Sound Valve-Control for Programmable Microfluidic Devices

Andreas Grimmer[1] Berislav Klepic[1] Tsung-Yi Ho[2] Robert Wille[1]
[1]Institute for Integrated Circuits, Johannes Kepler University, Linz, Austria
[2]National Tsing Hua University, Taiwan
andreas.grimmer@jku.at berislav.klepic@gmail.com tyho@cs.nthu.edu.tw robert.wille@jku.at

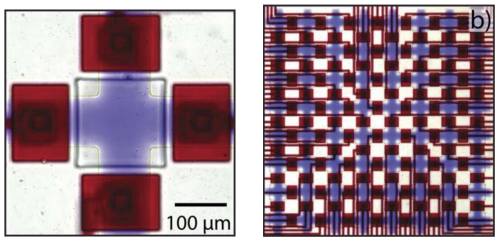
Abstract—In the domain of microfluidic devices, a paradigm shift from application-specific to fully-programmable solutions takes place (a similar development from ASICS to FPGAs has been observed in conventional circuitry). So-called *Programmable Microfluidic Devices* (PMDs) provide a promising platform in this regard. Here, fluids can be pushed into various reaction vessels whose inflow and outflow is controlled by valves. The regular structure in combination with the flexibility of defining various flow paths through valves allows to realize a vast range of biological or chemical applications by only changing the corresponding valve-control sequence. However, determining a sound valve-control constitutes a non-trivial task. Although first automatic approaches for this problem have recently been proposed, we show that they frequently yield impractical control sequences. In this work, we address this issue by providing a precise definition of the underlying design task. Afterwards, we present complementary solutions (both exact as well as heuristic) and discuss how they guarantee a sound valve-control. Experimental evaluations demonstrate that the proposed solutions are capable of automatically generating a sound valve-control for PMDs.

I. Introduction

Microfluidics bears the potential to revolutionize the fields of chemical synthesis and biological analysis [1]. Corresponding devices efficiently handle and process sample volumes in the nano- to pico-liter range and, by this, allow to automatically conduct complex applications such as protein crystallization, high-throughput screening in drug development, or single cell analysis [1, 2] using very small volumes of samples and reagents.

The majority of the corresponding devices are targeted and developed for specific applications. Here, a crucial task in the development of a corresponding device is its design and physical implementation which usually requires expert knowledge in microfabrication and fluid physics – yielding high costs in their development [2, 3]. This is similar to the domain of integrated circuits where, due to similar reasons, the development costs for an *Application-Specific Integrated Circuit* (ASIC) are substantial as well. Because of that, *Field Programmable Gate Arrays* (FPGAs) got established as a cheaper alternative which may not be as application-specific as an ASIC, but still allows designer to realize different functionality in hardware.

A similar development can currently be observed for microfluidics, where so-called *Programmable Microfluidic Devices* (PMDs, [2, 4]; also known as *Microfluidic Fully Programmable Valve Arrays* e.g. in [5]) have recently been introduced. They basically represent a counterpart to FPGAs for the microfluidic domain, onto which several operations usually conducted on an application-specific continuous flow-based device can be realized. PMDs offer similar advantages as FPGAs including a short time-to-market, a simple design cycle, a predictable project cycle, and also field reprogrammability. As FPGAs, they may also serve as first prototypes for application-specific realizations of flow-based chips.



(a) Node with four valves [2] (b) Grid of 64 nodes [2]

Fig. 1. Physical realization of a PMD

These advantages are possible because of the regular and controllable structure of PMDs: They are composed of so-called *nodes* which are arranged in terms of a rectangular grid. Each of these nodes is used as reaction vessel and is surrounded by up to four *valves* that allow a full control of the flow of samples to and from adjacent nodes. Fig. 1a shows a realization of a node. The red blocks are the four valves which can be individually opened or closed. These nodes are composed to larger structures as e.g. done in [2] – yielding a PMD as shown in Fig. 1b. This realization consists of 64×300 pico-liter nodes which are controlled by 114 individually addressable valves.

Using such structures allows to realize a vast range of applications by only changing the control of the valves – including but not limited to fluid metering and active mixing, surface immunoassays, as well as cell culture (see [2] for an overview). Therefore, operations can either directly be implemented on the PMD (e.g. active mixing is accomplished by activating valves that produce a peristaltic flow which mixes samples while incubating of samples is accomplished by using the nodes as reaction vessels) or realized by additional external hardware (e.g. detectors, cameras, or reservoirs). Videos at `https://goo.gl/T7JQ4t` show a physical realization of a PMD executing diverse applications.

However, properly controlling the respective valves (i.e. determining a *valve-control*) so that indeed the desired application is realized is left to be done by the bioengineers using the PMD. This is highly non-trivial as it requires the concurrent consideration of multiple samples which all have to be pushed to their desired targets and compete for resources. Moreover, in order to push a sample, a continuous open sequence of nodes, i.e. a *flow path*, from an input to an output is required [2].

