Exact Routing for Digital Microfluidic Biochips with Temporary Blockages

Oliver Keszocze Robert Wille Rolf Drechsler

Institute of Computer Science, University of Bremen, Bremen, Germany
Cyber-Physical Systems, DFKI GmbH, Bremen, Germany
{keszocze,rwille,drechsle}@informatik.uni-bremen.de

***Abstract*—Digital microfluidic biochips enable a higher degree of automation in laboratory procedures in biochemistry and molecular biology and have received significant attention in the recent past. Their design is usually conducted in several stages with routing being a particularly critical challenge. Previously proposed solutions for this design step suffer from two issues: They are mainly of heuristic nature and usually assume that the blockages to be bypassed are present the entire time. In contrast, we present a methodology which exploits the fact that blockages are often only present at certain intervals. At the same time, our approach guarantees exact solutions, i.e. always determines a routing with a minimal number of time steps. Experimental results show that, despite the huge complexity, optimal results can be achieved in reasonable run-time and that the consideration of temporary blockages indeed significantly improves the routing results.**

I. Introduction

Digital Microfluidic Biochips (DMFBs) are an emerging technology which attracted significant attention in the recent past. They enable an automation of laboratory procedures in biochemistry and molecular biology by providing a platform in which samples and corresponding operations can be controlled automatically. For this purpose, so called *droplets*, i.e. miniaturized and discretized liquids which serve as sample carriers, are applied onto a two-dimensional electrode grid. By assigning time-varying voltages to turn electrodes on and off, droplets can be moved around the entire grid. Peripheral devices such as dispensing ports, heating devices, and optical detectors allow to place samples onto the grid and to perform fundamental operations for the conducted experiments [1].

By this, DMFBs offer a flexible control mechanism, a high throughput and sensitivity, as well as a low sample/reagent volume consumption. This advances the execution of numerous biochemical assays [2] and eventually found significant interest in healthcare, environmental, and point-of-care-testing applications. According to a report released by Research and Markets in June 2013, the global biochip market will grow from 1.4 billion in 2013 to 5.7 billion by 2018 [3].

Motivated by these promising developments, researchers and engineers started to aid the design of biochips by means of automatic methods. They roughly followed the established (conventional) design flow composed of allocation, binding, scheduling, placement, and finally routing – although for each step, of course, dedicated solutions have been developed (see e.g. [4] for a tool implementing a collection of these solutions). Within this flow, the problem of routing poses a particularly critical challenge.

Here, a fixed grid with a precise placement of all operations (such as mixing, detection, heating, etc.) together with the initial and desired positions (denoted as *source positions* and *target positions*) of all considered droplets is given. Then, the task of routing is to determine a route from the source position to the target position for each droplet, such that unintended mixing of droplets is avoided and blockages (e.g. caused by operations in progress) are bypassed. The number of required time steps to accomplish a routing serves as the optimization objective.

Although routing in DMFBs is similar to conventional routing for VLSI systems, it offers some intrinsic properties but also requires to explicitly consider additional constraints. Accordingly, several approaches for routing in DMFBs have been presented in the recent past (see e.g. [5], [6], [7], [8], [9]; they are discussed in more detail later in Section II-C). However, these approaches particularly suffer from two issues:

1) They are mainly of heuristical nature, meaning that no optimal results are known thus far, and
2) they ignore the fact that blockages are often not present permanently but only at certain intervals, i.e. *temporary blockages* are not considered.

In this work, we propose a routing methodology which addresses these problems and copes with the exponentially hard complexity of determining a minimal solution by using solving engines for *SAT Modulo Theories* (SMT). They allow for solving instances composed of hundreds of thousands of variables and constraints and, hence, provide a promising core technology for the considered problem. In order to apply these solvers, the considered optimization problem (determine a routing with the minimal number of time steps) is formulated as a sequence of decision problems. Each decision problem is then formulated as an SMT instance and solved by a corresponding solving engine. While this scheme guarantees minimality of the result, the respective formulation additionally allows for an easy consideration of temporary blockages.

Experimental results confirm the improvements obtained by this methodology. Using SMT, minimal results can be obtained in reasonable run-time, despite the huge complexity. For the first time, this enabled a comparison of previously obtained results to the actual optimal solution. Moreover, the explicit consideration of temporary blockages allow for further reductions in the number of time steps needed to realize a routing.

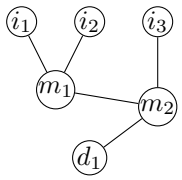
The remainder of this paper is structured as follows. Section II provides the background on DMFBs, the routing problem, as well as related work. Open potential for possible improvements of previously introduced routing approaches is discussed in Section III before the solution proposed in this work is described in Section IV. Finally, results of our experimental evaluation are summarized in Section V and the paper is concluded in Section VI.

II. Background

To keep this work self-contained, this section reviews the basics of DMFB design with a particular focus on the considered routing problem and briefly discusses previously proposed solutions.

