

Chapter 12

MODELING AND CONTROLLING PARALLEL TASKS IN DROPLET-BASED MICROFLUIDIC SYSTEMS

Karl F. Böhringer
University of Washington

Abstract: *Abstract*—This paper presents general, hardware-independent models and algorithms to automate the operation of droplet-based microfluidic systems. In these systems, discrete liquid volumes of typically less than 1 μ l are transported across a planar array by dielectrophoretic or electrowetting effects for biochemical analysis. Unlike in systems based on continuous flow through channels, valves, and pumps, the droplet paths can be reconfigured on demand and even in real time. We develop algorithms that generate efficient sequences of control signals for moving one or many droplets from start to goal positions, subject to constraints such as specific features and obstacles on the array surface or limitations in the control circuitry. In addition, an approach towards automatic mapping of a biochemical analysis task onto a droplet-based microfluidic system is investigated. Achieving optimality in these algorithms can be prohibitive for large-scale configurations because of the high asymptotic complexity of coordinating multiple moving droplets. Instead, our algorithms achieve a compromise between high run-time efficiency and a more limited, non-global optimality in the generated control sequences.

Key words: Droplet-based microfluidic system, digital microfluidic system, parallel manipulation, lab on a chip (LOC)

1. INTRODUCTION

Advances in microfabrication and microelectromechanical systems (MEMS) over the past decades have lead to a rapidly expanding collection of techniques to build systems for the handling and analyzing of very small quantities of liquids (see, e.g., [1, 2]). These microfluidic systems typically

consist of sub-millimeter scale components such as channels, valves, pumps, and reservoirs, as well as application-specific sensors and actuators. Microfluidic devices hold great promise, for example for novel fast, low-cost, portable, and disposable diagnostic tools. Applications include the massively parallel testing of new drugs, the on-site, real-time detection of toxins and pathogens, and PCR (polymerase chain reaction) for DNA sequence analysis. They usually operate with continuous flows of liquids, in analogy to traditional macro-scale laboratory set-ups, and can integrate all functionality into a complete *lab-on-a-chip (LOC)* or *bio-system-on-a-chip (bioSOC)*.

More recently, an alternative LOC approach has gained momentum using individual droplets, with volumes usually in the sub-microliter range. In these *droplet-based microfluidic systems*, droplets are generated, transported, merged, analyzed, and disposed on planar arrays of addressable cells; therefore they are also sometimes called *discrete* or *digital microfluidic systems*, and conveniently abbreviated *DMFS*. This architecture for microfluidic systems is attractive because of (a) greater flexibility – analyte handling may be reconfigured simply by re-programming rather than by changing the physical layout of the microfluidic components; (b) high droplet speeds – reportedly up to 25cm/s [3, 4]; (c) no dilution and cross-contamination due to diffusion and shear-flow; and (d) the possibility for massively parallel operation.

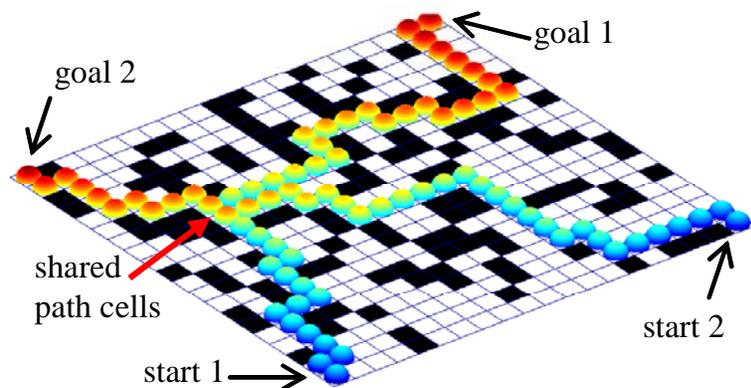


Figure 12-1. Two droplets moving in parallel on a DMFS consisting of a 20×20 array with obstacle cells (marked black). The droplets start from cells (1,1) and (20,1) and move to cells (20,20) and (1,20), respectively. Change in droplet color indicates the elapsed time. The droplets share cells (5,13) and (5,14) on their path but their coordinated schedule prevents any conflicts.

The DMFS approach assumes that it is advantageous to shift complexity from microfluidic hardware to control software. Therefore, for a DMFS to live up to its promise, it must be accompanied by a complementary set of software tools such that its usage can be largely automated. This includes software that helps the user to map a biochemical analysis protocol onto a given DMFS; as a specific subproblem, algorithms that automatically plan and schedule routes for simultaneous droplet motion are required. Fig. 12-1 shows a schematic example where two droplets move in parallel across a DMFS while circumnavigating numerous obstacles. Developing the formalisms, models, and control strategies for such automated droplet manipulation tasks is the goal of this paper.

Processing large numbers of discrete droplets simultaneously on an integrated microchip indicates a similarity to electronic digital circuits, giving rise to *microfluidic circuits* [5, 6]. This analogy also suggests that algorithms for layout, routing, and scheduling of droplet paths in a DMFS are computationally expensive, i.e., NP-hard.

Thus, this paper is organized as follows. Section 2 reviews background material on DMFS hardware, and discusses related work in control algorithms. Section 3 introduces a formal DMFS model and problem specification. Section 4 presents algorithms for coordinating parallel droplet motion on a DMFS, and investigates trade-offs between run-time efficiency and optimality. Section 5 extends these algorithms to allow for changes of the droplet type during DMFS operation, and develops an approach to automatically transform a laboratory protocol into a sequence of DMFS tasks. Section 6 concludes the paper with a summary and an outlook on future work.

2. RELATED WORK

Transferring a laboratory task such as DNA analysis, clinical diagnostics, or detection and manipulation of bio-molecules into a lab-on-a-chip system is a complex endeavor that can involve multiple challenges: the design of microfluidic hardware including sensing and actuation mechanisms for liquid analytes; the use of specialized techniques and materials such as modification and functionalization of surfaces with monolayers or antibodies; and the development of algorithms for layout and control of massively parallel microfluidic circuits.

Lab-on-a-chip design and manufacture has become an extensive research area with dedicated conferences (e.g., [7]) and journals (e.g., [8, 9]). This paper, however, focuses on the software aspects, and assumes a device

model that abstracts away from details of the physical implementation. Here, we discuss very briefly important aspects of droplet-based microfluidics that are relevant to motivate and justify our modeling assumptions.

2.1 Droplet Transport Techniques

The most successful conventional droplet based system in the life sciences is arguably the fluorescence activated cell sorter (FACS) [10-12], a machine that can sort droplets containing single cells at rates well above 100kHz. It generates charged droplets, analyzes them in free flight via a laser fluorescence detection system, and sorts them accordingly via a modulated electrostatic field. Lab-on-a-chip FACS systems exist but so far work at much lower processing rates [13-15].

In micro-scale lab-on-a-chip systems, droplets can be moved across a planar surface effectively with a variety of techniques, including electric fields (e.g., [3, 16-19]), surface acoustic waves (e.g., [20-22]), thermocapillary and Marangoni effects (e.g., [23, 24]), electrochemical surface modulation (e.g., [25]), conformational changes in molecular surface layers (e.g., [26]), or gradients in surface chemistry (e.g., [27, 28]) and texture (e.g., [29, 30]). For this paper, droplet transport with high speed, accuracy, and full software control is essential, making electric fields the most suitable approach; hence we briefly discuss the two main techniques in this realm, dielectrophoresis and electrowetting [31, 32].

2.1.1 Dielectrophoresis

In dielectrophoresis (DEP), neutrally charged objects are first polarized by an electric field, and then experience a net force due to the field. This force can only be non-zero if a field gradient exists, i.e., the positively and negatively polarized regions of the object occupy areas of different field strengths. If the object has stronger polarization than the surrounding medium then it is pulled towards the areas of higher field strength (this is called positive DEP), but if the surrounding medium has higher polarization, then the object is pushed towards areas of lower field strength (negative DEP). DEP can be considered the electrostatic analogy of induced magnetism. Common examples for DEP are charged clothes that attract (neutral) lint particles. More information on dielectrophoresis can be found, e.g., at [33]. A DMFS system employing DEP with more than ten thousand array elements was demonstrated in [34].