In order to support bioengineers in this task, first automatic design approaches have recently been proposed in [6]. But we will show in this work that they frequently yield impractical results. More precisely, the proposed approaches do not consider the realization of sound *flow paths* and, hence, generate valve-controls which eventually do not realize the desired application. Besides that, the proposed approaches consider samples only flowing from inputs to outputs (both located on the boundary of the PMD), but not samples contained on the grid itself – further significantly restricting the applicability.

In this work, we address these issues by introducing a precise definition of the design task. Using this definition, we

then present two *sound* methods which guarantee that the desired application is realized on a PMD. Both solutions are thereby complementary to each other: The first one realizes an *exact* approach which ensures that objectives (e.g. a maximal time limit for pushing all samples) given by the bioengineer are satisfied. Therefore, we use a symbolic formulation which is passed to a *Satisfiability Modulo Theories* solver (SMT solvers, see e.g. [7]). The second method is a *heuristic* approach which is based on established and sophisticated pathfinding algorithms. Both solutions have been made publicly available at `http://www.jku.at/iic/eda/pmd` so that bioengineers can use them and to allow other researchers to extended and adopted them.

The remainder of this paper is structured as follows: The next section introduces the realization of applications on PMDs and gives a precise definition of the valve-control design task. Section III motivates the work by reviewing the limitations of the related work. In Section IV and Section V, details on the two proposed solutions are provided. The results of our evaluation are summarized in Section VI and, finally, the paper is concluded in Section VII.

II. Realizing Applications on PMDs

Programmable Microfluidic Devices (PMDs, [2,4]) are used to conduct applications by controlling and manipulating continuous flows of samples. The flows are actively controlled using *micro-mechanical valves* as e.g. proposed in [8] and [9]. These valves are employed in a multi-layer platform composed of a flow- and control-layer. More precisely, by applying a pressure to the control-layer, a valve can be closed which seals the channel in the flow-layer. After releasing the pressure, the valve moves back and the flow in the channel resumes.

Four of these valves are used to build a *node* and these valves allow to control the sample flow. The nodes are used for conducting operations (e.g. mixing and incubating [2,4]) on samples. The samples are injected from *inputs* located at the border of the PMD, which are also used to apply an external pressure pushing the samples. Similarly, a PMD contains *outputs* which are connected to waste-chambers or output-reservoirs. More formally, the structure of a PMD can be defined as follows:

Definition 1 *PMDs are represented as a* grid *of nodes of size $W \times H$, where W is the width and H is the height. The nodes on the PMD are uniquely addressed by positions $P = \{(x,y) : 0 \leq x < W \wedge 0 \leq y < H\}$. Blockages B (with $B \subseteq P$) are blocked nodes and cannot be used. The set $In = \{in_a : 0 \leq a < 2 \cdot (W+H)\}$ and the set $Out = \{out_a : 0 \leq a < 2 \cdot (W+H)\}$ (where $In \cap Out = \emptyset$) identify positions at the border (in a clock-wise order starting at the left-top corner) connected to inputs and outputs, respectively.*

Example 1 *Fig. 2 shows a schematic of a PMD of size 7×7 (the valves have oval shapes and are filled red in case they are closed). The blockage $B = \{(4,1),(5,1),(4,2),(5,2)\}$ is highlighted in terms of a black-patterned area and its nodes cannot be used. Furthermore, this PMD has two inputs at border positions in_2 and in_3 as well as two outputs at border positions out_{14} and out_{15}.*

An application can be described using a protocol in form of a sequencing graph. Such graphs define the used samples, the operations to be executed, and their dependencies.

Example 2 *Fig. 3 shows an excerpt of a sequencing graph. This excerpt defines that samples (s_1 and s_2) first have to be mixed by operations (m_1 and m_2) and, afterwards, have to be detected by operations (d_1 and d_2).*

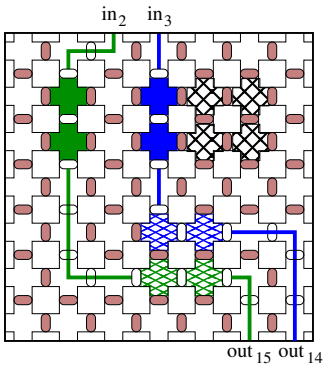


Fig. 2. Schematic of a PMD

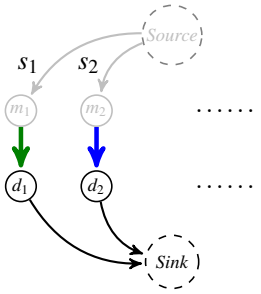


Fig. 3. Sequencing graph excerpt

For processing a sequencing graph on a PMD, multiple steps including scheduling, binding, and placement of the operations as well as washing have to be conducted. For this purpose, existing approaches of related technologies and especially of *flow-based microfluidic biochips* can be re-used with only minor modifications. Besides others, the work published in [10] gives a comprehensive overview of the proposed algorithms.