A. Design of Digital Microfluidic Biochips

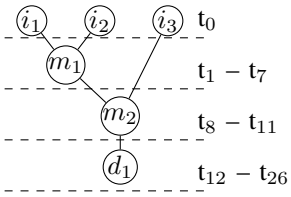
Usually, the design of DMFBs is conducted over several stages, including well known steps such as allocation, binding, scheduling, placement, and routing. The starting point for each design is a *sequencing graph* which specifies the desired functionality to be implemented. A *module library* provides realizations such as mixing, detecting, etc. as well as their respectively needed grid-size and timing requirements. The design objective is to realize the desired functionality onto a given grid within the smallest possible completion time utilizing the available operations from the module library.



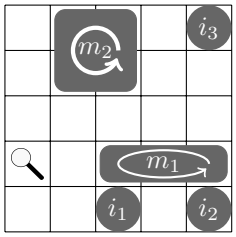
(a) Sequencing Graph

Op	Res	Dur
$i_{1,2,3}$	On-chip	1
m_1	1×3	7
m_2	2×2	4
d_1	LED	15

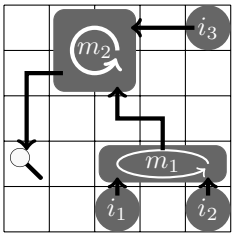
(b) Binding



(c) Scheduling



(d) Placement



(e) Routing

Fig. 1: Design of DMFBs

The respective design steps are briefly illustrated in Fig. 1. More precisely, the process of *allocation and binding* defines which biochemical operation specified in the sequencing graph is realized by which module from the module library. *Scheduling* determines the order of operations based on the binding result. Afterwards, the resulting configuration is realized on the actual chip. This includes the *placement*, i.e. the determination of the actual locations of the different operations – possibly enriched by information in which time steps the respective operation is executed. Based on that, *routing* schedules the movement of each droplet.

B. Routing in Digital Microfluidic Biochips

In this work, we focus on the routing problem within the flow described above. The routing problem is defined as follows:

Definition 1. *Given a fixed grid with a placement of all operations as well as the source positions and the target positions of all considered droplets, the process of* routing *is to determine a route from the source position to the target position for each droplet. The number of required time steps to accomplish a routing is applied as the optimization objective.*

In order to determine the routes, the following *constraints* have to be considered:

- Two droplets must not occupy the same cell on the grid. Moreover, in order to avoid unintended mixing of the fluids, two droplets which are not supposed to be mixed must not occupy adjacent cells [5].
- Operations are considered as *blockages*, i.e. the movement of droplets must not interfere with the position of any operation.

Example 1. *Consider the sequence graph as shown in Fig. 1(a) and its resulting placement in Fig. 1(d). A possible routing is depicted in Fig. 1(e).*

Note that routing is often split into several sub-problems. That is, a single routing problem does not necessarily consider the entire functionality to be realized at once. In the example from Fig. 1, routing the droplets i_1 and i_2 to the position of mixer m_1, the resulting droplet and droplet i_3 to the position of mixer m_2, and the resulting droplet to the position of the detector d_1 may be considered as three separate routing problems. The approach proposed in this work is capable of solving both problems, single sub-problems but also the whole routing task at once.

In order to formally define and address this problem, the following notation is used.

Definition 2. *Consider a* $(w \times h)$-grid*. Then,*

- $C = \{1, 2, \ldots, w\} \times \{1, 2, \ldots, h\}$ *denotes the set of all* positions *or* cells *of the grid,*
- $D = \{i_1, i_2, i_3, \ldots, i_k\}$ *denotes the set of* droplets *considered in the given routing problem,*
- (x_i^*, y_i^*) *denotes the* source position *of the droplet* $i \in D$*,*
- $(x_i^\dagger, y_i^\dagger)$ *denotes the* target position *of the droplet* $i \in D$*,*
- T_i^* *denotes the* spawn time *of the droplet* $i \in D$*, i.e. the time step in which* i *spawns at its source position,*
- $T_i^\dagger$ *denotes the best possible* target time *for the droplet* $i \in D$*, i.e. the time step in which* i *would arrive its target position when a direct path (and no blockages) is assumed,*
- $T^\dagger$ *denotes the best possible overall completion time, i.e.* $T^\dagger := \max_{i \in D}(T_i^\dagger)$*,*
- B *denotes the set of all* blockages*. Each element* $(x, y, t_1, t_2) \in B$ *provides the* (x, y)*-position of the blockage and the interval* $[t_1, t_2]$ *in which the blockage is present.*

Using this notation, the routing problem from above can be defined as the following task: For each droplet $i \in D$, determine a path from (x_i^*, y_i^*) at time step T_i^* to $(x_i^\dagger, y_i^\dagger)$ that does not violate the constraints mentioned above. In the literature, the respective routing tasks are defined in terms of so-called *nets*. One can distinguish between two-pin nets (one droplet has to be routed to a given target) and three-pin nets (two droplet share the same target).