2.1.2 Electrowetting

Electrowetting on dielectric (EWOD) exploits the decrease of contact angle that an aqueous droplet on a dielectric surface experiences when exposed to an electric field. If the field is localized at only one side of the droplet, then the difference in contact angle causes a pressure differential in the droplet, which drives it towards the region of higher field strength. Electrowetting and its applications in microfluidics have been investigated by several groups, including [3, 16, 17, 35, 36].

2.2 Droplet Transport Planning and Scheduling

Finding the optimal plan to generate, store, move, merge, split, and dispose multiple droplets on a droplet-based microfluidic system combines general path planning and scheduling with the more application-specific task of analyte droplet handling. Various researchers have studied parts of the overall problem and have shown important results on algorithmic solutions and their computational complexity.

One possible approach to this problem can be taken when the paths of the droplets are considered given a priori. This assumption leads to a scheduling problem, where the array cells en route are the limited resource that must be shared among different droplets. Griffith and Akella [37] show a solution with standard optimization tools guided by some user input, building on related work in coordinating multiple articulate robots [38, 39]. Many more references to related work in the areas of robot motion planning, flexible manufacturing systems, queuing theory and networking are also given in [37].

A related technique was used by Ding, Zhang, et al. [5, 6, 40] who attack the problem from the VLSI design perspective. As in [38, 39], this approach leads to an integer programming formulation. Both groups show NP-hardness of the scheduling problem even for fixed droplet routes.

VLSI circuit routing techniques could also be employed, which address the path planning problem but do not apply directly to the inherently two-dimensional layout of the droplet-based microfluidic platform.

This paper takes a different approach, by permitting the droplet paths to be chosen freely (except for constraints defined by the microfluidics hardware). Each droplet is interpreted as a point robot moving in a discrete two-dimensional configuration space. Under this assumption, path planning of the droplets becomes a motion planning problem with multiple moving robots. Erdmann and Lozano-Pérez showed in 1987 that this problem is NP-hard, but presented an algorithm that may find a good solution in polynomial

time [41]. Their approach assigns priorities to each robot (droplet) and generates paths successively, starting with the highest priority robot. Lower priority robots consider higher priority robots as time-varying obstacles that must be avoided. The algorithm is not complete, and generated solutions depend on the priority ranking of the robots and may not be optimal.

In [42], this author described the problem as a graph search, and suggested search techniques such as A*. Even though this brute-force approach, unlike the other work mentioned above, guarantees optimality and completeness, it is not practical for larger scale problems because of its computational complexity, which is exponential in the number of moving droplets. Reference [43] introduced a formal problem definition and showed initial results with a more efficient approach based on Erdmann's algorithm [41].

3. DMFS FORMAL HARDWARE SPECIFICATION

Let us briefly review the most important physical properties and design parameters of a droplet-based microfluidic system (DMFS). Motivated by these characteristics, we can then develop an abstract DMFS model that captures the essential operational features without depending on specific implementation details.

3.1 DMFS Design Specifications

- Layout: Typically, a DMFS consists of a planar, rectangular array A with $m \times n$ cells (but, e.g., an arrangement of hexagonal cells would also be possible).
- Control circuitry: Various addressing schemes are possible to activate individual cells in a DMFS. In different physical implementations of DMFS, we can distinguish, for instance, individually addressable electrodes for each cell (e.g., [36]), or simpler row/column addressing (e.g., [44, 45]). For the latter, entire rows and columns are activated, and the droplet is attracted to a neighboring cell $A(x,y)$ only if it lies at the intersection of active column x and row y .
- Parallelism: The DMFS controller may be capable of simultaneous activation of more than one cell, which will allow simultaneous motion of multiple droplets. The total number of addressable cells may be limited by a number significantly smaller than $m \times n$.
- Location of cells with special functions: Droplet generators, reservoirs, cells for merging and splitting of droplets, sensors, waste, etc. may

require dedicated cells with special embedded hardware. These cells may not be available when planning a droplet path across the array.

These specifications provide a physical framework within which a DMFS can operate. Based on this framework, we can establish a formal description of the problem of controlling droplets in a DMFS. Once a sufficiently general DMFS model exists, we can investigate algorithmic solutions at an abstract level without worrying about the varying details of specific hardware implementations.

3.2 Abstract DMFS Specification

A droplet-based microfluidic system is specified by the droplets on the DMFS array, the DMFS hardware itself, and the task to be performed.

3.2.1 Droplets

Droplets are described by their type T and their volume V . We are assuming here that all droplets in the DMFS have the same volume, except when two droplets have been merged. Therefore, we require that a merge operation is always immediately followed by a split operation that restores the original droplet volumes.

The droplet type T is a subset of all elementary droplet types, which we describe in general as a set $\mathcal{T} = \{T_1, T_2, T_3, \dots\}$; thus, T is an element of the power set of \mathcal{T} , $T \in \mathcal{P}(\mathcal{T})$. For example, if T_1 represents “deionized water”, T_2 “methanol”, and T_3 “isopropanol”, then a droplet of type $T = \{T_1, T_3\}$ describes a mixture of DI water and IPA. Note that this convention provides a simple representation of mixed droplets, but does not keep track of sample concentrations. If needed, different concentrations could be represented as different elementary types.

3.2.2 DMFS Arrays and Tasks

The DMFS consists of an array A of $m \times n$ cells. Each cell in the array is either empty or occupied, which we represent by specifying its droplet type T . Thus, the DMFS can be described by $A(x,y) = T_{x,y}$ for $(x,y) \in \{1 \dots m\} \times \{1 \dots n\}$ and $T_{x,y} \subseteq \mathcal{T}$. As a special case, $T_{x,y} = \emptyset$ indicates an empty cell. We call $A \in \mathcal{P}(\mathcal{T})^{m \times n}$ the state of the DMFS.

The location of a droplet can be specified by the pair $(x,y) \in \{1 \dots m\} \times \{1 \dots n\} = \mathcal{C}$; thus, \mathcal{C} is the configuration space [46] of a single droplet, and \mathcal{C}^d is the configuration space of d droplets, which we also call the droplet placement of the DMFS.

Time is assumed to be a discrete counter $t \in \{0, 1, 2, \dots\}$, i.e., transitions in the array occur in integer time steps from t to $t+\Delta t$, where $\Delta t = 1$ unless noted otherwise. We write A_t to refer to the state of the array at a specific time t .

At this point, we can already outline the definition of a DMFS task: given a start state $A_s \in \mathcal{R}(\mathcal{S})^{m \times n}$ and a goal state $A_g \in \mathcal{R}(\mathcal{S})^{m \times n}$, we need to find a timed sequence of valid transitions that results in the desired droplet motions from A_s to A_g . Various kinds of transitions exist; they include simple droplet transport from cell to cell, but also droplet generation, disposal, merging, and splitting. In addition to motion, droplets may also be modified by operations on cells that change their type. All these operations are chosen from the following list of valid droplet transitions, which are usually associated to specific cells or groups of cells on the array:

