# Channel Routing for Microfluidic Devices: A Comprehensive and Accessible Design Tool

Philipp Ebner*, Gerold Fink* and Robert Wille†‡
*Johannes Kepler University - Institute for Integrated Circuits, Linz, Austria
†Technical University of Munich - Chair for Design Automation, Munich, Germany
‡Software Competence Center Hagenberg GmbH (SCCH), Hagenberg, Austria
{philipp.ebner, gerold.fink}@jku.at, robert.wille@tum.de
https://www.cda.cit.tum.de/research/microfluidics/

***Abstract*—Microfluidics is a technology that enables moving analytic processes from expensive and bulky laboratory equipment to small-scale devices. Microfluidic devices, usually in the form of *Labs-on-a-Chip* (LoCs), have found many great applications in medicine, biology, and chemistry. In particular, LoCs that utilize channels to transport fluids or droplets between different components on the chip are a promising technology. However, the design process of such channel-based LoCs is in need of further automation efforts since the underlying design steps are still rather complex and conducted mainly by hand. An important task in microfluidic design automation is the so-called *channel routing*, where components on LoCs are connected by microfluidic channels. Methods that aim to automate this routing task must factor in the specific demands of microfluidic devices. Common requirements for microfluidic routing layouts are to prevent sharp channel bends and to realize a particular length of channels. Unfortunately, most of the available routing algorithms address these requirements only partly and insufficiently. In this work, we propose a router that is able to overcome these shortcomings and allows automatic channel routing with a minimal bending radius as well as a desired length. In order to make the router accessible to users with little to no design automation expertise, the solution is implemented as an online tool with a user-friendly and intuitive interface. The resulting tool can be accessed at https://www.cda.cit.tum.de/research/microfluidics/channel_router/.**

***Index Terms*—microfluidics, routing, channel-based, matching-length, rubber-band**

## I. Introduction

MICROFLUIDICS is an emerging field that studies the behavior and manipulation of fluids at small scales – typically ranging from micro to picoliters [1]. The resulting devices are able to replace bulky and expensive laboratory equipment by minimizing, integrating, and automating processes on a single chip. Corresponding systems are therefore called *Labs-on-a-Chip* (LoCs), which enable users to perform a broad range of biochemical experiments and have thus found great applications in medicine, biology, and chemistry [2]–[4]. As such, microfluidic devices are ideal prospects for point-of-care assays and similar diagnostics due to their small feature size and their ability to process small volumes of reagents [5].

Given their widespread applicability, the design process for LoCs is still in its infancy – with the majority of the work being done by hand and depending on assumptions, simplifications, as well as the designer's knowledge. The fact that the design process is rather complex frequently leads to error-prone designs and expensive iteration cycles for re-designs, which is a significant disadvantage. In order to provide an alternative to this manual approach, a variety of *Electronic Design Automation* (EDA) approaches and tools have been developed to assist designers with various tasks throughout the design process [6]–[12].

Routing is one of these design tasks, in which components and their respective input and output ports must be connected by channels inside a microfluidic chip while also taking into account several constraints. There are various platforms for microfluidic applications, but most require some sort of routing in their design cycle. For instance, flow-based devices controlled by micro-valves have a two-layer design, which necessitates routing both the control-layer (regulating the valves) and the flow-layer (consisting of the actual channels to transport the samples) [13]–[16]. Other microfluidic systems with a single flow layer, such as droplet-based [17], paper-based [18], or capillary-based [19] devices, have equivalent routing issues[1].

Taking inspiration from approaches developed for the electrical domain is a straightforward way to automate this task. Wire routing is a well-known procedure in EDA, with several methods available, e.g., for *Printed Circuit Boards* (PCBs) [26] and *Integrated Circuits* (ICs) [27]. On the other hand, these approaches usually produce designs with sharp-cornered channel bends (90°), which are – under certain circumstances such as non-laminar flow or high Reynolds numbers – not optimal for fluid transportation because they disturb the flow within the channel [28], [29]. Furthermore, such routers [13], [30], [31] are often designed to determine the shortest possible length of the associated links. In microfluidics, however, the channels are typically subject to additional requirements, such as the length of the channel to provide a specified hydrodynamic resistance, a maximum or minimum flow rate, the amount of time a fluid or droplet must travel through the channel, etc.

Hence, it is essential to have dedicated channel routing solutions for microfluidic devices that explicitly meet these

[1]Besides that, please note that the term "routing" is also frequently used to refer to the path-finding of droplets, e.g., in digital [20]–[23] or programmable [24], [25] microfluidics. Likewise, "channel routing" describes a different process in electronic design automation. However, in this work, we focus on the routing of channels in channel-based microfluidics.

domain-specific requirements. To our knowledge, only the works proposed in [29], [32] contributed (partially) in this direction, by addressing the problems of cornered channel bends and length matching constraints. However, these approaches consider the respective requirements only individually. Addressing *all* requirements simultaneously is a substantially harder task for which, thus far, no automatic solution exists yet.

Moreover, typical routing methods require expertise in certain software or even substantial programming knowledge in order to be used. Therefore, adjusting to specific routing solutions takes a considerable amount of time. Due to the lack of interfaces and a procedure that is still heavily reliant on design automation expertise, these routing algorithms have not gained widespread adoption in the microfluidic community.

In this work[2], we propose a *comprehensive* and *easily accessible* solution to this problem. More precisely, we developed a method that determines the desired routings and, at the same time, satisfies different domain-specific constraints. In addition to typical geometric properties such as width and spacing, the proposed solution guarantees a specified minimum bend radius for all channels. Moreover, it aims to route channels in such a way that they have a certain length. This in turn enables influencing important parameters such as hydrodynamic resistance, flow rates, and pass-through timings.