C. Related Work

In principle, routing in DMFBs is similar to conventional routing for VLSI systems. However, significant differences exists since e.g. electrical circuits must not be short-circuited while paths of droplets may cross each other. Consequently, dedicated approaches for routing DMFBs have been introduced in the past, which exploit the intrinsic properties but also explicitly consider the constraints in this domain.

In [5], different routing paths for each net are computed in a greedy fashion. Afterwards, for each net, one of these paths is chosen. When every net was assigned a path it is checked whether any constraint is violated. If necessary, paths are rechosen until a solution is found. The concept of *network-flows* is used in [6] to solve the routing problem via a two-stage algorithm. In [7], routing prioritized by a metric called *bypassibility of droplets* followed by a greedy compaction step is proposed. The approach assumes that a higher bypassibility will lead to less blockage of other droplets. Accordingly, highly prioritized droplets are routed first. The work [8] uses the concept of entropy, borrowed from thermodynamics, to determine a routing for droplets. This approach first produces sequential routing paths which, afterwards, are compacted using dynamic programming.

However, all these approaches employ a heuristical scheme. This means that they do not guarantee optimality with respect to the number of required time steps. Thus far, determining minimal results for relevant benchmarks suffered from the respective complexity. Indeed an exact approach (exploiting *Integer Linear Programming*, ILP) has been proposed in [9], but failed to generate exact results within the considered time limit. Hence, the ILP formulation has only been applied in a progressive fashion which approximates the minimum but cannot guarantee it. Afterwards, it was tried to address this problem using Boolean satisfiability [10]. However, this work

also employs a two-stage algorithm that does not guarantee optimality. Furthermore, these approaches address routing for cross-referenced instead of direct-addressed biochips considered in this work.

Besides that, all approaches mentioned above considered the blockages to be present on the grid during the entire time. But this is usually not the case. For example, the mixer of operation m_1 from Fig. 1 blocks the respective 2×2-grid only within the interval $[t_1, t_7]$; in the remaining time steps, a routing through the respective cells can be conducted. To the best of our knowledge, this has not been exploited for routing of droplets yet. A similar issue has been discussed in [11], but here re-routing of droplets in case of physical failures is considered; not the routing problem as such.

In [12], exact results are obtained for the one-pass synthesis of DMFBs, which includes routing. However, this work addresses a different and non-comparable problem. Also, the movement of droplets is modeled differently.

Overall, existing approaches have some drawbacks, which affect the quality of the obtained results and leave potential for improvements. This potential is discussed in the next section, which provides the motivation of this work.

III. Exact Routing with Temporal Blockages

Previously proposed approaches for DMFB routing as reviewed above particularly suffer from two issues:

1) They are mainly of heuristic nature, meaning that no optimal results are known thus far, and
2) they ignore the fact that blockages are often not present permanently but only in certain intervals of time.

The first issue is crucial as it does not allow an evaluation of the quality of existing (heuristically obtained) solutions. Thus far, existing routing approaches have always been compared against each other, but, due to the lack of exact results, not against the optimum. Although an exact approach has been discussed in [9], its complexity prevented a successful termination of the algorithm for the considered benchmarks within the given time limit, i.e. no exact solutions have been determined. This negatively affects the evaluation. For example, improving a heuristical result by 10% is significant, if this leads to an optimal solution, but marginal if the generated results are still orders of magnitude away from the optimum. To obtain such conclusions, exact approaches need to be available.

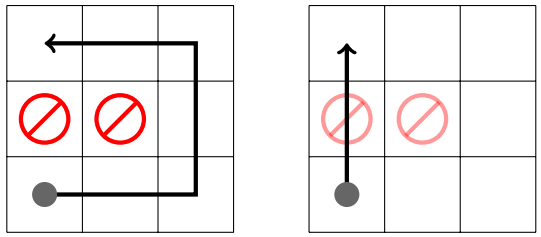
The second issue, i.e. ignoring the fact that blockages are often only present at certain intervals, prevents the determination of a better routing. This is illustrated by the following (trivial) example.

Example 2. *Consider the routing problem depicted in Fig. 2. The droplet shall be routed from position* $(0,0)$ *to position* $(0,2)$ *while blockages at cells* $(0,1)$ *and* $(1,1)$ *that are present at the time interval* $[1,2]$*, have to be considered. Previously proposed approaches do not consider the time interval and, hence, always bypass the blockages. This leads to a solution shown in Fig. 2(a) requiring six time steps. In contrast, considering this interval, would enable a solution in which the droplet remains on its source position for one time step (until the blockage disappears) and, then, approaches the target position in a straight-forward fashion. This leads to a solution shown in Fig. 2(b) requiring three time steps only.*

Hence, addressing these drawbacks would lead to significant improvements in the way routing of DMFBs is conducted today. In the following, we introduce an alternative routing approach for microfluidic biochips that explicitly solves these shortcomings.

IV. Proposed Solution

This section presents a methodology which exploits the fact that blockages are often only present in certain intervals, and, at the same time, generates an exact solution with respect to the number of time steps.