- **Droplet generation:** For $(x,y) \in \mathcal{C}$ and some $T \in \mathcal{R}(\mathcal{S})$, a droplet is generated at coordinate (x,y) if $A(x,y) = \emptyset$ at time t and $A(x,y) = T$ at time $t+\Delta t$.
- **Disposing:** Definition analogous to droplet generation.
- **Moving:** Let (x,y) and $(x',y') \in \mathcal{C}$ and $|x-x'| + |y-y'| = 1$ (i.e., $A(x,y)$ and $A(x',y')$ are directly adjacent). At time t , $A(x,y) = T$ and $A(x',y') = \emptyset$ and at time $t+\Delta t$, $A(x,y) = \emptyset$ and $A(x',y') = T$.
- **Merging:** Let (x,y) , (x',y') , and $(x'',y'') \in \mathcal{C}$ such that (x',y') and (x'',y'') are directly adjacent to (x,y) but not adjacent to each other. At time t , $A(x,y) = \emptyset$, $A(x',y') = T_1$, and $A(x'',y'') = T_2$, and at time $t+\Delta t$, $A(x,y) = T_1 \cup T_2$ and $A(x',y') = A(x'',y'') = \emptyset$, where $T_1 \cup T_2$ is the droplet type that results in merging droplet types T_1 and T_2 .
- **Splitting:** Definition analogous to merging.
- **Checking:** For $(x,y) \in \mathcal{C}$, we require that a droplet remains at $A(x,y)$ from time t to time $t+\Delta t$. This allows, for example, sensing operations to be performed that neither change the location nor the type of the droplet (e.g., fluorescence detection).
- **Changing:** For $(x,y) \in \mathcal{C}$, we define a function $f: \mathcal{R}(\mathcal{S}) \rightarrow \mathcal{R}(\mathcal{S})$ such that $A(x,y) = T_1$ at time t and $A(x,y) = T_2$ at time $t+\Delta t$, and $f(T_1) = T_2$. This allows transition operations that modify the droplet type but not its location (e.g., heating/cooling for PCR).
- **Blocking:** For $(x,y) \in \mathcal{C}$, we define a set of forbidden droplet types $\mathcal{F}_{x,y} \subseteq \mathcal{R}(\mathcal{S})$ that are not allowed on $A(x,y)$. In particular, if $\mathcal{F}_{x,y} \equiv \mathcal{S}$ then $A(x,y)$ is blocked for all droplets.

Finally, valid placement and motion of droplets on the array is subject to constraints:

- **Placement:** To avoid accidental merging of droplets, at least one empty cell is required between two occupied cells at all times, i.e., for any (x,y) and $(x',y') \in \mathcal{C}$ with $A(x,y) \neq \emptyset$ and $A(x',y') \neq \emptyset$, $|x-x'| > 1$ or $|y-y'| > 1$.

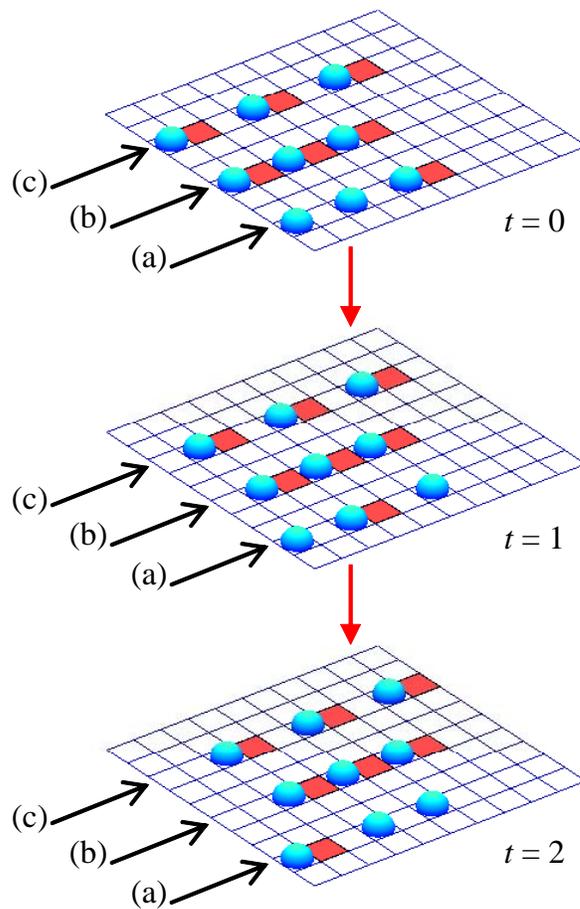


Figure 12-2. Parallel droplet transitions: Droplets (blue) and their activated neighbor cells (red squares) are shown at the instant when motion is commencing. The transitions in rows (a) and (c) are valid, but invalid in row (b) because during these transitions, two of the droplets have more than one activated neighbor cell, which could lead to unintentional splitting or merging.

- Parallel transitions: The previous constraint on placements must in particular also hold during transitions, i.e., for all pairs of droplet placements across the transition interval $[t, t+\Delta t]$ (see Fig. 12-2), except when merging or splitting is intended.

4. DROPLET PATH PLANNING

This section focuses on a central task in the control of DMFS: generating efficient paths for multiple droplets that move from a given start configuration A_s to a desired goal configuration A_g . For now, we require that the types of the droplets remain unchanged during the transition from A_s to A_g (this constraint will be removed in Section 5). We will first give a simple, complete algorithm based on A* search, but find that its computational complexity is very high (exponential in the number of droplets). We then present a more efficient algorithm for the *DMFS motion planning problem* that trades off completeness for faster execution times, while maintaining some “local” optimality guarantees.

4.1 Basic A* Search

This approach maintains a graph data structure to keep track of the droplet locations in the DMFS array. At any given time t , the state of the DMFS is described by A_t and identified with a node in this graph. A transition between two states A_t and $A_{t+\Delta t}$ defines a directed edge; this transition must conform with the conditions set forth in Section 3.2 above. Finding an optimal control strategy to transform start state A_s into goal state A_g then becomes a standard graph search problem: the shortest path between nodes A_s and A_g can be determined, e.g., using the A* algorithm known from artificial intelligence programming [47].

The A* algorithm outlined below maintains two lists of states, *Open* and *Closed*, which keep track of nodes that still need to be explored, and nodes that have already been processed, respectively. For each node, we maintain its predecessor p , the cost incurred g (i.e., number of transitions from A_s), the cost remaining h (i.e., number of transitions to A_g), and the total cost $f=g+h$. As has been widely discussed in the literature, h , which is not known in advance, can be estimated with an “admissible” heuristic function. The Manhattan metric provides such an admissible cost estimate, i.e., if droplet i at time t is at $(x_{t,i}, y_{t,i})$ and its goal is $(x_{g,i}, y_{g,i})$ then $h(t)$ can be estimated as $\sum_i |x_{g,i} - x_{t,i}| + |y_{g,i} - y_{t,i}|$.

Algorithm 1: A* for droplet path planning

Input: start state A_s , goal state A_g

Output: shortest path from A_s to A_g

```

Open ← {  $A_s$  };
Closed ←  $\emptyset$ ;
while Open  $\neq$   $\emptyset$  begin

```

```
o ← pop state with smallest f from Open;  
Q ← list of all valid motion transitions from o;           // Line 5  
for each q in Q begin  
  q.g ← o.g + 1;           // q is one step beyond o  
  q.h ← distance estimate from q to Ag;  
  q.f ← q.g + q.h;  
  q.p ← o;           // keep track of path from As via o to q  
  if q = Ag, return q;           // goal found, success  
  if not ( $\exists q' \in Open$  such that  $q' = q$  and  $q'.f < q.f$ )  
  and not ( $\exists q' \in Closed$  such that  $q' = q$  and  $q'.f < q.f$ )  
  then add q to Open;           // found new state q to be explored  
end  
  add o to Closed;           // finished exploring node o  
end  
return  $\emptyset$ ;           // search exhausted, failure
```

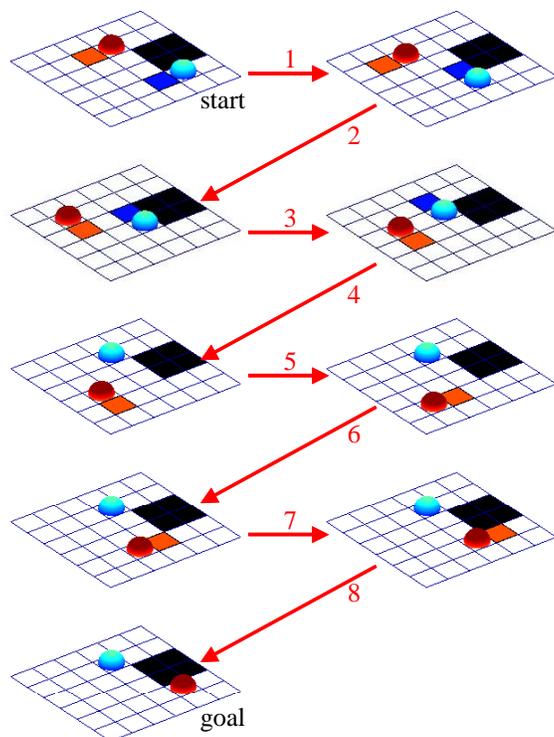


Figure 12-3. Two droplets moving simultaneously on a 6×6 DMFS array while avoiding an obstacle (black cells). The two droplets start at cells (5,2) and (4,5) and trade their places in 8 parallel transitions. The activated neighbor cells for their next transitions are also shown.