Afterwards, we incorporated this method into a front-end which is easily accessible and can even be used by end-users from the microfluidic domain without detailed expertise in design automation. The tool is set up in such a way that a routing problem can be specified in a graphical, push-button fashion. Eventually, the resulting design is delivered in a widely used file format and can be easily used for further processing.

The remainder of this work is structured as follows: First, we go into more detail about the considered routing problem in the next section. In Sec. III, we present the general idea proposed in order to solve the problem. Afterwards, we discuss the corresponding implementation and its challenges in Sec. IV. The resulting tool and its applicability are presented in Sec. V, before the paper is concluded in Sec. VI.

## II. Considered Problem

In this section, we describe the considered problem in more detail and discuss the challenges that come along with it. To this end, we use Fig. 1 that provides a corresponding illustration of the problem we want to address.

More precisely, in the considered scenario different components are placed on a 2D-layer, which are supposed to perform different microfluidic operations such as mixing, heating, incubation, etc. Each of these components has one or more inlets as well as outlets, through which fluids/droplets can enter or exit the component. Now, a typical design task is to connect these components as well as input/output ports of the chip in a certain way (based on a corresponding specification), e.g., to guide fluids/droplets through the chip and, by this, realize a desired experiment – constituting a *routing* problem.

[2]A preliminary version of this work has been presented before in [33].

In the domain of conventional electronic circuits, routing is an established process for which numerous methods have been proposed, e.g., for PCBs or ICs. Accordingly, it seems obvious to simply use these methods as a basis for microfluidic devices as well. However, for channel-based microfluidic devices as considered in this work, these schemes do not properly work. While some of the existing solutions are capable of incorporating length constraints and implementing rounded bends on their own, the combination of these constraints, in addition to consideration of parameters such as microfluidic channel resistances, and the lack of simple user interfaces for the intended target group (designers of microfluidic devices), still poses a problem. In particular, the following challenges constitute crucial problems for the current design process of microfluidic devices:

*Rounded Corners:* For some microfluidic applications, it is important to ensure a proper flow of the respective fluids. To this end, corners, i.e., changes in the direction of the channels, can be implemented by sharp, rectangular turns or, alternatively, by smooth, rounded bends – ideally with a certain bending radius. The effects of the shape of channel corners on the overall behavior of a microfluidic device are negligible for applications with a laminar flow regime and a low Reynolds number. However, such effects become substantial in non-laminar, turbulent flows [28]. More precisely, particles such as cells or droplets are subject to additional forces during sudden changes of direction, i.e., sharp channel corners may affect the behavior of particles in microfluidic devices under these circumstances. In the worst case, such devices are rendered inoperable. Methods inspired by conventional routing methods for electronic circuits rely on grid-based schemes and, hence, usually yield routing layouts with rather angular corners (also known as *Manhattan layouts*). However, if the aforementioned conditions apply, such layouts are not optimal for channel-based microfluidic devices, and rounded bends are preferable.

*Length Constraints:* Frequently, channels in a microfluidic device are not only supposed to connect the different components but additionally must satisfy a desired length due to various constraints. Such constraints are, e.g., a specific hydrodynamic resistance, a maximal/minimal flow rate through the channel, a certain time a fluid/droplet needs to pass a channel, etc. But again, approaches inspired by routing methods for conventional ICs do not satisfy this constraint and, hence, frequently determine routes that do not ensure specific lengths, resistances, or similar constraints.

**Example 1.** *Let's assume we have the three components placed on a layout as shown in Fig. 1. Applying any method that is inspired by routing for electronic devices will result in a solution, where the corresponding in- and outlets are connected in an angular and direct fashion as shown in the figure. More precisely, all corners of the channels have angles of* $90^\circ$ *and, hence, most certainly will disrupt the flow of the fluids inside them. Furthermore, while the connected channels are indeed correctly routed, they have more or less arbitrary lengths which can become critical in situations where a certain channel length is desired. Overall, the resulting routing most*

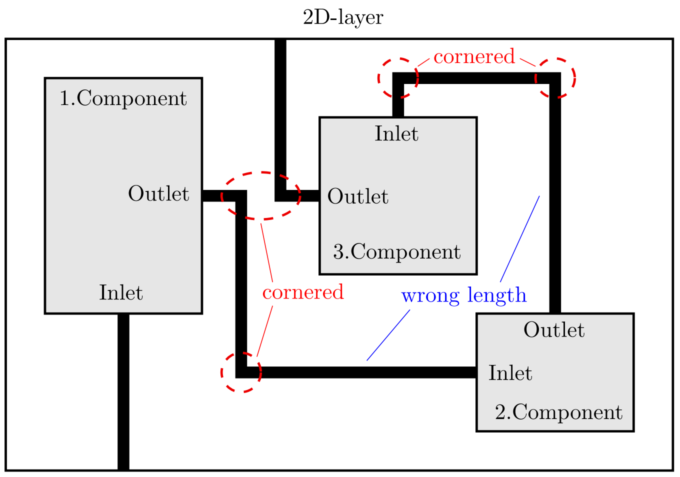


Fig. 1: Routing layout

*likely will not be feasible for microfluidics.*

*Accessibility:* Despite the fact that, thus far, no solution exists which *comprehensibly* addresses the requirements from above, accessibility is another huge problem when it comes to design automation for microfluidics. In fact, the vast majority of the solutions proposed in the past require a substantial programming or EDA-background (notable exceptions are, among others, [6]–[8]). Because of this, many design automation solutions, even if they generate great results, often do not get established in practice.