(a) W/o temporary blockages (b) W/ temporary blockages

Fig. 2: Considering temporary blockages

A. General Idea

Determining an exact routing for a DMFB obviously is an optimization problem of significant (computational) complexity. In order to cope with this seriously large search space to be explored, we propose to address this problem by exploiting the deductive power of solvers for *Satisfiability Modulo Theories* (SMT). The SMT problem is defined as follows:

Definition 3. *Let* Φ *be a logical formula in first-order logic which may additionally be restricted by further functions or predicates. The* SMT problem *is to determine an assignment to the variables of* Φ *such that* Φ *evaluates to true or to prove that no such assignment exists. In the former case, the corresponding SMT instance* Φ *is called* satisfiable*; otherwise,* Φ *is called* unsatisfiable*. In this work, we assume* Φ *being composed of Boolean variables whose assignments can additionally be treated as natural numbers* 0 *and* 1 *in order to allow a notion of cardinality. The formula itself is restricted by logical operations as well as cardinality constraints.*

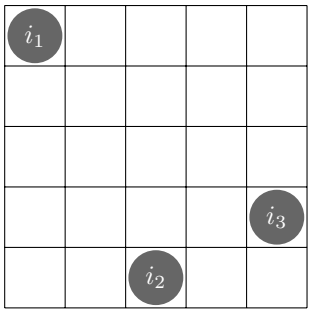
Example 3. *Let* $\Phi = x_1 \Leftrightarrow (x_2 + x_3 + x_4 = 2)$*. Then,* $x_1 = 1, x_2 = 0, x_3 = 1$*, and* $x_4 = 1$ *is a satisfying assignment for* Φ*.*

Today there exist SMT algorithms which solve many practical problem instances, i.e. instances composed of hundreds of thousands of variables and constraints, in reasonable time. Hence, SMT solving has become the state-of-the-art for many design problems. Motivated by these performances, the corresponding solving engines are applied in order to solve the routing problem addressed here.

However, SMT solvers tackle decision problems, while determining an exact routing, i.e. a minimal route for all droplets, obviously is an optimization problem. Hence, we propose an iterative scheme: the routing problem is approached as a sequence of decision problems. More precisely, the following flow is applied: First a decision problem is formulated checking whether the desired routing can be accomplished within $T = T^\dagger$ time steps[1]. If this is the case, the determined routing is already optimal. Otherwise, T is incremented until one of the following decision problems is satisfiable. In this case, the desired routing has been determined using T time steps. Since T is iteratively increased, starting from the lower bound $T^\dagger$, minimality is ensured.

For this purpose, the respective decision problems "Is the desired routing accomplishable in T time steps?" have to be formulated as SMT instances. The resulting formulation must be satisfiable (unsatisfiable), if such a routing does exist (does not exist). In the satisfiable case, the determined assignment must furthermore allow to extract the precise routing. An SMT formulation which incorporates these properties is described next.

[1]Recall that $T^\dagger$ denotes the best possible overall completion time as defined in Def. 2 and, hence, is applied as lower bound for the considered problem.



(a) Placement of droplets

Droplet 1: $d^t_{1,5,1} = 1$
Droplet 2: $d^t_{3,1,2} = 1$
Droplet 3: $d^t_{5,2,3} = 1$

Other positions:
Droplet 1
$d^t_{1,1,1} = 0$
$d^t_{1,2,1} = 0$
$\vdots$
$d^t_{5,5,1} = 0$

Droplet 2
$d^t_{1,1,2} = 0 \ldots$

(b) Corresponding assignment

Fig. 3: Representation of a time step t

B. Formulation of the Routing Problem as SMT Instance

To formulate the corresponding SMT instance, a Boolean function Φ over the following variables is created:

Definition 4. *Consider a $w \times h$-grid for which a routing over T time steps shall be determined. Then, the Boolean variables $d^t_{x,y,i}$ with $(x, y) \in C$, $i \in D$, and $1 \leq t \leq T$ represent whether there is a droplet with identifier i present on the cell (x, y) at time step t. The value true (false) means present (absent).*

Example 4. *Consider the 5×5-grid shown in Fig. 3(a) which represents a time step t of a possible routing problem. Here, the droplets i_1, i_2, and i_3 are occupying cells $(1, 5)$, $(3, 1)$, and $(5, 2)$, respectively. This is represented by setting all variables as indicated in Fig. 3(b).*

Having all these variables, it is up to the solving engine to assign values for each time step t. Considering all time steps $1 \leq t \leq T$ together, this eventually creates a routing on the given grid. However, without any further constraints, no valid routing can be determined, i.e. neither are droplets and blockages correctly placed onto the grid nor is the desired routing objective enforced. Consequently, constraints over these variables are added.