Fig. 12-3 shows a simple example where two droplets swap their position while avoiding an obstacle. The A* algorithm is guaranteed to always find an optimal solution if one exists, and indicate failure otherwise. However, the downside of this approach is its high asymptotic complexity. Suppose the number of droplets is d . In the simplest case, all are of the same type T_0 . Then the number of different placements of droplets on the array is $\binom{m}{d}$, which for modest numbers $m=n=10$ and $d=10$ yields more than 1.7×10^{13} possibilities. If all droplets are of distinct type $T_1 \dots T_d$, this number increases by $d!$ (to $\approx 6.3 \times 10^{19}$). One might hope that in practice, most of these choices need not be explored. However, at each step, d droplets offer up to $4d$ choices to be moved, assuming 4 neighbor cells per droplet. Thus, finding a strategy with s steps could mean checking up to $(4^d)^s$ choices or risk missing the solution, resulting again in astronomical numbers even for $s < 10$.

We conclude that the search graph explored with the A* algorithm has $O((mn)!)^d$ nodes and a branching factor of $O(4d)$, leading to a run-time

complexity exponential in d , which is prohibitive for any non-trivial array size with more than a few droplets.

4.2 Prioritized A* Search

The discussion above has shown that droplet path planning for DMFS has two main aspects: generating efficient droplet path plans, and finding efficient algorithms to generate these plans. Erdmann and Lozano-Pérez's [41] NP-hardness results for coordinating multiple moving objects indicate that compromises need to be made to obtain practical solutions, and completeness or optimality in motion plans has to be traded off with efficiency in plan generation. They propose to impose a priority order on the moving objects, and sequentially find "locally" optimal solutions. In our case, the order can be assigned at random, or based on application-specific guidelines (e.g., water may have lower priority than droplets containing expensive or volatile compounds):

Algorithm 2: Prioritized A* for droplet path planning

Input: start state A_s , goal state A_g , priority order for droplets

Output: path from A_s to A_g

```

 $S \leftarrow \emptyset;$  // partial prioritized solution
for all droplets  $i$  in decreasing priority order begin
  call Algorithm 1 to determine an optimal path for droplet  $i$  while
    considering all droplets with higher priority as moving obstacles and
    ignoring all droplets with lower priority;
  if solution for droplet  $i$  exists
  then add solution to  $S$ ;
  else return  $\emptyset;$  // failure
  end
return  $S;$  // success

```

Fig. 12-1 was generated using this algorithm. It eliminates the exponential complexity in d , where d is the number of droplets in the DMFS. Instead, the prioritized algorithm is linear in d . As a trade-off, (a) it is no longer complete: existing solutions may be missed; and (b) the solution may not be "globally" optimal: while each droplet i finds a "locally" optimal path among the moving droplets of higher priority, the complete solution will in general depend on the priority order and not be "globally" optimal. Thus, as was pointed out in [41], selecting the priority order can greatly influence the final solution. For instance, if a short path is important for a specific droplet

type, then it should receive high priority. However, total run time is dominated likely by low priority droplets, since they may take convoluted paths to circumnavigate all higher priority droplets. A good heuristic for assigning priorities will take these points into account, as well as other, application-specific factors. For example, droplets whose type appears frequently on the DMFS could be assigned lower priorities than rare droplet types, because it is likely that one of the abundant droplets is already close to a desired destination.

4.3 Parallel Droplet Motion

The algorithms given so far are able to generate plans with simultaneous motion of multiple droplets. Beside the physical limitations to parallelism discussed with Fig. 12-2, the DMFS control hardware may impose additional constraints. For example, [44, 45] describe a DMFS with simpler row/column addressing, where a droplet moves to a neighboring cell $A(x,y)$ only if it lies at the intersection of activated column x and row y . Such conditions are encoded in line 5 of Algorithm 1:

“ $Q \leftarrow$ list of all valid motion transitions from o ; ”

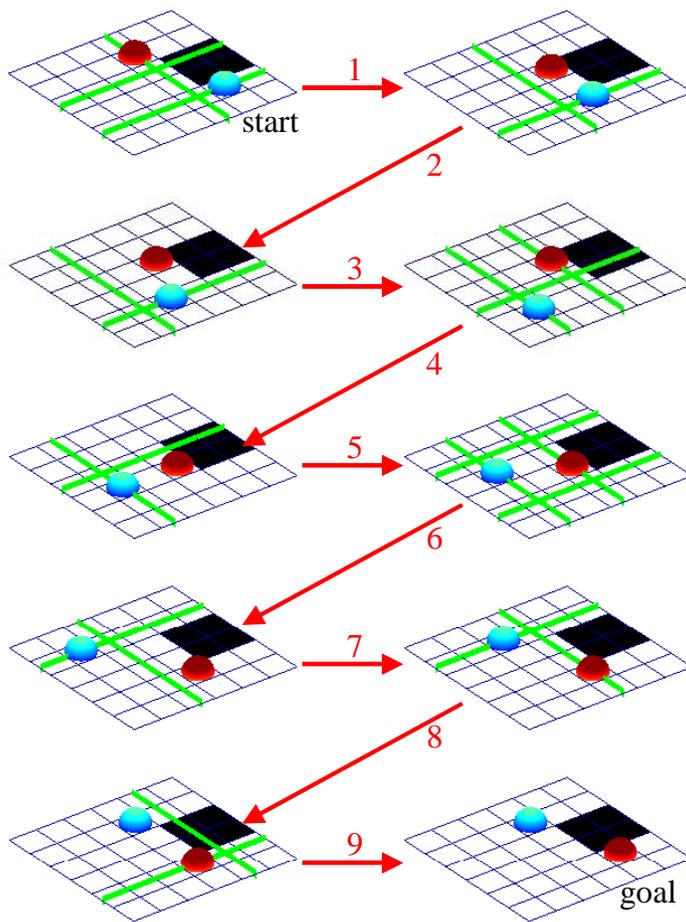


Figure 12-4. Two droplets trading places as in Fig. 12-3, but here droplets move only to neighbor cells whose row and column has been activated (indicated by a green line). An optimal strategy now requires 9 steps. Note that even though parallel droplet motion occurs in several steps, transitions 1 and 4 in Fig. 12-3 would not be possible with this addressing scheme

Generation of this list of transitions must be implemented depending on the hardware specifications. Fig. 12-4 shows an optimal solution for the same start and goal states as in Fig. 12-3 but with this more limited row/column addressing scheme.

4.4 Duplicate Droplet Types

An important special case occurs when multiple droplets in the DMFS have the same droplet type. This is a likely scenario in practice, especially in DMFS with large numbers of droplets. In this case, there is no unique mapping between droplets in A_s and A_g (or with any intermediate state A_i). This complicates the calculation of the cost estimate h , but can also provide for more efficient plans by choosing opportune droplets that are closest to their respective goals.