However, an important task dedicated for PMDs remains left: how to properly control the valves? More precisely, the valves have to be configured so that samples can flow from their source positions to their target positions. For a sample, the source position can be a dedicated input from which it is injected or any arbitrary position on the PMD. Similarly, the target position can be a dedicated output or another arbitrary position on the PMD[1]. In order to push the sample through the grid, a continuous *flow path* is required [2] which is to be realized by a sequence of open valves connecting nodes from an *input* (injecting the pressure) through the sample (pushing the sample) eventually to an *output* (allowing air and/or other fluids to leave the PMD).

The flow of a sample from a source position to a target position requires an amount of time. This time depends, besides physical properties such as the applied pressure and the viscosities of the samples, on the length of the path covered by the sample (in terms of nodes). As in [6], we define a *time step* to denote the "real time" a sample requires to flow from one node to an adjacent position. In each time step where the sample should flow towards its target, a flow path is required. During all these time steps, the nodes on the flow path are occupied and cannot be used for other flow paths or operations.

In order to realize an application described by a sequencing graph, such flow paths have to be determined (and, afterwards, realized by properly controlling the respective valves) for all time steps. This can be formally described as follows:

Definition 2 *Let $S = \{s_0, s_1, \ldots s_{N-1}\}$ be a set of samples with N being its number. Each of these samples $s \in S$ has a length l_s (in terms of nodes), as well as source positions $src_s = \{src_s[i] : src \in P \cup In \text{ and } 0 \leq i < l_s\}$ and target positions $tgt_s = \{tgt_s[i] : tgt \in P \cup Out \text{ and } 0 \leq i < l_s\}$. Then, for pushing a sample s, a flow path from any input in $\in I$, through the sample s, and finally to any output $out \in Out$ has to be determined. In detail, in one time step a sample is pushed exactly one node further (towards the output) in the flow path. To this end, it has to be ensured that the flow path of s does not intersect/overlap with the flow path of another sample $s' \in S \setminus \{s\}$ or an operation. Overall, the task is to push all samples of S to their target positions by determining flow paths in each required time step.*

[1] Note that, depending on the sample volume and the volume of a node on the PMD, source and target positions may be defined by multiple nodes on the grid.

Example 3 *Again, consider the sequencing graph shown in Fig. 3. Further, let's assume the mixing operations m_1 and m_2 have already been completed and produced two samples s_1 with $l_{s_1} = 2$ and s_2 with $l_{s_2} = 2$. Fig. 2 shows the intermediate state of the PMD where sample s_1 is highlighted green and sample s_2 is highlighted blue. The task is now to push both samples from their source positions ($src_{s_1} = \{(1,2),(1,1)\}$ and $src_{s_2} = \{(3,2),(3,1)\}$ which are shown as filled areas) to their new target positions ($tgt_{s_1} = \{(4,5),(3,5)\}$ and $tgt_{s_2} = \{(4,4),(3,4)\}$ which are shown as patterns) where the next detecting operations d_1 and d_2 can be conducted. Fig. 2 shows possible flow paths by means of green and blue lines, e.g. using the green flow path for 6 time steps would allow to push sample s_1 to its target tgt_{s_1}.*

As a further objective besides the introduced task and flow path, the valve-control sequence should take as less as possible time steps. However, due to the limited resources (area of the PMDs, other operations, inputs, and outputs) and intersections of flow paths, it is likely that not all samples can be pushed concurrently.

III. Related Work and Motivation

Determining a sound valve-control which realizes applications on PMDs as reviewed above is a highly non-trivial task. For each sample $s \in S$, it has to be defined how to push the sample through the grid, in what time step should the sample be pushed, and how to realize the respectively needed flow path. Moreover, many applications require the consideration of multiple samples concurrently. As these samples compete for nodes, inputs, and outputs, it is cumbersome and error-prone to manually determine a control sequence for the valves for all time steps. Additionally, the overall number of time steps required to complete the task should be minimized.

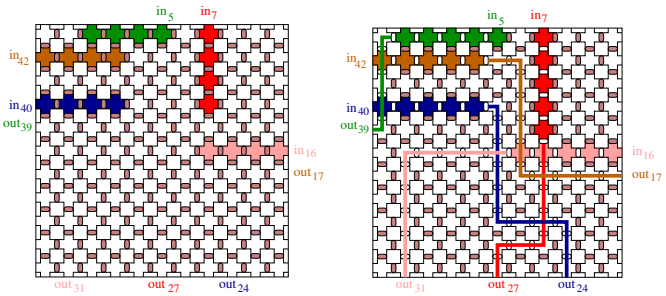
Motivated by that, the development of automatic design solutions for a valve-control received interest. First approaches towards this have been presented in [6]. They are used to solve tasks such as illustrated by the following example:

Example 4 *Consider the PMD shown in Fig. 4a which has five inputs ($In = \{in_5, in_7, in_{16}, in_{40}, in_{42}\}$) and five outputs ($Out = \{out_{17}, out_{24}, out_{27}, out_{31}, out_{39}\}$). The task considered and taken from [6] is to determine valve states which allow five samples $S = \{s_1, \ldots, s_5\}$ to flow from a corresponding input to an output. For example, the sample s_1 with final length $l_{s_1} = 5$ (shown as green sample in Fig. 4a) has to flow from the input in_5 to the output out_{39}.*

However, the approaches presented in [6] come with major limitations:

- The approaches only support a sample flow from an input to an output but do not support a sample flow starting or ending on the PMD. More formally, the source position src_s and target position tgt_s can only be an input $in \in In$ or an output $out \in Out$, respectively. Consequently, it does not allow a sample to flow to a position on the grid where e.g. a mixing operation can take place. As a result, these approaches cannot be used for the implementation of a sequencing graph describing a practical application.
- An even more significant limitation of the approaches is that they do not consider the realization of a flow path, i.e. samples might be pushed without a guaranteed sequence of open valves connecting nodes from an input (injecting the pressure) through the sample to an output (cf. Definition 2). Because of this, solutions are frequently obtained which cannot be executed on a PMD.

In fact, these two limitations can be observed by analyzing the results obtained and reported in [6]:



(a) PMD state after 4 time steps (b) PMD state after 5 time steps

Fig. 4. A result obtained from [6], which cannot be executed on a PMD.

Example 5 *Fig. 4a and 4b show results for the task specified in Example 4 for time step 4 and 5, respectively. It can be seen that only samples which flow from inputs to outputs are considered (first limitation). Moreover, the obtained results ignore the fact that dedicated flow paths have to be realized for each sample (second limitation). In fact, the lines in Fig. 4b show the flow paths which would be necessary to realize the shown movement. Since these paths constantly cross each other, they could not be realized on a PMD. The approaches presented in [6] oversimplify the problem and do not yield a sound valve-control.*

In the remainder of this work, we show that the methods presented in [6] indeed almost always determine a valve-control sequence which is not sound and does not realize the intended application (cf. Section VI). However, before that, we introduce alternative methods which overcome the limitations of this previous work in two complementarily different fashions. The first one (described in Section IV) solves the design task in an *exact* fashion by considering the entire search space and, hence, obtains solutions guaranteeing the bioengineer's objectives. Since this usually requires large computational power, the second one (described in Section V) solves the design task using a *heuristic* which still generates good solutions (guaranteeing a sound valve-control) in acceptable runtimes.

IV. Exact Determination of a Valve-Control

In this section, the exact method for determining a sound valve-control is presented. In this regard, "exact" means that the method either determines a valve-control satisfying all objectives or proves that no such solution exists. To this end, we utilize solvers for *Satisfiability Modulo Theories* (SMT solvers, see e.g. [7]) which have been proven effective for other design automation tasks e.g. for digital microfluidic biochips [11, 12], for continuous-flow microfluidic biochips [13], and for microelectrode-dot-array digital microfluidics [14].

In order to use SMT solvers, the considered problem is formulated as a decision problem, i.e. "Is there a sound valve-control sequence which pushes all samples from their source position to the desired target position, ensures valid flow paths, and additionally completes this within a given upper bound T of time steps?". To this end, we first introduce the symbolic formulation representing all possible solutions. Afterwards, we describe the constraints enforcing that not an arbitrary solution, but the desired one (being sound and satisfying all objectives) is obtained. Finally, we discuss the pros and cons of the proposed exact method at the end of this section.

A. Symbolic Formulation

To symbolically formulate all possible solutions, it is not necessary to represent the state of all valves using dedicated variables. Instead, we use a symbolic formulation modeling the sample positions, the used inputs and outputs, as well as the flow paths. A solution (i.e. an assignment of the variables) then

allows to derive the respective valve states. More precisely, we first introduce variables representing the positions of cells which are occupied by a sample at a particular time step.

Definition 3 *For each sample $s \in S$ with length l_s and for every time step t with $0 \leq t < T$, we introduce* position-*variables $pos_s^t = \{(pos_s^t[i].x, pos_s^t[i].y) : 0 \leq i < l_s\}$ describing the occupied nodes of the grid. Each pair $(pos_s^t[i].x, pos_s^t[i].y)$ are two integer-typed variables. The position at index 0 (i.e. $(pos_s^t[0].x, pos_s^t[0].y)$) is called* head *and the position at index $l_s - 1$ (i.e. $(pos_s^t[l_s-1].x, pos_s^t[l_s-1].y)$) is called* tail.