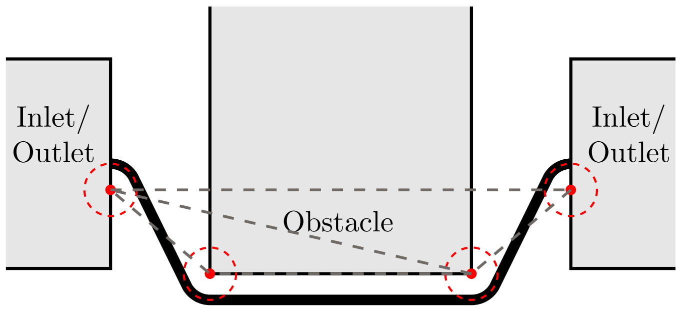
## III. General Ideas

In this section, we present the general ideas behind the proposed channel router, which will overcome the shortcomings discussed above. The main idea is based on a so-called rubber-band router [34]–[36]. As the name suggests, a connected channel between two inlet/outlets is modeled as a rubber-band. This has the effect that, when an obstacle prevents a direct connection, the channel just bends around the obstacle as shown in Fig. 2a. Hence, this router method does *not* rely on common grid-based algorithms and, thus, can produce an “any-angle” layout.

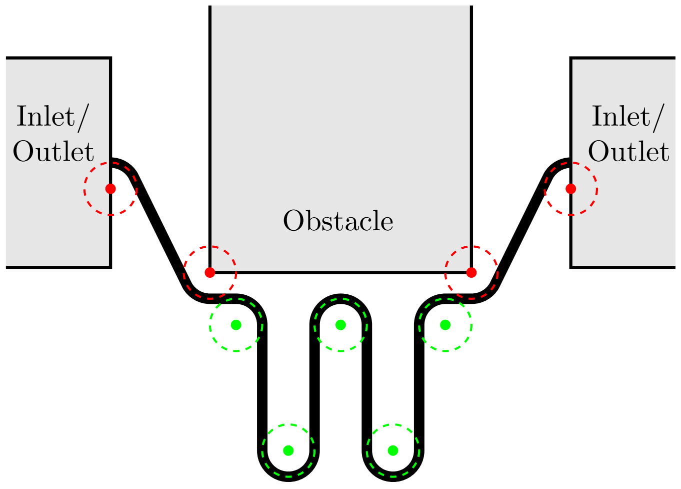
Having that as a main idea, we address the shortcomings discussed before as follows:

*Addressing Rounded Corners:* Critical points in the layout such as inlet/outlets or corners of obstacles are modeled as waypoints (marked as red circles in Fig. 2a). Channels must have a certain distance to these round waypoints and can only pass them by a circular path. This distance can be adjusted by adapting the radius of these waypoints. By this, a desired bending radius of the channel is guaranteed. As a result, a channel will always be a combination of straight segments and arcs when a straight connection is not possible. This finally prevents angular corners of common routing algorithms.

*Addressing Length Constraints:* When a channel should realize a desired length, a common concept in microfluidics is to add a meander structure to the channel. They allow to adjust the length of the channel while, at the same time, do not occupy too much space on the chip. In the proposed



(a) Obstacle affects channel



(b) Additional waypoints for meander structure

Fig. 2: General idea of the proposed rubber-band routing

routing idea, such meander structures can be realized by placing additional waypoints (marked as green circles in Fig. 2b) at convenient spots near the corresponding channel. Afterwards the algorithm forces the channel to “connect” to these waypoints in a proper way. By adjusting the position of these extra waypoints, the desired channel length can be generated.

*Addressing Accessibility:* In order to make the router more accessible, a main idea is to realize the corresponding methods as an online tool, which allows to open it directly inside a browser and prevents further installations processes. Moreover, the interface of the tool should be developed in a sense that also users with little to no programming or EDA-experience can interact with it easily, i.e., the tool should be well suited for microfluidic engineers. To this end, user inputs should either be made by drawing the corresponding components/obstacles/connections inside a drawing area, or by entering the corresponding information through user-friendly input masks. Additionally, instead of defining a desired length of a channel, the user should be able to specify, e.g., a certain hydrodynamic resistance of a channel. This resistance will then be converted automatically to the corresponding length. Once all inputs are defined, a routing layout can then be literally generated in a push-button fashion and exported as *Scalable Vector Graphics* (SVG) file.

Having sketched the main ideas behind the proposed routing method, there are still challenges which need to be addressed, e.g., how to position the waypoints, how to find valid paths, or how to realize a desired channel length. All these challenges will be discussed in more detail in the next section.

## IV. Implementation Details

In this section, we cover the details of the proposed router and its challenges for the implementation. To this end, we first outline the basic algorithmic procedure which is based on an A* search. Afterwards, we describe the corresponding expansion strategy needed to realize the respective connections and the meander creation needed to realize the desired lengths.

### A. Algorithmic Procedure

The basic procedure to determine a valid routing consists of an initialization phase and the actual path finding realizing the desired connections:

*Initialization:* First, waypoints are generated for all corners of all components/obstacles and, for each end of a channel, further waypoints are placed such that the inlets are located on the circumference of that waypoint as shown for the inlets in Fig. 2a. This basically provides the main entities for the routing method, i.e., those waypoints are supposed to be connected until a desired connection from a start to destination inlet is realized. In order to ease this process later, another initialization step is executed: a Delaunay triangulation. In theory, all waypoints could be part of the waypoint sequence that realizes a desired channel connection. However, the relevant successor waypoints in the search process are typically found near the current search position (e.g., in the proximity of obstacles). Therefore, a Delaunay triangulation is conducted in order to significantly reduce the number of possible search nodes. Here, the area spanned by all waypoints is partitioned into triangles where each waypoint is located on the corner of one or more triangles, as shown by the gray dashed lines in Fig. 2a. This allows to storing a list of *neighbors* for each waypoint (namely those sharing a triangulation edge with it), which is used to aid the routing process by reducing the number of waypoints to be checked for the next step to the neighboring ones (rather than all of them). Since the waypoints are typically well distributed, the search space is reduced with this neighbor list.