Suppose we are given two sets of d droplet placements, S_1 and $S_2 \in \mathcal{C}^d$, and all droplets have the same type T . We can find the minimum cost match between S_1 and S_2 efficiently (in analogy to bipartite graph matching [48]) by a greedy algorithm that sequentially matches up coordinate pairs with minimal Manhattan distance until all coordinates are paired up. This pairing leads to a monotone underestimate of the actual cost, and can thus be used as an admissible estimate for h . With this addition, both the basic and the prioritized A* algorithm for droplet path planning can efficiently handle inputs with duplicate droplet types.

5. DMFS TASK PLANNING

The previous section addressed the *DMFS motion planning problem*. However, transitions of the droplet types (due to mixing or other processing as discussed in Section 3) are essential parts of DMFS operation. Thus, this section extends the previously introduced algorithms to the *general DMFS planning problem*, which allows all remaining droplet transitions listed in Section 3.2, including merging, splitting, and changing of droplet type. This ultimately leads to the much broader question of how to transform a general laboratory protocol into a specific sequence of commands that can be executed on a DMFS.

5.1 Basic Graph Search

A straightforward algorithm to solve the general DMFS planning problem can be derived from Algorithm 1, where we can again modify line 5 to allow the complete set of transitions listed in Section 3.2, including in particular also changes of droplet type. However, this causes some immediate problems: (a) the number of possible transitions from each state (i.e., the branching factor in the search graph) becomes very large; (b) it is difficult to find an admissible heuristic for the A* algorithm, causing it to degenerate into breadth-first-search; in combination, this would result in

very inefficient searches. These problems would apply equally to a modified Algorithm 2.

5.2 DMFS Task Protocols

To develop a more useful algorithm, it is important to keep in mind that the tasks to be executed here typically are laboratory protocols. Thus, it is reasonable to assume that the user (e.g., a chemical engineer or a researcher in molecular biology) has carefully worked out the individual steps in this protocol, and identified the intermediate products that are being generated during its execution. With this additional input, we can find efficient algorithms to perform these tasks on a DMFS, while leaving the task design to a knowledgeable human operator.

We now introduce a very simple *DMFS task language*; the user of a DMFS specifies the tasks to be executed in this language, based on the laboratory protocol for the process of interest. Our Algorithms 3 and 4 then interpret this task description and translate it into actual DMFS commands.

DMFS Task Language

// Textual description of DMFS protocol
// $x \in \{1 \dots m\}; y \in \{1 \dots n\}; \Delta x, \Delta y \in \{0, 1, \dots\}; t \in \{1, 2, \dots\}$
// $T \in \mathcal{P}(\mathcal{S})$; $f: \mathcal{P}(\mathcal{S}) \rightarrow \mathcal{P}(\mathcal{S})$
// *id* is an arbitrary textual identifier for a cell

in $x y T id$ [time t]
out $x y T id$ [time t]
waste $x y id$
mergesplit $x y \Delta x \Delta y T id$ [time t]
check $x y T id$ [time t]
change $x y f id$ [time t]
block $x y T id$
connect *from-id to-id*

The statements in this language correspond to the DMFS array transitions listed in Section 3.2 with the following additional explanations:

x and y are the cell coordinates in the array. In general, we assume that transitions happen on a *single* array cell, except merge/split operations, which may require larger cells (specified by Δx and Δy such that $x + \Delta x \leq m$ and $y + \Delta y \leq n$).

- A droplet of type T_d is allowed on a cell with specified type T only if $T_d \subseteq T$.

- **in**, **out**, **mergesplit**, **check**, and **change** have an optional argument **time** t with default value $t = 1$ that specifies the time required for the transition.
- “**waste** $x\ y\ id$ ” is a short form for “**out** $x\ y\ *id$ ” which implies that the droplet type does not matter because the droplet will be discarded.
- “**block** $x\ y\ T\ id$ ” prohibits any droplet of type $T_d \subseteq T$.
- “**connect** *from-id to-id*” implies a single droplet moving between the two specified cells.

Note that identifiers need not be unique. However, if multiple transitions have the same identifier, then they belong to the same *cell group* and must describe the same transition. For example, we can write “**in** 1 1 H₂O DI-input” and “**in** 3 1 H₂O DI-input” to specify two cells (1,1) and (3,1) that provide a supply of DI water. Thus if we write “**connect** DI-input mix” then our algorithm will choose one of the DI water inputs to route a droplet to the cell with identifier “mix”.

The following Fig. 12-5 gives a sample DMFS task input. Four input droplets of three different initial droplet types go through a sequence of merges, splits, type transitions, and checks, before finally reaching an output or waste cell. The user specifies these steps and their locations on the DMFS array. Our algorithms automatically generate the order of these operations, the selection of specific cells from cell groups, and the exact droplet paths and schedule.

```

in 0 0 {R} red
in 0 2 {G} green
in 0 4 {B} blue
in 0 6 {B} blue
mergesplit 4 2 0 0 magenta
  connect red magenta
  connect blue magenta
mergesplit 4 5 0 0 cyan
  connect green cyan
  connect blue cyan
mergesplit 4 8 0 0 pale
  connect magenta pale
  connect cyan pale
change 14 1 {R,B}→{M} modify
  connect magenta modify
change 14 3 {G,B}→{C} change
  connect cyan change
change 14 5 {R,G,B}→{W} process
  connect pale process
check 10 2 all sensor

```

```

check 10 5 all sensor
    connect modify sensor
    connect change sensor
out 20 0 all out
out 20 2 all out
out 20 4 all out
    connect process out
    connect pale out
waste 20 10 trash
    connect sensor trash
    connect sensor trash

```

Figure 12-5. Sample DMFS tasks. There are three cell groups: *blue*, *sensor*, and *out*, consisting of multiple cells with the same identifier and the same transitions (**in**, **check**, **out**, respectively). The keyword **all** indicates the entire set of droplet types \otimes . Note that only two out of the three **out** cells will be used.

In large DMFS with many moving droplets and many **in**, **out**, **mergesplit**, **check**, and **change** cells, choosing the locations where droplets are processed should also be automated. A greedy algorithm and simulated annealing are discussed in [49] to attack this NP-hard layout problem.

While this list of statements may look tedious, it is simply a textual description of a graph in which every node represents a transition (**in**, **out**, **waste**, **mergesplit**, **check**, **change**) at a specific location on the DMFS, and every edge corresponds to a droplet motion (**connect**). We call this directed graph, which specifies the flow of the droplets through the DMFS, the *task graph*. It gives a more intuitive representation of the DMFS task to be executed and will be discussed in the following section.

5.3 DMFS Planning Algorithm

The final part of this paper is dedicated towards translating a DMFS task description, given in the language from the previous subsection, into a sequence of commands that can be executed on the array. This algorithm will do the following: (1) Generate the task graph from the textual input. (2) Identify initial transitions (typically, **in** nodes) that do not have any incoming edges. (3) Assign levels to all nodes in the task graph according to their precedence relationships such that transitions on the same level can be executed in parallel.

Algorithm 3: Task Graph Generation

Input: DMFS task description *tasks*

Output: task graph *G* with level assignments

```

parse tasks and generate the corresponding task graph G;
old  $\leftarrow \emptyset$ ;
new  $\leftarrow$  all nodes in G;
current  $\leftarrow$  all nodes in G that do not have predecessors;
i  $\leftarrow 0$ ;
while current  $\neq \emptyset$  begin
  mark all nodes in current with level i;
  add all nodes in current to old;
  current  $\leftarrow$  all nodes in new that have only predecessors in old;
  remove all nodes in current from new;
  i  $\leftarrow i + 1$ ;
end
if new =  $\emptyset$ 
then return G (with level numbers);           // success
else return  $\emptyset$ ;                           // failure

```

If the directed graph is acyclic then this algorithm finds a level assignment with a minimum number of levels (which we call l), thus maximizing the potential for parallel execution of the transitions represented by its nodes and edges. Note that these level assignments merely reflect precedence relationships, not actual execution times: droplet transitions on a specific level and droplet motions between levels may have varying transition times (and the latter are not yet known). Thus, faster droplets may have to wait until slower droplets are finished on each level.