Example 6 *Consider the PMD state of time step $t = 0$ shown in Fig. 2. The positions of sample s_1 and s_2 are described by $pos_{s_0}^0 = \{(1,2),(1,1)\}$ and $pos_{s_1}^0 = \{(3,2),(3,1)\}$, respectively.*

Next, we introduce a formulation which allows to model the flow path. Recall, external pumps connected to the inputs produce a pressure which is used to push the sample inside the chip and this pressure has to exit the chip through outputs. To represent which input is used to push the sample and which output is used to release the pressure, we introduce the following variables:

Definition 4 *For each sample $s \in S$ and for every time step t with $0 \leq t < T$, we introduce two integer-typed variables in_s^t and out_s^t. Their respective values describe which input/output is used for pushing the sample/allowing the pressure to leave the PMD.*

Example 7 *Consider the PMD state shown in Fig. 2. The input in_2 is used for pushing sample s_1 at time step $t = 0$, i.e. $in_{s_1}^0 = 2$. Similarly, the output out_{15} is used for releasing the pressure of sample s_1, i.e. $out_{s_1}^0 = 15$.*

Finally, we introduce a symbolic representation of the flow path between these inputs/outputs and the sample. Therefore, a flow path can be split in two sequences, i.e. a sequence of open valves from any input to the tail of the sample and a sequence of open valves from the head of the sample to any output. In order to describe these two sequences, we do not symbolically model all used nodes which are occupied by the sequences as this would result in a (too) large number of variables. Instead, we represent the flow path with so-called *bend points*. These bend points describe positions on the grid. By connecting them and the used input/output with straight lines, the flow path is described. Bend points are used to describe 90 degree turns of the flow path.

The used number of bend points per sequence, allows to configure how often a sequence can turn by 90 degrees. In the case with only one bend point, the sequence is either a straight line or has an L-shape. Our implementation allows to freely configure the number of bend points and, therefore, gives the bioengineer a mechanism to control the complexity of the resulting solutions and also the complexity of the resulting SMT-instance. Formally, we introduce the following variables:

Definition 5 *For each sample $s \in S$ and for every time step t with $0 \leq t < T$, we introduce* bend point-*variables $bend_s^t = \{(bend_s^t[i].x, bend_s^t[i].y) : 0 \leq i < 2 * k\}$, where k represents the number of used bend points per sequence. Then, the first $0 \leq i < k$ variables $bend_s^t[i]$ are used for the first sequence between the input to the tail of the sample and the remaining k variables are used for the second sequence from the head of the sample to the output.*

Example 8 *Consider again the PMD state and especially the two flow paths for sample s_1 and s_2 as shown in Fig. 2. Assume the bioengineer chooses $k = 2$ bend points per sample. These two flow paths are described by an assignment equal to $bend_{s_1}^0 = \{(2,0),(1,0),(1,5),(5,5)\}$ and $bend_{s_2}^0 = \{(3,0),(3,0),(3,4),(6,4)\}$, respectively. For instance, the first sequence of the flow path for sample s_1 can be obtained by connecting straight lines between input in_2, to $(2,0)$, to $(1,0)$, and finally to the tail of the sample at $(1,1)$. Accordingly, the second sequence of the flow path for sample s_1 can be obtained by connecting straight lines between the head of the sample at $(1,2)$, to $(1,5)$, to $(5,5)$, and finally to the output out_{15}.*

These two flow paths allow the samples to flow one node further, i.e. at time step $t = 1$ the samples are located at $pos_{s_0}^1 = \{(1,3),(1,2)\}$ and $pos_{s_1}^1 = \{(3,3),(3,2)\}$. Note that not all bend point-variables have to change the direction. This is the case for the first sequence represented by $bend_{s_2}^0$, where the first two points do not change the direction and are assigned equally.

The resulting symbolic formulation now allows to represent arbitrary flow paths and, hence, arbitrary valve-controls. However, passing this symbolic formulation to a solving engine would yield an arbitrary assignment of the variables. Hence, it is not guaranteed that the samples correctly flow from their source positions to their target positions and that flow paths are realized. Therefore, we have to restrict the assignments of the symbolic formulation using constraints introduced in the next section.

B. Constraints

Because of space limitations, we only sketch some of the used constraints in the following. While this should be sufficient to get the general idea, a complete implementation is publicly available at `http://www.jku.at/iic/eda/pmd`.

First, constraints are added which ensure that the samples start at time step $t = 0$ at their respective source positions, i.e.

$$\bigwedge_{s \in S} pos_s^0 = src_s.$$

Second, constraints are added which ensure that the samples reach their respective target positions at some time step t_t and afterwards stay there, i.e.

$$\bigwedge_{s \in S} \exists t_t \left(0 \leq t_t < T \wedge \bigwedge_{t_t \leq t < T} pos_s^t = tgt_s \right).$$

Note that these two constraints are adapted when the sample is injected from an input or the sample exits the PMD through an output.

