaj, T. and Jha, S. K. (2018). Microfluidic platform for aptamer based fluorimetric analysis of analytes. In *Proceedings of the 11th International Joint Conference on Biomedical Engineering Systems and Technologies - Volume 1: BIODEVICES, (BIOSTEC 2018)*, pages 218–223. INSTICC, SciTePress.

Ding, J., Chakrabarty, K., and Fair, R. B. (2001). Scheduling of microfluidic operations for reconfigurable two-dimensional electrowetting arrays. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20(12):1463–1468.

Grissom, D. and Brisk, P. (2012a). Fast online synthesis of generally programmable digital microfluidic biochips. In *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pages 413–422. ACM.

Grissom, D. and Brisk, P. (2012b). Path scheduling on digital microfluidic biochips. In *Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE*, pages 26–35. IEEE.

Grissom, D., Curtis, C., Windh, S., Phung, C., Kumar, N., Zimmerman, Z., Kenneth, O., McDaniel, J., Liao, N., and Brisk, P. (2015). An open-source compiler and pcb synthesis tool for digital microfluidic biochips. *INTEGRATION, the VLSI journal*, 51:169–193.

Grissom, D., O'Neal, K., Preciado, B., Patel, H., Doherty, R., Liao, N., and Brisk, P. (2012). A digital microfluidic biochip synthesis framework. In *VLSI and System-on-Chip, 2012 (VLSI-SoC), IEEE/IFIP 20th International Conference on*, pages 177–182. IEEE.

Grissom, D. T. and Brisk, P. (2014). Fast online synthesis of digital microfluidic biochips. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 33(3):356–369.

Micheli, G. D. (1994). *Synthesis and optimization of digital circuits*. McGraw-Hill Higher Education.

O'Neal, K., Grissom, D., and Brisk, P. (2012). Force-directed list scheduling for digital microfluidic biochips. In *VLSI and System-on-Chip (VLSI-SoC), 2012 IEEE/IFIP 20th International Conference on*, pages 6–pp. IEEE.

Pollack, M. G., Shenderov, A. D., and Fair, R. (2002). Electrowetting-based actuation of droplets for integrated microfluidics. *Lab on a Chip*, 2(2):96–101.

Ricketts, A. J., Irick, K., Vijaykrishnan, N., and Irwin, M. J. (2006). Priority scheduling in digital microfluidics-based biochips. In *Proceedings of the conference on Design, automation and test in Europe: Proceedings*, pages 329–334. European Design and Automation Association.

Schulte, T. H., Bardell, R. L., and Weigl, B. H. (2002). Microfluidic technologies in clinical diagnostics. *Clinica Chimica Acta*, 321(1-2):1–10.

Srinivasan, V., Pamula, V. K., Pollack, M. G., and Fair, R. B. (2003). Clinical diagnostics on human whole blood, plasma, serum, urine, saliva, sweat, and tears on a digital microfluidic platform. In *Proc. MicroTAS*, pages 1287–1290.

Su, F. and Chakrabarty, K. (2004). Architectural-level synthesis of digital microfluidics-based biochips. In *Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design*, pages 223–228. IEEE Computer Society.

Su, F. and Chakrabarty, K. (2006). Benchmarks for digital microfluidic biochip design and synthesis. *Duke University Department ECE*.

Su, F. and Chakrabarty, K. (2008). High-level synthesis of digital microfluidic biochips. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 3(4):1.

Su, F., Hwang, W., and Chakrabarty, K. (2006). Droplet routing in the synthesis of digital microfluidic biochips. In *Design, Automation and Test in Europe, 2006. DATE'06. Proceedings*, volume 1, pages 1–6. IEEE.

Topcuoglu, H., Hariri, S., and Wu, M.-y. (2002). Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Transactions on parallel and distributed systems*, 13(3):260–274.

Verpoorte, E. and De Rooij, N. F. (2003). Microfluidics meets mems. *Proceedings of the IEEE*, 91(6):930–953.