Algorithm 3 assumes that there are no resource conflicts between droplets on any given level, i.e., no two transitions require the same cell on the DMFS array. If this cannot be guaranteed during the specification of the DMFS task, then the algorithm must be modified to assign conflicting transitions to different levels. See, e.g., [50] for a comprehensive approach to dealing with such resource constraints.

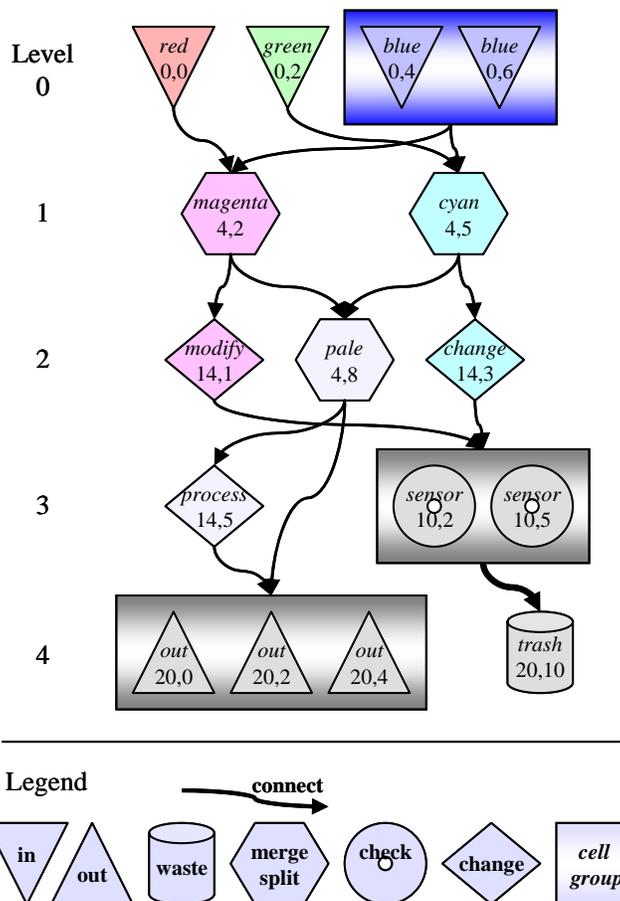


Figure 12-6. Task graph with level assignments generated from the task description in Fig. 12-5. Transitions on the same level can be executed in parallel. Note: (a) there are two droplets moving from the *sensor* cell group to *trash*, indicated by a double thickness arrow; (b) only two of the three output cells will be used.

Fig. 12-6 shows the task graph generated by Algorithm 3 from the DMFS command input given in Fig. 12-5. From its level assignment, we can immediately generate array states A_{i-} and A_{i+} that correspond to each level $i \in \{0, \dots, l\}$, such that A_{i-} and A_{i+} are the state of A immediately before and after the transitions of level i , respectively. Then, we can use Algorithm 2 to determine the droplet motions between arrays A_{i-1+} and A_{i-} for all $0 < i \leq l$:

Algorithm 4: DMFS Planning

Input: DMFS task description *tasks*

Output: task graph G and corresponding droplet motions S

```

G ← call Algorithm 3 with input tasks
if G = ∅ then return ∅;           // no task graph exists, failure
S ← ∅;
for i ← {1... l} begin           // l is the maximum level number of G
determine Ai-1+ and Ai-, using G and S;
Si ← call Algorithm 2 with start Ai-1+ and goal Ai-;
if Si = ∅
then return ∅;                   // no droplet path i exists, failure
else add Si to S;                // droplet path i found
end
return G and S;                  // success

```

Table 12-1. Start, intermediate, and goal states generated from the task graph in Fig. 12-6. For each state A_i , $i \in \{0, \dots, 4\}$, droplet placements and their respective types are shown before and after the transition on level i . Three or four droplets move simultaneously during the four transitions from A_{i-1+} to A_{i-} (for $i > 0$), indicated by downward arrows.

DMFS Task States and Transitions				
State	Droplet Placements			
A_0	(0,0)	(0,2)	(0,4)	(0,6)
+	{R}	{G}	{B}	{B}
	↓	↓	↓	↓
A_1	(4,2)	(4,5)	(4,2)	(4,5)
-	{R}	{G}	{B}	{B}
+	{R,B}	{G,B}	{R,B}	{G,B}
	↓	↓	↓	↓
A_2	(14,1)	(14,3)	(4,8)	(4,8)
-	{R,B}	{G,B}	{R,B}	{G,B}
+	{M}	{C}	{R,G,B}	{R,G,B}
	↓	↓	↓	•
A_3	(10,2)	(10,5)	(14,5)	(4,8)
-	{M}	{C}	{R,G,B}	{R,G,B}
+	{M}	{C}	{P}	{R,G,B}
	↓	↓	↓	↓

A_4	(20,10)	(20,10)	(20,4)	(20,2)
-	{M}	{C}	{P}	{R,G,B}

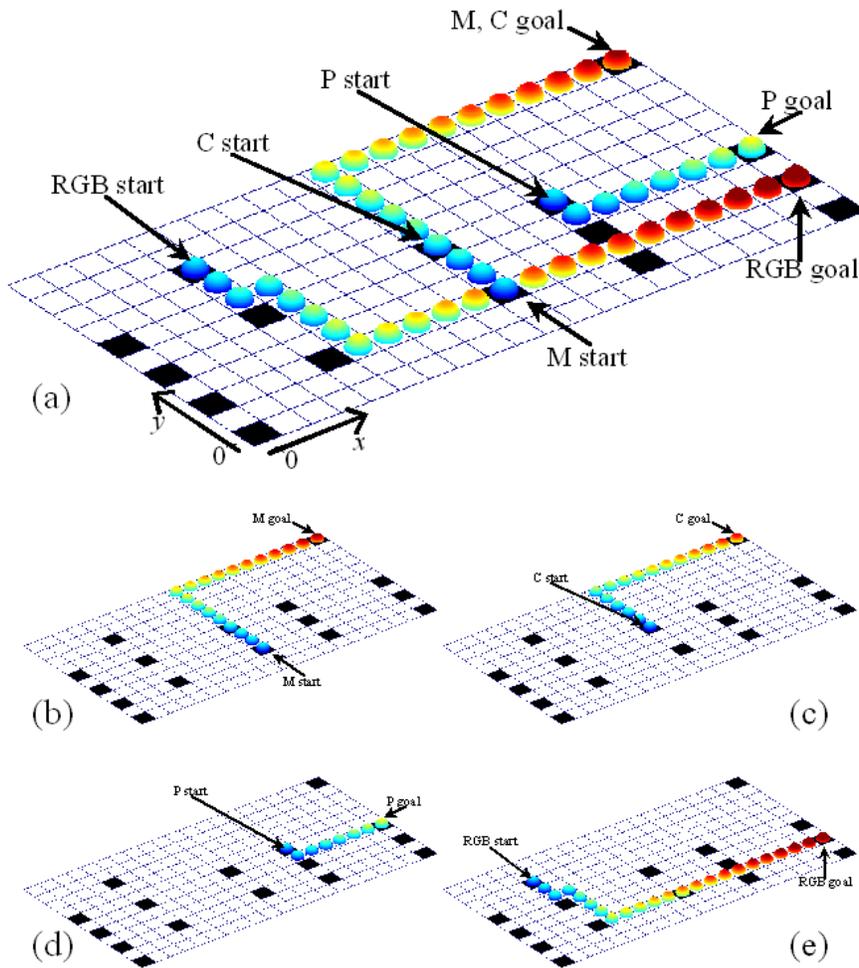


Figure 12-7. Simultaneous droplet motion during transition between states A_{3+} and A_{4-} . (a) shows all droplets, with change in color indicating progressing time. Cells with special functions are marked as black squares. (b), (c), (d), and (e) show individual droplet paths for the droplets of type {M}, {C}, {P}, and {R,G,B}, respectively. Note: In (b) and (c), droplet {C} follows the path of droplet {M} at a distance of 3 cells; in (e), the droplet circumnavigates the mergesplit cells at (4,2) and (4,5) but is allowed to pass over the sensor cell at (10,2).