Next, we have to restrict the position-variables pos_s^t to positions P of the grid, i.e.

$$\bigwedge_{s \in S} \bigwedge_{0 \leq i < l_s} \bigwedge_{0 \leq t < T} 0 \leq pos_s^t[i].x < W \wedge 0 \leq pos_s^t[i].y < H.$$

Similarly, the bend point-variables are restricted to positions of the grid. Moreover, some areas on the PMD are blocked e.g. due to other operations. Therefore, we add constraints which ensure that the position- and the bend point-variables cannot take values within these blockages B. Also, the input and output variables are restricted to the respective border position, i.e.

$$\bigwedge_{s \in S} \bigwedge_{0 \leq t < T} in_s^t \in In \quad \text{and} \quad \bigwedge_{s \in S} \bigwedge_{0 \leq t < T} out_s^t \in Out.$$

The constraints introduced so far restrict the allowed values but do not ensure a correct flow of samples including their flow paths. As discussed above, in one time step, a sample can flow

to an adjacent node or can stay. This movement is ensured using the position of the head of the sample from the previous time step $t-1$, i.e.

$$\bigwedge_{s\in S}\bigwedge_{0\le t<T}\underbrace{(pos_s^t[0].x = pos_s^{t-1}[0].x \wedge pos_s^t[0].y = pos_s^{t-1}[0].y-1)}_{Up_s^t}$$
$$\vee \underbrace{(\ldots)}_{Down_s^t} \vee \underbrace{(\ldots)}_{Left_s^t} \vee \underbrace{(\ldots)}_{Right_s^t} \vee \underbrace{(\ldots)}_{Pause_s^t}.$$

In case the sample flows in an adjacent node in time step t, the values of the remaining position variables can be derived from the previous time step $t-1$, i.e.

$$\bigwedge_{s\in S}\bigwedge_{0\le t<T} Up_s^t \vee Down_s^t \vee Left_s^t \vee Right_s^t \rightarrow$$
$$\bigwedge_{1\le i<l_s} pos_s^t[i].x = pos_s^{t-1}[i-1].x \wedge pos_s^t[i].y = pos_s^{t-1}[i-1].y.$$

Otherwise, when the sample pauses its flow in time step t, a constraint assigns the position variables equally to the predecessor time step $t-1$.

In case a sample flows in an adjacent node in time step t, we have to ensure a sound flow path. Therefore, we add constraints which restrict the bend point-variables so that they describe straight lines between the used input and tail of the sample as well as the head of the sample and the used output. This can be done by only allowing at most one coordinate (either x- or y-coordinate) to change its value.

Left is to ensure that no flow path intersects or overlaps with another flow path or a blockage. Therefore, we add constraints which check if lines are crossing or any two samples occupy the same position. Note that these constraints also ensure that inputs and outputs are used by at most one sample at a certain time step.

C. Discussion

The symbolic formulation described above can easily be extended and/or adjusted in order to support further constraints and objectives. Furthermore, this formulation considers *all* possible solutions (including the full consideration of e.g. whether and how samples can concurrently be pushed on the PMD). Overall, this allows to determine the minimum number of time steps required to push all samples to the desired target positions e.g. by first checking the availability of a solution for $T = 1$ time step (or a known lower bound) and increasing T until a sound solution can be determined.

To ensure all that, a rather huge search space has to be considered – making the proposed exact method applicable for small instances only. However, this still proves beneficial e.g. for evaluating how far heuristic solutions are from being minimal. Fortunately, in most cases bioengineers do not need exact solutions. Instead they require methods to quickly obtain a physically correct solution. This motivates a fast, extendible, and scalable method based on a heuristic, which is presented in the next section.

V. Heuristic Determination of a Valve-Control

In this section, the heuristic method for determining a sound valve-control is presented. To cope with the huge search space, we simplify the considered problem by (1) only considering solutions where samples continuously flow from their source positions to their target positions and (2) determining the flow path for each sample one after another rather than concurrently (taking into account valve-sequences already determined for previously considered samples). In contrast to the exact method proposed above, this does not guarantee anymore that a solution within a certain amount of time steps is determined. But therefore, reasonable solutions can be obtained in only a fraction of the time needed by the exact method. Furthermore, compared to the method previously proposed in [6], the determination of a *sound* valve-control is guaranteed.

More precisely, the proposed heuristic solution considers all samples $s \in S$ one after another and generates a flow path composed of the sub-paths

1. from any input $in \in I$ to the tail of the source position $src_s[l_s-1]$,
2. from the head of the source position $src_s[0]$ to the tail of the target position $tgt_s[l_s-1]$, and
3. from the head of the target position $tgt_s[0]$ to any output $out \in Out$.

An open sequence of valves consisting of these three sub-paths gives a sound flow path, which allows the sample to flow from its source to its target.

Determining the three sub-paths reduces the determination of a valve-control for one single sample to a classical pathfinding problem. Here, we apply a standard routing algorithm, namely maze routing with a rip-up and reroute method [15,16]. In the following, we provide an overview of our algorithm.

The algorithm starts at time step $t_s = 0$ and tries to realize as much as possible flow paths. As introduced above, a flow path consists of three sub-paths which are determined by the maze routing method. If a flow path for a sample can be determined, we block the respective nodes for the amount of time steps the sample requires to flow to its target position (i.e. the length of the second sub-path specifies the required time steps). If a sample cannot be completely routed, we apply a rip-up and reroute method. More precisely, we determine which path blocks a successful routing of the currently considered path (i.e. by a wave expansion [15]). Then, we rip-up the respective routing, route the previously unroutable path, and finally also reroute the ripped-up path.