Enforcing the Source and Target Configuration

First, constraints are introduced enforcing that the source position (x^*_i, y^*_i) and the target position $(x^\dagger_i, y^\dagger_i)$ for each droplet $i \in D$ are properly represented. This can easily be accomplished by

$$\bigwedge_{i \in D} d^{T^*_i}_{x^*_i, y^*_i, i} \wedge d^{T}_{x^\dagger_i, y^\dagger_i, i}. \tag{1}$$

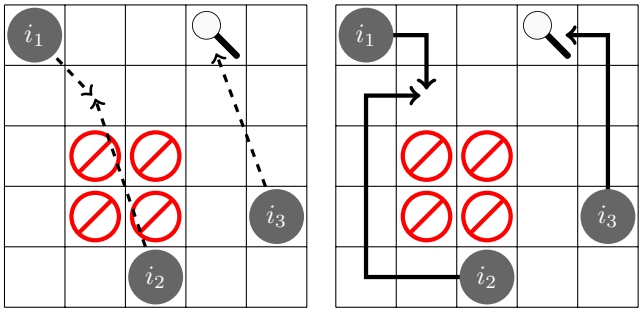
Example 5. *Consider the routing problem for a 5×5-grid as shown in Fig. 4(a), which serves as running example for the rest of this section. The droplets i_1 and i_2 as well as the target cell $(2, 4)$ form a three-pin net, while droplet i_3 and the target cell $(4, 5)$ form a two-pin net. The following constraints enforce the source and target position for the droplets i_1, i_2, and i_3 with $T = \max_{i \in D} T^\dagger_i = 4$:*

$$d^1_{1,5,1} \wedge d^4_{2,4,1} \wedge d^1_{3,1,2} \wedge d^4_{2,4,2} \wedge d^1_{5,2,3} \wedge d^4_{4,5,3}.$$

Formulating the Movement of Droplets

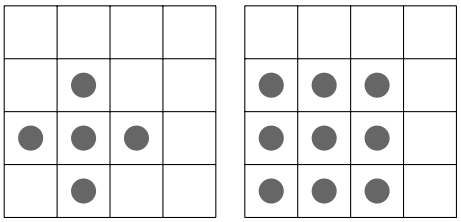
In order to formulate the movement of droplets, we need the notation of a *neighborhood* of an (x, y)-cell. More precisely, we introduce a *5-neighborhood* $N_5(x, y)$ which contains all the cells a droplet at cell (x, y) can reach within one time step. Fig. 5(a) provides a visualization of the 5-neighborhood for the cell $(2, 2)$. Having this, a droplet i which is present at a certain cell (x, y) at a certain time step t requires that the same droplet either spawned at that time step (this is already covered by the constraints from Eq. (1)) or occupied another cell within the 5-neighborhood of (x, y) the time step before. The latter requirement is formulated as

$$\bigwedge_{i \in D} \bigwedge_{(x,y) \in C} \bigwedge_{T^*_i < t \leq T} \left(d^t_{x,y,i} \Rightarrow \bigvee_{(x',y') \in N_5(x,y)} d^{t-1}_{x',y',i} \right). \tag{2}$$



(a) Routing Problem (b) Solution

Fig. 4: Running example



(a) $N_5(2, 2)$ (b) $N_9(2, 2)$

Fig. 5: Neighborhoods of the cell $(2, 2)$

Example 6. *Consider again the running example shown in Fig. 4(a). The constraint representing the movement of droplet i_3 from its starting position $(5, 2)$ at time step $t = 1$ to the cell $(5, 3)$ at time step $t = 2$ is*

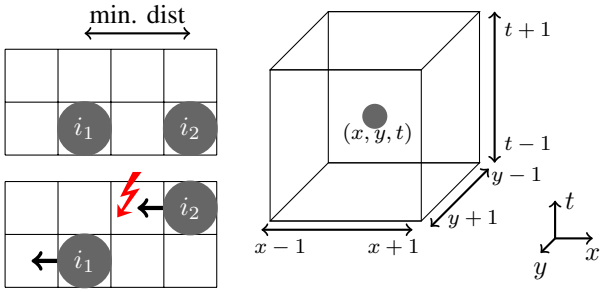
$$d^2_{5,3,3} \Rightarrow \ (d^1_{5,3,3} \vee d^1_{5,2,3} \vee d^1_{5,4,3} \vee d^1_{4,3,3}).$$

Note that the 5-neighborhood does not contain the cell $(6, 3)$ as $(6, 3) \notin C$. The constraints for the remaining cells, droplets, and time steps are generated in a similar fashion.

Fluidic Constraints

An important constraint to be considered during the routing is that in order to avoid unintended mixing, two droplets that are not supposed to be mixed must not occupy adjacent cells [5]. For this purpose, another extended neighborhood notation is applied, i.e. we introduce a *9-neighborhood* $N_9(x, y)$ which contains all the cells surrounding (x, y). Fig. 5(b) provides a visualization of the 9-neighborhood for the cell $(2, 2)$. Having this, each droplet must not occupy the 9-neighborhood of another droplet (i.e. all droplets must keep their minimal distance as illustrated at the top of Fig. 6(a)) and each droplet must not enter the 9-neighborhood constituted by the occupation of another droplet one time step before (as illustrated at the bottom of Fig. 6(a)). For each droplet, this eventually leads to a cube in the position-time-space which no other droplet is allowed to enter (see Fig. 6(b)).