Table 12-1 lists all the states and transitions generated by Algorithm 4 from the task graph in Fig. 12-6. Fig. 12-7 attempts to visualize parallel motion of multiple droplets on the DMFS for the transition from A_{3+} to A_{4-} .

Algorithm 3 is linear in the number of nodes and edges in the task graph. The complexity of Algorithm 4 is dominated by the calls to Algorithm 2, which occur $l - 1$ times total. These algorithms were implemented in Java. The total run times for the examples in this paper are in the millisecond range. The code is available upon request from the author.

6. CONCLUSION

This paper makes the following contributions: (1) A formal, hardware independent model of droplet-based microfluidic systems (DMFS). (2) Novel algorithms for motion and task planning with DMFS, leading to efficient (albeit not necessarily optimal or complete) solutions for coordinating large numbers of simultaneously moving droplets on a two-dimensional array. (3) An approach to automate the transition from general laboratory protocols to DMFS control command sequences. (4) Results using an implementation of these algorithms in Java.

The developed models and algorithms are “modular”, such that results from the different sections are largely independent; e.g., DMFS *task* planning in Section 5 does not rely on a particular droplet *path* planning algorithm so some other algorithm could be readily substituted for prioritized A*. Similarly, the path planning algorithms from Section 4 could be applied to a different task planning algorithm.

Droplet manipulation based on electrowetting on arrays with up to hundred cells has been demonstrated by several groups (e.g., [3, 44, 51]), and an electrophoresis-based system with integrated CMOS addressing of tens of thousands of cells by [34]. The computational complexity for generating optimal droplet motion plans has been shown to be prohibitive even for much smaller systems. Thus, we have focused on finding an acceptable trade-off between efficiency and optimality.

A very different approach to this problem could be to limit droplet manipulation to a few standard, “pre-packaged” strategies. For example, on a 100×100 array, about 50 droplets could move in parallel across the array, followed by another wave of 50 droplets, etc., resembling a repetitive “peristaltic” motion [43]. However, in this case, the fundamental advantage of flexibility and reprogrammability in DMFS versus conventional (channel, valve, and pump based) microfluidic architectures is lost. In addition, the question still remains how to initially generate the “pre-packaged” strategies

if they involve more complicated motion paths by many simultaneously moving droplets.

Other future work should explore the following directions:

1. Polynomial approximation algorithms exist for NP-hard problems (e.g., traveling salesman [52, 53]), which guarantee a tight limit on non-optimality. If, e.g., a control strategy for a complex DMFS can be generated in polynomial time that is guaranteed to be at most twice as long as an optimal solution then this might be sufficient for most practical purposes.
2. While the (prioritized) A* algorithm has been effective in solving graph search problems, it is incomplete and worst-case exponential in the branching factor. More detailed benchmark tests could provide insights about scenarios where the algorithm fails to find solutions efficiently.
3. The optimal level number l produced by Algorithm 3 does not automatically imply maximal parallelism in droplet motion. Some nodes in the task graph can be assigned to a range of levels without affecting l , but varying level assignments may produce droplet motion plans with varying efficiency. For example, the droplet motion from (4,8) to (20,2) in Table 12-1 and Fig. 12-7(e) can be executed during transition $A_{2+} \rightarrow A_{3-}$ or during $A_{3+} \rightarrow A_{4-}$.
4. A related question is whether it is essential to allow parallel droplet motion in line 5 of Algorithm 1. An alternative approach would first generate plans without parallelism, and then post-process the generated plan to identify all droplet motions that could be executed in parallel.
5. More generally, it may be possible to improve the output of Algorithm 2 with some post-processing that locally improves the droplet motions.
6. The previous three points hint that our DMFS formalism could be developed much further. A general approach in this direction based on *state complexes* was given recently in [54], which presents efficient algorithms to detect and optimize parallelism.
7. As mentioned in Section 3.1, parallelism may be limited by the hardware controller to a number smaller than the total droplet count. This was not explicitly addressed in this paper, but could again appear as an additional constraint in line 5 of Algorithm 1.

7. ACKNOWLEDGMENT

The author thanks Srinivas Akella, Sankar Basu, Bruce R. Donald, Mike Erdmann, Rajinder Khosla, Eric Klavins, Xiaorong Xiong, and the anonymous reviewers for helpful insights and comments, Ji Hao Hoo and

Tsung-Hao Suh for programming, Rohit Malhotra also for programming of an earlier software version, and Masayoshi Esashi, Hiroyuki Fujita and Osamu Tabata for their hospitality during a sabbatical visit at their laboratories.

Support for this project was provided in part by NSF SGER grant 0342632, NIH grant 1 P50 HG002360-01, and an invitational fellowship for research in Japan from the Japan Society for the Promotion of Science.

8. REFERENCES

1. Kovacs, G.T.A., *Micromachined Transducers Sourcebook*. 1998: McGraw-Hill.
2. Stone, H.A., A.D. Stroock, and A. Ajdari, *Engineering Flows in Small Devices: Microfluidics Toward a Lab-on-a-Chip*. Annual Review of Fluid Mechanics, 2004. **36**:381-411.
3. Moon, H., S.K. Cho, R.L. Garrell, and C.-J. Kim, *Low voltage electrowetting-on-dielectric*. Journal of Applied Physics, 2002. **92**(7):4080-4087.
4. Fair, R.B., V. Srinivasan, H. Ren, P. Paik, V.K. Pamula, and M.G. Pollack. *Electrowetting-based on-chip sample processing for integrated microfluidics*, in *IEEE International Electron Devices Meeting (IEDM)*. 2003.
5. Zhang, T., K. Chakrabarty, and R.B. Fair, *Integrated hierarchical design of microelectrofluidic systems using SystemC*. Microelectronics Journal, 2002. **33**:459-470.
6. Zhang, T., K. Chakrabarty, and R.B. Fair, *Design of Reconfigurable Composite Microsystems Based on Hardware/Software Codesign Principles*. IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems, 2002. **21**(8):987-995.
7. *International Conference on Miniaturized Chemical and Biochemical Analysis Systems (microTAS)*. Annual.
8. *Sensors and Actuators B: Chemical*. Monthly, Elsevier.
9. *Lab on a Chip*. Monthly, Royal Society of Chemistry.
10. Shapiro, H.M., *Practical flow cytometry*. 1995, New York: Wiley.
11. Melamed, M.R., T. Lindmo, and M.L. Mendelsohn, *Flow cytometry and sorting*. 1990, New York: Wiley.
12. Crosland-Taylor, P.J., *A device for counting small particles suspended in a fluid through a tube*. Nature, 1953. **171**(4340):37-38.
13. Fu, A.Y., C. Spence, A. Scherer, F.H. Arnold, and S.R. Quake, *A microfabricated fluorescence-activated cell sorter*. Nature Biotechnology, 1999. **17**(11):1109-1111.