This process is continued until no more samples can be routed starting at the currently considered time step $t_s = 0$. Afterwards, the algorithm tries to continue the routing at a later time step. This next start time step t_s is determined by the arrival time of any of the currently pushed sample at its target position (then, it is guaranteed that the nodes of the flow path are not used anymore). At this time step, the algorithm tries again to realize as much as possible flow paths of the samples which have not yet been completed. This is continued until all samples of S reach their target.

Overall, the method can guarantee that all samples finally reach their target due to the rip-up and reroute method. As a disadvantage, the heuristic cannot guarantee a certain upper bound on the number of needed time steps. However, as evaluations summarized in the next section confirm, acceptable results are usually obtained.

VI. Evaluation

In order to evaluate the proposed methods, we conducted a quality and performance evaluation as well as a comparison to related work. To this end, the exact solution described in Section IV has been implemented in Java and the SMT solver Z3 [17] in its latest version is used. The heuristic solution described in Section V has been implemented partly in C and in Java. For the maze routing with the rip-up and reroute method, we used the open source implementation of [18]. Our implementations are publicly available at `http://www.jku.at/iic/eda/pmd`. As benchmarks, we used test cases from [6] and additionally applied the heuristic solution to *In-Vitro Diagnostics* (IVD) applications which have been taken from [19] and adapted for PMDs by adding two inputs and two outputs. All experiments have been conducted on a 3.8 GHz Intel Core i7 machine with 32GB of memory running 64-bit Ubuntu 16.04. A summary of the obtained results is provided next.

TABLE I
QUALITY AND PERFORMANCE EVALUATION

Case	Exact TS	Exact CPU	Heuristic TS	Heuristic CPU
6x6; 2 Samples	20	308	26	0.093
6x6; 3 Samples	20	21156	26	0.088
6x6; 4 Samples	27	9134	41	0.094
8x8; 2 Samples	25	519	34	0.105
8x8; 3 Samples	25	28465	34	0.136
8x8; 4 Samples	timeout		52	0.106
10x10; 2 Samples	30	10236	41	0.092
10x10; 3 Samples	timeout		41	0.090
10x10; 4 Samples	timeout		64	0.095

A. Quality and Performance

In this section, we compare the obtained quality and the performance of the exact solution with the heuristic solution. The results are summarized in Table I. Both methods produce sound solutions and we use the number of time steps of the obtained result (column "TS") and the computation time (column "CPU") as criteria. Since, due to the computational complexity, the exact solution can only handle small test cases (defined by the grid size and number of samples; see column "Case"), we scaled down the test cases from [6] for this evaluation.

We can observe that the exact solution is capable of determining valve-control sequences which in all cases take less time steps to push all samples to their targets. As a clear drawback, we can see that the exact method is applicable for rather small test cases only and times out for larger instances (a timeout of 10 hours has been applied). In contrast, the heuristic approach always produces results in a fraction of a second. The comparison with the values obtained by the exact approach additionally confirms that the obtained heuristic results are also of feasible quality.

B. Comparison to Related Work

In this section, we compare the proposed solutions to the related work [6] discussed in Section III. Table II summarizes the obtained results for the test cases from [6] and additionally for two IVD applications. In this evaluation, we explicitly checked each result for soundness, i.e. checked whether a flow of a sample is backed by a corresponding flow path. Column "S?" in Table II lists whether (denoted by ✓) or not (denoted by ✗) this is the case.

Our evaluations revealed that all results obtained using method "Routability" and "MIS" of [6] are unsound (making the obtained time steps TS meaningless). Only the method named "Sequential", which realizes the flow of each sample sequentially, produces sound results. That is, already here the superiority of the proposed (heuristic) method compared to the current state-of-the-art is shown: The only solution proposed in [6] which yields sound results (namely method "Sequential") requires up to a factor of 3 more time steps than the method proposed in this work.

Besides that, the evaluation considering the IVD applications confirm further benefits. Since this includes samples whose source and target positions are not necessarily at the inputs and outputs of the PMD, respectively, the previously proposed approaches are not applicable for this case. That is, up to now, such applications have not been supported by any automatic method and required a bioengineer to manually determine a corresponding valve-control. Using the proposed solution, this now can be conducted automatically in negligible runtime.