However, fully enforcing these fluidic constraints is in contradiction to the intended realization of three-pin nets, i.e. the determination of a routing of two droplets to the same target cell. To resolve this contradiction, fluidic constraints



(a) On the grid (b) As a cube in the position-time-space

Fig. 6: Visualization of the fluidic constraints

are usually applied between different nets only (see e.g. [5], [6]). In order to formulate this, the set of droplets D is partitioned according to the given nets. Fluidic constraints are only enforced between these partitions. The set of all partitions is denoted by P.

Overall, this allows for the formulation of fluidic constraints as follows:

$$\bigwedge_{p\in P}\bigwedge_{i\in p}\bigwedge_{\substack{j\in D\\ j\notin p}}\bigwedge_{\substack{1\le t\le T\\ (x,y)\in C}}\left(d^t_{x,y,i} \Rightarrow \bigwedge_{\substack{(x',y')\in\\ N_9(x,y)}} \overline{d^t_{x',y',j}} \wedge \bigwedge_{\substack{(x',y')\in\\ N_9(x,y)}} \overline{d^{t-1}_{x',y',j}}\right) \tag{3}$$

Example 7. *Consider again the running example shown in Fig. 4(a). The three-pin net and the two-pin net lead to a partitioning of the set $D = \{i_1, i_2, i_3\}$ of all droplets into $P = \{\{i_1, i_2\}, \{i_3\}\}$. That is, fluidic constraints are applied between i_1 and i_3, i_2 and i_3, i_3 and i_1, as well as i_3 and i_2. For droplet i_1, time step $t = 4$, and cell $(5, 5)$, this leads to*

$$d^4_{5,5,1} \Rightarrow\ (\overline{d^4_{5,5,3}} \wedge \overline{d^4_{5,4,3}} \wedge\ \overline{d^4_{4,5,3}} \wedge \overline{d^4_{4,4,3}} \wedge \overline{d^3_{5,5,3}} \wedge \overline{d^3_{5,4,3}} \wedge\ \overline{d^3_{4,5,3}} \wedge \overline{d^3_{4,4,3}}).$$

As the position is in the corner of the grid, only four cells are in the 9-neighborhood. The constraints for the remaining cells, droplets, and time steps are generated in a similar fashion.

Enforcing Blockages

In order to represent the blockages on the grid, the values of the $d^t_{x,y,i}$-variables have to be restricted properly. This also enables an easy consideration of temporary blockages. In fact, it is sufficient to enforce that for each blockage and at the respective time intervals, no droplet is present at the respective cells. This is formulated by

$$\bigwedge_{(x,y,t_1,t_2)\in B}\ \bigwedge_{i\in D}\ \bigwedge_{t_1\le t\le t_2} \overline{d^t_{x,y,i}}. \tag{4}$$

Example 8. *Consider again the running example shown in Fig. 4a. The set B, describing the blockages, is given by $B = \{(2, 2, 1, 4), (2, 3, 1, 4), (3, 2, 1, 4), (3, 3, 1, 4)\}$. Every blockage has the interval $[1, 4]$ assigned to it. The constraint representing this blocking in the SMT instance is*

$$\overline{d^1_{2,2,1}} \wedge \overline{d^1_{2,2,2}} \wedge \overline{d^1_{2,2,3}} \wedge \cdots \wedge \overline{d^4_{3,3,1}} \wedge \overline{d^4_{3,3,2}} \wedge \overline{d^4_{3,3,3}}.$$

TABLE I: Characteristics of the applied benchmarks

Benchmark	Size	# sub-probl.	# nets	D_{max}
in-vitro1	16×16	11	28	5
in-vitro2	14×14	15	35	6
protein1	21×21	64	181	6
protein2	13×13	78	178	6

Consistency Constraints

Finally, further constraints are added ensuring that the obtained solution obeys certain consistency properties. More precisely,

- from the spawn time to the target time, each droplet must occupy exactly one cell per time step, i.e. $\bigwedge_{i\in D}\bigwedge_{T^*_i\le t\le T^\dagger_i}\left(\sum_{(x,y)} d^t_{x,y,i} = 1\right)$,
- before its spawn time, each droplet must not occupy any cell, i.e. $\bigwedge_{i\in D}\bigwedge_{1\le t<T^*_i}\left(\sum_{(x,y)} d^t_{x,y,i} = 0\right)$, and,
- after the target time, each droplet may, but does not have to occupy a cell, i.e. $\bigwedge_{i\in D}\bigwedge_{T^\dagger_i<t\le T}\left(\sum_{(x,y)} d^t_{x,y,i} \le 1\right)$.