*Path Finding:* After the initialization, determining the channel connections is conducted through an A* search. This graph search algorithm determines the shortest connection by utilizing a heuristic to estimate the remaining length and favors waypoints that are close to the target. In our model, a node represents the connection up to the current point which is always located on the circumference of a waypoint, i.e., the connection or channel is *attached* to the waypoint. The search starts at a channel inlet, i.e., the position on the circle around the corresponding waypoints as depicted for the inlet in Fig. 2a. In every iteration, the algorithm chooses the most promising node in terms of minimal path length and the estimated remaining connection length. To this end, we simply employ the Euclidean distance as a heuristic for the remaining path length which is a reasonable choice for 2D-problems. The algorithm terminates when either the destination inlet is reached in the same way as the initial node, or all search nodes have been exhausted without finding a connection. The crucial part is the *node expansion*, i.e., to determine successor nodes while, at the same time, not violating any spacing constraints. Since each placement of channel segments may influence other connections and even itself, we have to compute subsequent nodes on-the-fly. This so-called *expansion strategy* determines how a node is expanded, which is explained in the following.

### B. Expansion Strategy

As already mentioned, the current node represents the connection up to the current endpoint, which is always located on the circumference of a waypoint. The goal is now to expand this node by connecting to other waypoints. Theoretically, every waypoint may generate a possible successor node, but neighboring waypoints can be efficiently determined from the triangulation conducted during the initialization.

Likewise, the endpoint of the consecutive node is also positioned on the circumference of another waypoint, such that the connection is a tangent to both of these circles. For simplicity, it is assumed that new channel connections are attached to already populated waypoints on the outside only. Attaching a channel on the *inside* of already attached channels would require a local rerouting and is not covered in this work.

Therefore, node expansion consists of making connections between circles (illustrated in Fig. 3). The radii of these circles are determined by channel width, spacing, and bend radius such that minimal spacing to the waypoint center is accounted for. If the next waypoint already has previously attached channels, this radius does not only depend on the currently routed channel's dimensions, but additionally on the already attached channels. Therefore, the new circle's radius $c_r$ can be computed from the outermost attached circle $a$ and the current connection $c$ with

$$c_r = \max\left(c_{br}, a_r + \max(a_s, c_s) + \frac{1}{2}\left(a_w + c_w\right)\right),$$

where $a_r$ is the outermost radius, $a_w$ is the width, and $a_s$ is the spacing of the outermost circle. Similarly, $c_w$ is the width, $c_s$ is the spacing, and $c_{br}$ is the minimal bend radius of the current connection. If there is no (outermost) circle, i.e., the next waypoint has no attached channels, the same calculation is applied with $a_r = 0$, $a_s = 0$, and $a_w = 0$.

**Example 2.** *Let's assume we want to connect to a waypoint that already has one attached channel* $a$ *with* $a_r = 100\,\mu\mathrm{m}$, $a_s = 50\,\mu\mathrm{m}$, *and* $a_w = 20\,\mu\mathrm{m}$ *(as illustrated in Fig. 4). If the connecting channel* $c$ *has the geometric properties* $c_{br} = 150\,\mu\mathrm{m}$, $c_s = 60\,\mu\mathrm{m}$, *and* $c_w = 20\,\mu\mathrm{m}$, *then the computed radius amounts to* $c_r = 180\,\mu\mathrm{m}$.

Then, the tangents to the previous and next waypoint circle are computed. In general, there are up to two possible tangents that match the clockwise direction of the previously generated channel path. Each of these segments may establish a new node, as depicted in Fig. 3.

Therefore, a single node expansion is reduced to simple geometric operations by adding an arc and a straight channel

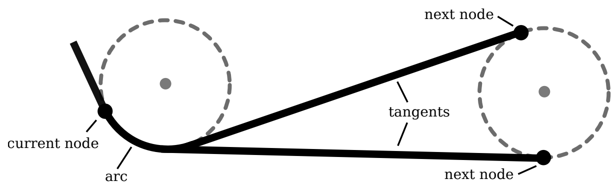


Fig. 3: Node expansion

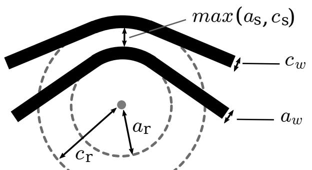


Fig. 4: Radii of attached channels

segment to the current path, both of which need to be validated against geometrical constraints such as minimal distances to obstacles. More precisely, straight channel segments may violate geometrical constraints in essentially three different ways:

- *Crossing other Channel Segments:* If it crosses the preceding tangent as illustrated in Fig. 5a, i.e., it self-intersects the previous segment, attempt to directly connect the waypoints instead. This is necessary to guarantee a connection between waypoints that are not neighbors by triangulation. Otherwise, if it crosses other channel segments (cf. Fig. 5b), it is discarded.
- *Too Close to other Waypoint:* If a third waypoint intercepts the segment, as depicted in Fig. 5c, attempt to route to that specific waypoint instead.
- *Crossing Obstacles:* Discard the segment if it crosses an obstacle (cf. Fig. 5d).