TABLE II
COMPARISION TO RELATED WORK

Case	*Routability* [6] S?	TS	CPU	*MIS* [6] S?	TS	CPU	*Sequential [6]* S?	TS	CPU	*Prop. Heuristic* S?	TS	CPU
10x10; 5 Samples	✗	26	0.001	✗	45	0.001	✓	82	0.001	✓	51	0.109
10x10; 10 Samples	✗	37	0.033	✗	62	0.033	✓	163	0.001	✓	89	0.131
20x20; 20 Samples	✗	58	0.241	✗	146	0.322	✓	538	0.001	✓	251	0.259
30x30; 30 Samples	✗	83	1.112	✗	232	1.119	✓	1151	0.001	✓	490	0.552
40x40; 40 Samples	✗	106	2.513	✗	302	2.879	✓	1973	0.001	✓	753	1.121
50x50; 50 Samples	✗	131	5.430	✗	423	5.767	✓	3007	0.001	✓	1073	2.290
IVD-1 16x16; 32 Samples	Not applicable									✓	297	0.175
IVD-2 14x14; 40 Samples	Not applicable									✓	290	0.176

VII. CONCLUSION

In this work, we presented an exact and a heuristic solution for determining a sound valve-control allowing all samples to be pushed to their targets. These two methods overcome limitations of the related work and allow bioengineers to automatically determine sound solutions which indeed realize their intended applications. Experimental evaluations confirmed the practicability of the methods by evaluating their performance and comparing them to the related work. Therefore, the determination of valve-sequences for applications including In-Vitro Diagnostics have been considered. Both methods have been made publicly available to bioengineers to use them as well as to researchers to extended and adopted them.

REFERENCES

[1] George M Whitesides. The origins and the future of microfluidics. *Nature*, 442(7101):368–373, 2006.

[2] Luis M Fidalgo and Sebastian J Maerkl. A software-programmable microfluidic device for automated biology. *Journal on Lab on a Chip*, 11(9):1612–1619, 2011.

[3] Jessica Melin and Stephen R Quake. Microfluidic large-scale integration: the evolution of design rules for biological automation. 36:213–231, 2007.

[4] Erik C Jensen, Bharath P Bhat, and Richard A Mathies. A digital microfluidic platform for the automation of quantitative biomolecular assays. *Journal on Lab on a Chip*, 10(6):685–691, 2010.

[5] Chunfeng Liu, Bing Li, Bhargab B Bhattacharya, Krishnendu Chakrabarty, Tsung-Yi Ho, and Ulf Schlichtmann. Testing microfluidic fully programmable valve arrays (fpvas). In *Design, Automation and Test in Europe*, pages 91–96, 2017.

[6] Yi-Siang Su, Tsung-Yi Ho, and Der-Tsai Lee. A routability-driven flow routing algorithm for programmable microfluidic devices. In *Asia and South Pacific Design Automation Conference*, pages 605–610, 2016.

[7] Armin Biere, Marijn J. H. Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*. IOS Press, 2009.

[8] Marc A Unger, Hou-Pu Chou, Todd Thorsen, Axel Scherer, and Stephen R Quake. Monolithic microfabricated valves and pumps by multilayer soft lithography. *Science*, 288(5463):113–116, 2000.

[9] William H Grover, Robin HC Ivester, Erik C Jensen, and Richard A Mathies. Development and multiplexed control of latching pneumatic valves using microfluidic logical structures. *Journal on Lab on a Chip*, 6(5):623–631, 2006.

[10] Paul Pop, Ismail Emre Araci, and Krishnendu Chakrabarty. Continuous-flow biochips: Technology, physical-design methods, and testing. *Journal on Design & Test*, 32(6):8–19, 2015.

[11] Oliver Keszocze, Robert Wille, Tsung-Yi Ho, and Rolf Drechsler. Exact one-pass synthesis of digital microfluidic biochips. In *Design Automation Conference*, pages 1–6, 2014.

[12] Oliver Keszocze, Robert Wille, Krishnendu Chakrabarty, and Rolf Drechsler. A general and exact routing methodology for digital microfluidic biochips. In *Int'l Conf. on Computer-Aided Design*, pages 874–881, 2015.

[13] Andreas Grimmer, Qin Wang, Hailong Yao, Tsung-Yi Ho, and Robert Wille. Close-to-optimal placement and routing for continuous-flow microfluidic biochips. In *Asia and South Pacific Design Automation Conference*, pages 530–535, 2017.

[14] Oliver Keszocze, Zipeng Li, Andreas Grimmer, Robert Wille, Krishnendu Chakrabarty, and Rolf Drechsler. Exact routing for micro-electrode-dot-array digital microfluidic biochips. In *Asia and South Pacific Design Automation Conference*, 2017.

[15] Charles J Alpert, Dinesh P Mehta, and Sachin S Sapatnekar. *Handbook of algorithms for physical design automation*. 2008.

[16] Andrew B Kahng, Jens Lienig, Igor L Markov, and Jin Hu. *VLSI physical design: from graph partitioning to timing closure*. 2011.

[17] Leonardo De Moura and Nikolaj Bjørner. Z3: An Efficient SMT Solver. In *Tools and Algorithms for Construction and Analysis of Systems*, pages 337–340, 2008.

[18] Ameer M. Abdelhadi. Multi-sink lee-moore shortest path maze router. `https://github.com/AmeerAbdelhadi/Multi-Sink-Lee-Moore-Shortest-Path-Maze-Router`, 2011.

[19] Fei Su and Krishnendu Chakrabarty. High-level synthesis of digital microfluidic biochips. *Journal on Emerging Technologies in Computing Systems*, 3(4):1, 2008.