Solving and Interpreting the Solution

A conjunction of the constraints introduced above in Equations (1)-(4) together with the consistency constraints eventually yields an SMT formulation Φ of a decision problem which states that a given routing problem can be solved in T time steps. The resulting formulation is then passed to a corresponding solving engine. If the solver returns unsatisfiable, it has been proven that no routing with T time steps exists and T is increased accordingly. If instead the solver returns satisfiable, a precise routing, i.e. the positions of all droplets for all time steps, can be derived from the respective $d^t_{x,y,i}$-variables (as illustrated before in Fig. 3).

Example 9. *Consider again the running example shown in Fig. 4(a). Passing the formulation as sketched above for $T = 6$ yields a satisfying solution including e.g. $d^1_{1,5,1}$, $d^2_{2,5,1}$, and $d^3_{2,4,1}$ set to 1. From this assignment, the routing of droplet i_1 can be derived. In a similar fashion, the routing of the other droplets are obtained eventually leading to the overall solution as shown in Fig. 4(b).*

V. Experimental Evaluation

The proposed approach has been implemented in a Ruby program that generates the respective instances described above using the SMT-LIB2 format [13] extended by a logic for cardinality constraints. The extension has been implemented on top of the open source toolkit *metaSMT* [14]. As solving engine we utilized the SMT solver *Z3* [15]. Afterwards, the proposed approach has intensely been evaluated on a 2.6 GHz Intel Core i5 machine with 8 GB of memory running 64bit Xubuntu 13.10. This section summarizes and discusses the results of the conducted experiments.

A. Comparison to Previous Work

As reviewed in Section II-C, determining optimal results for relevant benchmarks suffered from the respective complexity – thus far, no optimal results have been obtained in feasible run time. In a first series of experiments, we evaluated whether the SMT-based approach presented in this work can cope with this problem. For this purpose, we applied benchmarks which have previously been used to evaluate the (heuristic) approaches from [8], [9], [10] to the proposed exact approach. The characteristics of these benchmarks, i.e. their name, grid-size, number of routing sub-problems (denoted as *# sub-probl.*), number of nets, as well as largest number of droplets to be considered (denoted as D_{max}), are summarized in Table I.

TABLE II: Comparison to previous work

	Fluidic constraints with $N_9(x,y)$							Fluidic constraints with $N_5(x,y)$				
	Progressive ILP [9]		Two-stage SAT [10]		Proposed exact approach			Entropy [8]		Proposed exact approach		
Benchmark	max T	avg. T	max T	avg. T	max T	avg. T	Time	max T	avg. T	max T	avg. T	Time
in-vitro1	24	13.09	19	12.36	19	12	1726.3	18	12.47	18	11.2	1091.8
in-vitro2	21	10.93	20	10.20	16	10.07	907.8	17	10.43	16	10.07	793.9
protein1	26	16.15	23	15.78	20	15.28	11766.9	20	15.51	20	15.28	9754.2
protein2	29	10.47	21	9.25	20	9.54	4958.9	20	10.04	20	9.53	3950.5

Results are provided in Table II. For each benchmark (composed of several routing problems) and each approach, the maximal and average number of time steps needed to perform the respective routings are given (denoted as max T and avg. T, respectively). Additionally, for the proposed approach, the required run-time (in CPU seconds) is provided. Note that previously proposed approaches apply different interpretations of *fluidic constraints*: In [8], a 5-neighborhood instead of a 9-neighborhood is used to avoid unintended mixing in the moving step. Our approach can handle both interpretations by simply replacing $N_9(x,y)$ in the right part of the constraint in Eq. (3) with $N_5(x,y)$. To allow a consistent evaluation, the respective results are distinguished accordingly in Table II.

The results clearly confirm that *exact* results can be determined in feasible run time for all benchmarks. For the first time, this enables a qualitative comparison of previously obtained (heuristic) results to the actual minimum. In fact, using the exact approach, routings with a lower maximal as well as the average number of time steps can be generated[2]. Nevertheless, since the differences are rather small, we were able to confirm that the previously proposed heuristics are already quite close to the actual optimum. Besides that, it is interesting to see that the different interpretations of the fluidic constraints have very little effect on the resulting routing.

B. Exploitation of Temporary Blockages

As discussed in Section III, previously proposed approaches ignore the fact that blockages are often not present the entire time but only in certain intervals. Thus far, this prevented the determination of better routings. This has been evaluated in a second series of experiments. For this purpose, the routing problems introduced in [8, Fig. 1 and 2] and the 59^{th} subproblem of the *protein2*-benchmark have been applied as benchmarks. The blockages have additionally been enriched with temporal information in terms of intervals. More precisely, first we determined the minimal number of time steps assuming all blockages to be present the entire time (denoted by *init.* T). Afterwards, T is applied to specify the length of the respective intervals with respect to different percentage values: For example, if $T = 10$ then blockages are assumed to be present onto the grid for 9 (90%), 7 (70%), and 5 (50%) time steps. The precise intervals are determined randomly. Blockages which can be merged into rectangles containing more than one cell are assumed to share the same interval. This creates proper benchmarks which allow to observe the effect of temporary blockages.