Arcs around waypoints need to be validated in a similar fashion, whereas the new node is always discarded in case of conflicts. Arcs can violate constraints in the following four ways:

- *Obstacle Crossing:* If an arc directly crosses an obstacle, as shown in Fig. 6a.
- *Close to Obstacles:* When an arc is too close to an obstacle, and thus spacing constraints are not satisfied (cf. Fig. 6b).
- *Close to Channels:* If an arc is too close to other channel segments, then the spacing constraints of both, arc and channel, are not satisfied, as shown in Fig. 6c.
- *Close to Waypoints:* If other waypoints (and their attached channels) are too close to the arc so that mutual spacing constraints are violated (cf. Fig. 6d).

All expansions that pass these checks are possible successor nodes. Following the A* scheme, the one is further explored which yields the lowest cost until the shortest length solution is obtained for the connection. However, as we might not necessarily be interested in a connection with the lowest distance but a precisely given one, we additionally add meanders to achieve the desired length. This is covered in the next section.

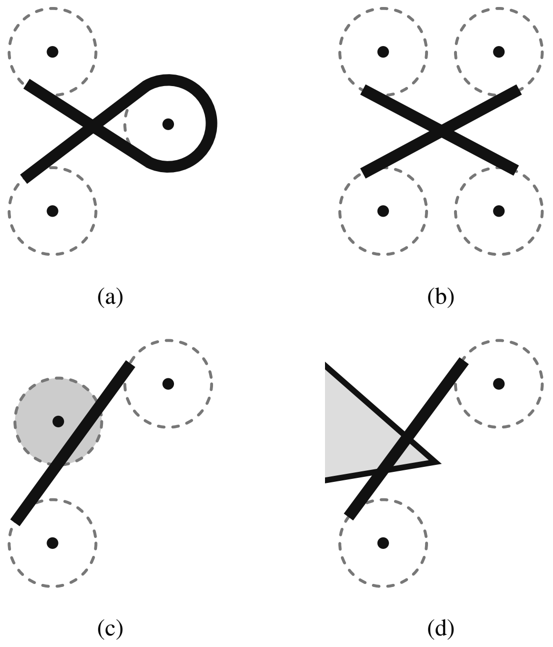


Fig. 5: Channel segment constraint violations

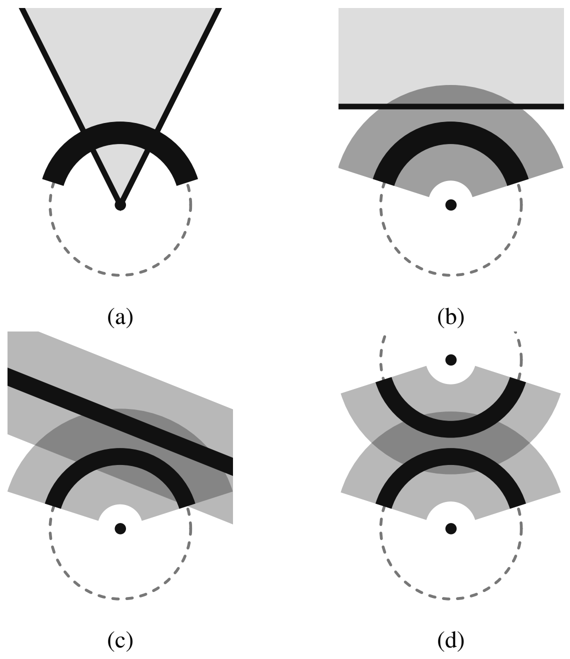


Fig. 6: Waypoint arc constraint violations

## C. Creation of Meanders

In order to match the desired length of specific channels, the algorithm tries to incorporate meander structures into the initial routing. We assume that the desired length is longer than the length of the initial routing[3], so we aim to extend it until the length criterion is met. To this end, we employ a two-step process: First, we iteratively insert meander bends until the channel has at least the desired length (but may be

[3]Please note that, if a channel is already longer than its desired length, we cannot find a solution. This problem is also discussed later in Sec. V-C.

longer), i.e., an *upper bound* routing is computed, which is a simpler task than directly determining an exact length routing. Afterwards, we use this result to determine a routing that matches the desired length (employing a bisection approach). Before this process is explained in detail, the general process of creating meander bends is examined. This is achieved by inserting special meander waypoints at suited positions along the straight segments of a channel.

**Meander waypoint placement:** Meander waypoints differ from regular waypoints in the intended effect on the overall design. On one hand, only single channels form a meander, and therefore, meander waypoints have exactly one attached channel in a valid routing. On the other hand, meander waypoints do not have to apply the same spacing constraints to attached channels. While regular waypoints have the purpose of ensuring a certain distance between the attached channel and an obstacle, i.e., the obstacle corner located at the waypoint's position, there is no obstacle edge at meander waypoints. Hence, the required spacing is halved and the radius $c_r$ around the waypoint can be computed with

$$c_r = \max\left(c_{br}, \frac{1}{2}\left(c_s + c_w\right)\right),$$

where $c_w$ is the width, $c_s$ is the spacing, and $c_{br}$ is the minimal bend radius of the channel.

Having that, the next task is to place these waypoints in order to enable routing channels into the typical meander shape. The goal is to replace each straight channel segment with a corresponding meander. In the following, we describe the procedure to insert a meander instead of a single channel segment.

At each end of the meander, the equivalent space of $2c_r$ is reserved for *entry waypoints* (indicated in blue in Fig. 7) that redirect the channel sideways. The remaining space is filled with meander waypoints (marked as green circles in Fig. 7), whereupon the number of meander bends $n$ (and, therefore, the number of necessary waypoints) can be computed as

$$n = \left\lfloor \frac{l - 4c_r}{2c_r} \right\rfloor,$$

where $l$ is the length of the original channel segment that is to be replaced with the meander. In order to achieve a more regularly shaped meander, the meander waypoints are equally distributed along the original segment's direction. The resulting distance $d$ between meander waypoints therefore amounts to

$$d = \frac{l - 4c_r}{n}.$$

After resolving the waypoint distribution along the original channel segment, the waypoints are displaced by a certain offset – alternating between the left and right side of the original channel segment. This lateral offset of meander waypoints (illustrated in Fig. 7) can be chosen arbitrarily[4]. By altering the lateral offset for single meander waypoints, the length of the resulting meander and, therefore, the length of the entire channel, can be adjusted to the respectively given requirement. However, a large offset may render it impossible for the setup to produce a valid routing, e.g., when a meander waypoint is moved too close to an obstacle or another channel.