14. Krueger, J., K. Singh, A. O'Neill, C. Jackson, A. Morrison, and P. O'Brien, *Development of a microfluidic device for fluorescence activated cell sorting*. Journal of Micromechanics and Microengineering, 2002. **12**:486-494.
15. Tartagni, M., L. Altomare, R. Guerrieri, A. Fuchs, N. Manaresi, G. Medoro, and R. Thewes, *Microelectronic Chips for Molecular and Cell Biology*, in *Sensors Update*, H. Baltes, G.K. Fedder, and J.G. Korvink, Editors. 2004, Wiley-VCH. p. 156-200.
16. Beni, G. and M.A. Tenan, *Dynamics of electrowetting displays*. Applied Physics, 1981. **52**(10):6011-6015.
17. Pollack, M.G., R.B. Fair, and A.D. Shenderov, *Electrowetting-based actuation of liquid droplets for microfluidic applications*. Applied Physics Letters, 2000. **77**(11):1725-1726.
18. Jones, T.B., M. Gunji, M. Washizu, and M.J. Feldman, *Dielectrophoretic liquid actuation and nanodroplet formation*. Journal of Applied Physics, 2001. **89**(2):1441-1448.
19. Nanolytics, www.nanolytics.com.
20. Wixforth, A., *Verfahren und Vorrichtung zur Manipulation kleiner Flüssigkeitsmengen auf Oberflächen*, German Trademark and Patent Office. 2002, Advalytix AG, 85649 Brunnthall, DE: Germany.
21. Wixforth, A. and C. Gauer, *Mischvorrichtung und Mischverfahren für die Durchmischung kleiner Flüssigkeitsmengen*, in *European Patent Office*. 2004: European Union.
22. Wixforth, A., A. Rathgeber, C. Gauer, and J. Scriba, *Vorrichtung und Verfahren zur Vermessung kleiner Flüssigkeitsmengen und/oder deren Bewegung*, German Trademark and Patent Office. 2002, Advalytix AG, 80799 München, DE: Germany.
23. Kataoka, D.E. and S.M. Troian, *Patterning Liquid Flow at the Microscopic Scale*. Nature, 1999. **402**(6763):794-797.
24. Darhuber, A.A., J.P. Valentino, J.M. Davis, S.M. Troian, and S. Wagner, *Microfluidic actuation by modulation of surface stresses*. Applied Physics Letters, 2003. **82**(4):657-659.
25. Gallardo, B.S., V.K. Gupta, F.D. Eagerton, L.I. Jong, V.S. Craig, R.R. Shah, and N.L. Abbott, *Electrochemical principles for active control of liquids on submillimeter scales*. Science, 1999. **283**(5398):57-60.
26. Lahann, J., S. Mitragotri, T.-N. Tran, H. Kaido, J. Sundaram, I.S. Choi, S. Hoffer, G.A. So-morjai, and R. Langer, *A Reversibly Switching Surface*. Science, 2003. **299**(5605):371-374.
27. Chaudhury, M.K. and G.M. Whitesides, *How to make water run uphill?* Science, 1992. **256**(5063):1539-1541.

28. Daniel, S., S. Sircar, J. Gliem, and M.K. Chaudhury, *Ratcheting Motion of Liquid Drops on Gradient Surfaces*. Langmuir, 2004. **20**(10):4085-4092.
29. Sandre, O., L. Gorre-Talini, A. Adjari, J. Prost, and P. Silberzan, *Moving droplets on asymmetrically structured surfaces*. Physical Review E, 1999. **60**(3):2964-2972.
30. Shastry, A., M. Case, and K.F. Böhringer. *Engineering Surface Texture to Manipulate Droplets in Microfluidic Systems*, in *IEEE Conference on Micro Electro Mechanical Systems (MEMS)*. 2005. Miami Beach, FL.
31. Jones, T.B., J.D. Fowler, Y.S. Chang, and C.-J. Kim, *Frequency-Based Relationship of Electrowetting and Dielectrophoretic Liquid Microactuation*. Langmuir, 2003. **19**(18):7646-7651.
32. Zheng, J. and T. Korsmeyer, *Principles of droplet electrohydrodynamics for lab-on-a-chip*. Lab on a Chip, 2004. **4**:265-277.
33. Gascoyne, P.R.C., www.dielectrophoresis.org.
34. Fuchs, A., N. Manaresi, D. Freida, L. Altomare, C.L. Villiers, G. Medoro, A. Romani, I. Chartier, C. Bory, M. Tartagni, P.N. Marche, F. Chatelain, and R. Guerrie. *A Microelectronic Chip Opens New Fields in Rare Cell Population Analysis and Individual Cell Biology*, in *Micro Total Analysis Systems (MicroTAS)*. 2003. Squaw Valley, CA.
35. Paik, P., V.K. Pamula, and R.B. Fair, *Rapid droplet mixers for digital microfluidic systems*. Lab on a Chip, 2003. **4**:253-259.
36. Cho, S.K., H. Moon, and C.-J. Kim, *Creating, transporting, cutting, and merging liquid droplets by electrowetting-based actuation for digital microfluidic circuits*. Journal of Microelectromechanical Systems, 2003. **12**(1):70-80.
37. Griffith, E. and S. Akella. *Coordinating multiple droplets in planar array digital microfluidics systems*, in *Sixth Workshop on the Algorithmic Foundations of Robotics*. 2004. Utrecht, Zeist, The Netherlands.
38. Peng, J. and S. Akella. *Coordinating Multiple Robots with Kinodynamic Constraints along Specified Paths*, in *Workshop on the Algorithmic Foundations of Robotics (WAFR)*. 2002.
39. Akella, S. and S. Hutchinson. *Coordinating the Motions of Multiple Robots with Specified Trajectories*, in *IEEE International Conference on Robotics and Automation*. 2002. Washington D.C.
40. Ding, J., K. Chakrabarty, and R.B. Fair, *Scheduling of Microfluidic Operations for Reconfigurable Two-Dimensional Electrowetting Arrays*. IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems, 2001. **20**(12):1463-1468.
41. Erdmann, M. and T. Lozano-Pérez, *On Multiple Moving Objects*. Algorithmica, 1987. **2**(4):477-521.

42. Böhringer, K.F. *Optimal Strategies for Moving Droplets in Digital Microfluidic Systems*, in *Seventh International Conference on Miniaturized Chemical and Biochemical Analysis Systems (MicroTAS'03)*. 2003. Squaw Valley, CA.
43. Böhringer, K.F. *Towards Optimal Strategies for Moving Droplets in Digital Microfluidic Systems*, in *IEEE International Conference on Robotics and Automation (ICRA)*. 2004. New Orleans, LA.
44. Fan, S.-K., P.-P.d. Guzman, and C.-J. Kim. *EWOD Driving of Droplet on NxM Grid Using Single Layer Electrode Patterns*, in *Solid-State Sensor, Actuator, and Microsystems Workshop*. 2002. Hilton Head Island, SC.
45. Fan, S.-K., C. Hashi, and C.-J. Kim. *Manipulation of multiple droplets on NxM grid by cross-reference EWOD driving scheme and pressure contact packaging*, in *IEEE International Conference on Microelectromechanical Systems*. 2003. Kyoto, Japan.
46. Lozano-Pérez, T., *Spatial planning: A configuration space approach*. IEEE Transactions on Computers, 1983. **C-32**(2):108-120.
47. Nilsson, N.J., *Principles of Artificial Intelligence*. 1982, Berlin Heidelberg New York: Springer Verlag.
48. Aho, A.V., J.E. Hopcroft, and J.D. Ullman, *Data Structures and Algorithms*. 2 ed. 1987, Reading, WA: Addison-Wesley.
49. Su, F. and K. Chakrabarty. *Design of fault-tolerant and dynamically-reconfigurable microfluidic biochips*, in *Design, Automation and Test in Europe (DATE)*. 2005.
50. Su, F. and K. Chakrabarty. *Architectural-level synthesis of digital microfluidics-based biochips*, in *IEEE International Conference on Computer Aided Design*. 2004.
51. Srinivasan, V., V.K. Pamula, M.G. Pollack, and R.B. Fair. *Clinical Diagnostics on Human Whole Blood, Plasma, Serum, Urin, Saliva, Sweat, and Tears on a Digital Microfluidic Platform*, in *Micro Total Analysis Systems (MicroTAS)*. 2003. Squaw Valley, CA.
52. Christofides, N. *Worst-case analysis of a new heuristic for the traveling salesman problem*, in *Symposium on New Directions and Recent Results in Algorithms and Complexity*. 1976. Orlando, FL: Academic Press.
53. Arora, S., *Polynomial Time Approximation Schemes for Euclidean Traveling Salesman and Other Geometric Problems*. Journal of the ACM, 1998. **45**(5):753-782.
54. Abrams, A. and R. Ghrist, *State Complexes for Metamorphic Robots*. The International Journal of Robotics Research, 2004. **23**(7-8):811-826.