Results are provided in Table III. The initially determined number of time steps (*init.* T) denotes the best possible result available thus far. Then, for each benchmark and each length of intervals, 10 instances are considered. The best (*min*) and average (*avg.*) number of time steps obtained from those instances are reported in the remaining columns. The results show that routing can indeed be significantly improved if blockages are not assumed the entire time. In the best case, the number of time steps is reduced from 16 to 4. This demonstrates the potential of using timing information of blockages.

[2]Only one exception exists for the *protein2*-benchmark, where a smaller average value is reported in [10]. Even after thorough evaluations, we were not able to re-produce this result from [10].

TABLE III: Exploitation of temporary blockages

Benchmark	Size	init. T	90% min/avg.	70% min/avg.	50% min/avg.
protein2.59	13×13	16	16/16	4/11.1	4/5
[8, Fig. 1]	13×13	15	15/15	15/15	13/14.4
[8, Fig. 2 (a)]	13×13	13	13/13	13/13	10/11.3
[8, Fig. 2 (b)]	16×16	24	24/24	23/23.0	21/22.7

VI. Conclusion & Outlook

In this work we considered the exact routing for digital microfluidic biochips. Previously proposed solutions suffer from the fact that they are of heuristic nature only and assume blockages to be present the entire time. We proposed an SMT-based approach which addresses both issues. For the first time, this enabled the generation of minimal results and, hence, a qualitative comparison of previously proposed approaches to the actual minimum. Furthermore, the potential of considering temporary blockages has been shown. This motivates a deeper consideration of this timing information during routing processes and provides the basis for further work in this direction.

References

[1] J. Song, R. Evans, Y.-Y. Lin, B.-N. Hsu, and R. Fair, "A scaling model for electrowetting-on-dielectric microfluidic actuators," *Microfluidics and Nanofluidics*, vol. 7, no. 1, pp. 75–89, 2009.

[2] T.-Y. Ho, J. Zeng, and K. Chakrabarty, "Digital microfluidic biochips: A vision for functional diversity and more than Moore," in *Int'l Conf. on CAD*. IEEE Press, 2010, pp. 578–585.

[3] "Microfluidic applications in the pharmaceutical, life sciences, in-vitro diagnostic and medical device markets report 2013," http://www.researchandmarkets.com/research/z6nhg7/microfluidic.

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

[5] F. Su, W. Hwang, and K. Chakrabarty, "Droplet routing in the synthesis of digital microfluidic biochips," in *Design, Automation and Test in Europe*, vol. 1. IEEE, 2006, pp. 1–6.

[6] P.-H. Yuh, C.-L. Yang, and Y.-W. Chang, "Bioroute: A network-flow based routing algorithm for digital microfluidic biochips," in *Int'l Conf. on CAD*. IEEE Press, 2007, pp. 752–757.

[7] M. Cho and D. Z. Pan, "A high-performance droplet routing algorithm for digital microfluidic biochips," *IEEE Trans. on CAD*, vol. 27, no. 10, pp. 1714–1724, 2008.

[8] T.-W. Huang and T.-Y. Ho, "A fast routability- and performance-driven droplet routing algorithm for digital microfluidic biochips," in *Int'l Conf. on Comp. Design*. IEEE, 2009, pp. 445–450.

[9] P.-H. Yuh, S. Sapatnekar, C.-L. Yang, and Y.-W. Chang, "A progressive-ilp based routing algorithm for cross-referencing biochips," in *Design Automation Conf.* ACM, 2008, pp. 284–289.

[10] P.-H. Yuh, C. C.-Y. Lin, T.-W. Huang, T.-Y. Ho, C.-L. Yang, and Y.-W. Chang, "A SAT-based routing algorithm for cross-referencing biochips," in *System Level Interconnect Prediction Workshop*. IEEE Press, 2011, pp. 6:1–6:7.

[11] K. Hu, B.-N. Hsu, A. Madison, K. Chakrabarty, and R. Fair, "Fault detection, real-time error recovery, and experimental demonstration for digital microfluidic biochips," in *Design Automation Conf.* EDA Consortium, 2013, pp. 559–564.

[12] O. Keszocze, R. Wille, and R. Drechsler, "Exact One-pass Synthesis of Digital Microfluidic Biochips," in *Design Automation Conf.*, 2014.

[13] C. Barrett, A. Stump, and C. Tinelli, "The SMT-LIB Standard: Version 2.0," in *International SMT Workshop (Edinburgh)*, A. Gupta and D. Kroening, Eds., 2010.

[14] F. Haedicke, S. Frehse, G. Fey, D. Große, and R. Drechsler, "metaSMT: Focus on your application not on solver integration," in *International Workshop on Design and Implementation of Formal Tools and Systems*, 2011, pp. 22–29, metaSMT is available at https://github.com/agra-uni-bremen/metaSMT.

[15] L. De Moura and N. Bjørner, "Z3: An efficient SMT solver," in *Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2008, pp. 337–340, Z3 is available at http://z3.codeplex.com/.