[4]When the lateral offset is set to $-c_r$, then the corresponding waypoint vanishes, i.e., producing the same effect as having no meander waypoint at the designated spot.

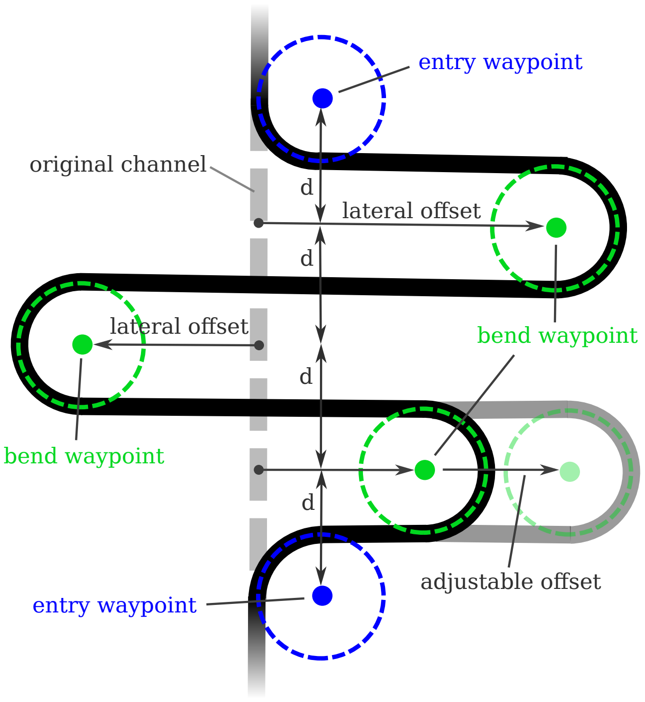


Fig. 7: Meander waypoint placement

With the waypoint placement taken care of, the remaining task is to determine the lateral offsets of meander waypoints such that a routing with the desired length is generated. This is achieved in a two-step process.

**Upper bound routing:** The first part consists of step-by-step increments of the meander waypoint offsets until an upper bound routing is found, i.e., a routing where the channel has *at least* the desired length. Overall, this leads to a meander insertion procedure as sketched in Fig. 8. The meander generation starts with an initial routing with no lateral offsets. As long as the *actual* channel length is smaller than the *desired* channel length (the condition marked as C1), we attempt to further increase lateral offsets of meander waypoints and, therefore, channel length.

To this end, the next available meander waypoint is chosen. If the waypoint offset has not been marked as *maxed out* (the condition marked as C2) (i.e., at its maximal possible offset since previous iterations showed that the lateral offset could not be further increased due to the presence of obstacles), we increase its lateral offset by a certain predefined step size. The choice of step size is arbitrary, however, in this work, a step size of $c_r$ produces reasonable results due to the nature of the problem. In any case, the length of the resulting channel increases with each step. Accordingly, the routing is updated after each change of the meander waypoint offsets.

If the result of the updated routing is valid (the condition marked as C3), the result is stored as the current intermediate result. Otherwise, if such a routing turns out to be impossible, we can assume that the altered meander offset produces a

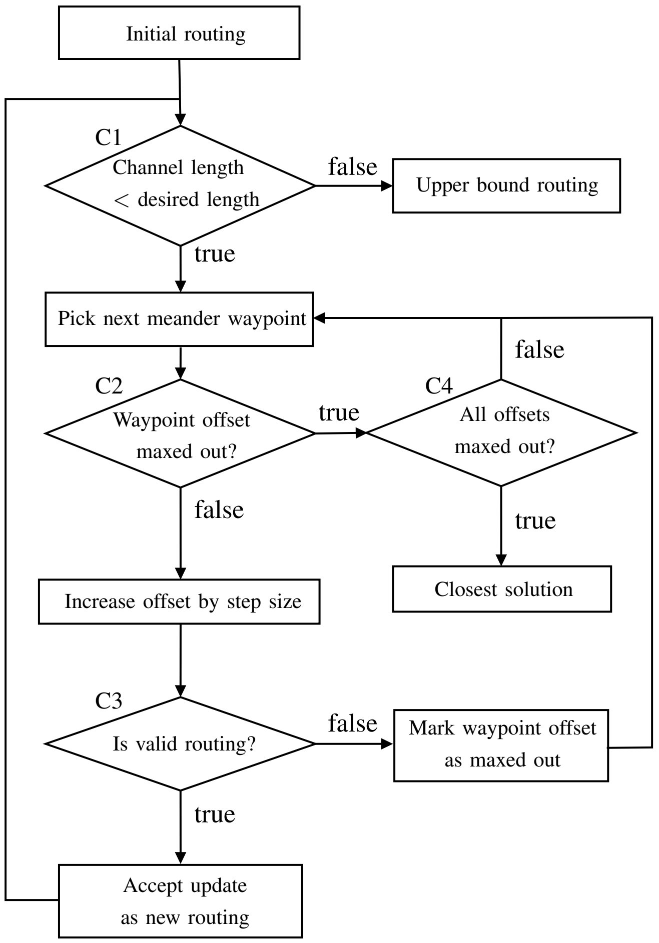

Initial routing
C1
Channel length < desired length
false
Upper bound routing
true
Pick next meander waypoint
C2
Waypoint offset maxed out?
true
C4
All offsets maxed out?
false
true
Closest solution
false
Increase offset by step size
C3
Is valid routing?
false
Mark waypoint offset as maxed out
true
Accept update as new routing

Fig. 8: Meander insertion

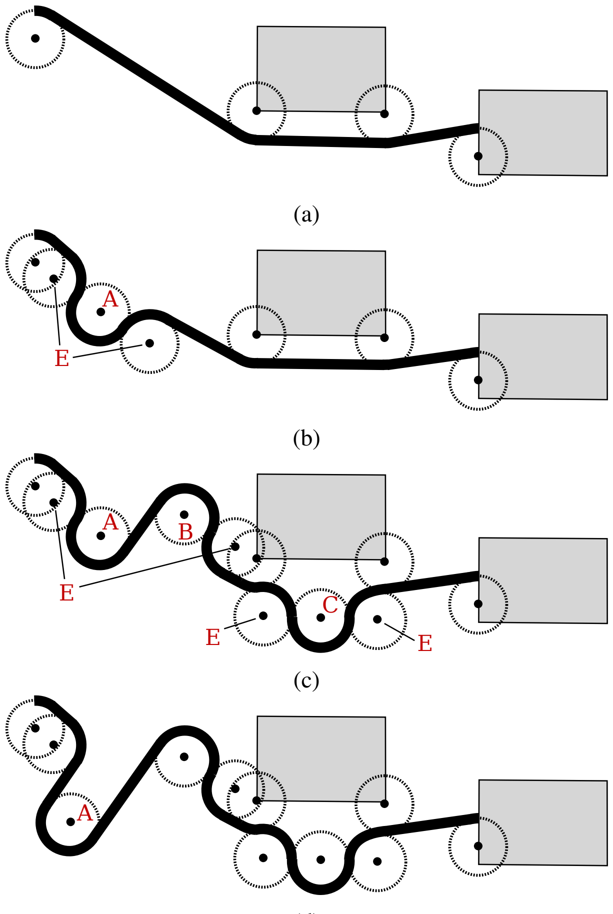


Fig. 9: Illustration of the meander waypoint insertion

collision with other obstacles, and, therefore, is marked as maxed out. In both cases, we move on to the next meander waypoint and repeat the process.

This procedure is repeated until the desired channel length is reached or surpassed. If all meander waypoints are at their maximal offset (the condition marked as C4), i.e., maxed out, without reaching the desired channel length, it is impossible to find a desired solution under the assumptions of the applied model. In this case, it is up to the designer to make changes to the problem specification (this is discussed in more detail later in Sec. V-C). However, the latest intermediate routing result constitutes the closest solution that has been found.

To illustrate the core idea of this procedure, Ex. 3 and Fig. 9 sketch the progression over time of the described method for a small example[5].

**Example 3.** *Let's assume we have an initial routing of a channel and intend to increase its length by the described process, a sketch of which is shown in Fig. 9. The initial state is depicted in Fig. 9a: Since all meander waypoints vanish for the initial lateral offsets, no additional waypoints are created. In the next step (Fig. 9b), a first meander waypoint A is inserted (and accordingly, two entry waypoints E) in place of the first straight segment. Several steps later (Fig. 9c), every line segment has been replaced with the maximum number of meander waypoints A, B, and C (the last segment is too short to fit any meander bends). Thus, in Fig. 9d, we continue by further displacing the leftmost meander waypoint A from the original line. This process could be continued until the desired length is reached or no further displacement is possible.*

[5]The locations of the meander waypoints are determined with the formulas for the meander waypoint placements. The upper bound routing algorithm merely alters the introduced lateral offsets.

**Bisection:** The first step of the two-step process results in a routing that does not necessarily have the exact desired length, but may be larger. As the second step, a routing with the exact desired target length can be generated by applying the bisection method [37], which is illustrated in Fig. 10 and described in the following.

At this point, our current result, generated by the first step (Fig. 8), has a larger channel length then the target, implementing an upper bound. Because waypoint offsets have been increased by a constant step size, we know that the length of the previous, intermediate solution is strictly smaller than the target length, which constitutes a lower bound. We use this fact to determine the ideal solution in between these bounds. Therefore, the bisection acts on the offset of a single meander waypoint (as indicated in Fig. 7 with the *adjustable offset*), such that the ideal offset lies somewhere in the interval

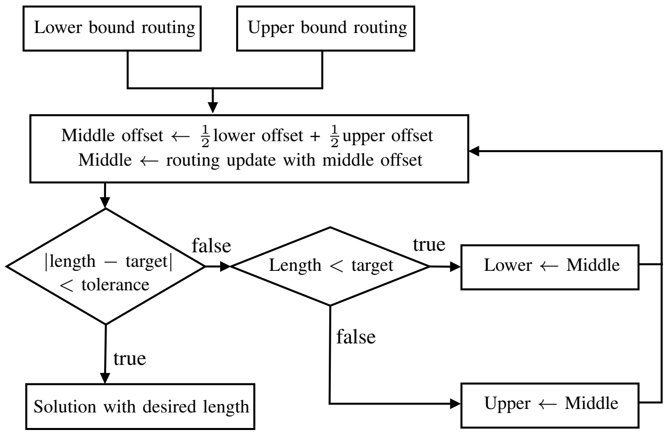

Lower bound routing
Upper bound routing
Middle offset ← $\frac{1}{2}$lower offset + $\frac{1}{2}$upper offset
Middle ← routing update with middle offset
\|length − target\| < tolerance
false
Length < target
true
Lower ← Middle
false
true
Solution with desired length
Upper ← Middle

Fig. 10: Bisection

between the lower and upper bound routing.

In each iteration, the possible interval of the waypoint offset is halved. The routing is updated with a waypoint offset set to the middle of the interval. Then, the resulting channel length of the routing is analyzed:

- If the length is equal to the target with respect to a certain tolerance, we have found our desired solution.
- Otherwise, if the length is smaller than the target, the computed routing constitutes the lower bound for the next iteration.
- Otherwise, the computed routing constitutes the upper bound for the next iteration since the length is larger than the target.

By applying this scheme, the routing converges towards a solution with the desired length.

Eventually, carrying out this procedure for all channels generates a valid routing, such that the channels conform to the specified bend radius as well as the desired length. The results are then visualized in a graphical user interface, which is introduced in the next section.

## V. Resulting Tool

All ideas and all implementations described above have been implemented in terms of an online tool, which can be accessed by visiting https://www.cda.cit.tum.de/research/microfluidics/channel_router/. The tool particularly aims at microfluidic engineers looking for an efficient and easy-to-use solution to realize the channel routing in their devices. In this section, we demonstrate how to use the tool and what results can be generated with it.

### A. Using the Tool

The tool itself is shown in Fig. 11, which basically consists of a drawing and control area on the left- and right-hand side, respectively. The drawing area is the actual workspace of the tool and holds a visual representation of the current layout, while the control area contains buttons to start or clear the routing as well as an input mask to have a more detailed control over the specified layout. By clicking on the corresponding buttons inside the control area, the user can add the following two objects to the drawing area:

*Components* represent the actual building blocks on a microfluidic chip and must not be crossed by a channel, i.e., they also serve as obstacles inside the layout. The tool allows to define these components in the form of $n$-gons (i.e., polygons with $n$ sides). This has the advantage that both simple but also rather complex objects can be created. To this end, the polygons can be directly drawn inside the drawing area as shown in Fig. 11: Here, the top-left component is currently selected and, hence, its corners can be freely dragged around in order to represent the desired shape (the middle point can be used to drag the whole component around). Additionally, components can also be specified as a list of points and their corresponding $x$ and $y$ positions in order to allow for a more precise control over the exact shape and position.

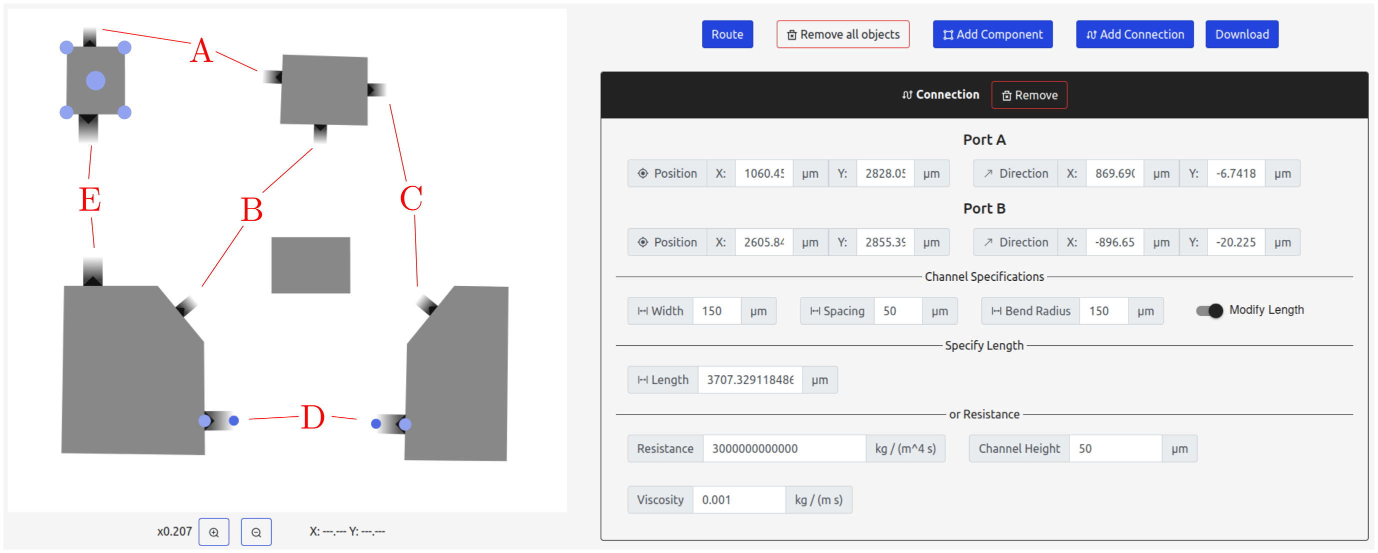
*Connections* represent the inlets/outlets of components that should be connected inside a microfluidic chip. Hence, a single connection is basically a pair of two points that define the start and end point of a channel. A connection can also be easily added by directly drawing the start and end point (i.e., the inlets/outlets) on the boundary of corresponding components inside the drawing area, as shown in Fig. 11. Usually, the directions of a channel at the start and end point are perpendicular to the edge of the corresponding component, but the direction can also be defined manually by adjusting the second (smaller and darker) point near the start and end point of the connection. Moreover, when a new connection is added, the user is prompted to provide the desired channel width, the minimal space between two channels, the minimal bending radius, as well as an (optional) length of the channel (as shown on the right-hand side of Fig. 11). Note that, alternatively to the channel length, the hydrodynamic resistance of the channel can also be provided. However, in this case also the height of the channel, and the viscosity of the fluid have to be specified. The tool then automatically calculates the correspondingly needed length based on these parameters. Again, all these parameters as well as the coordinates and the start and end points can also be specified directly inside an input mask for a more precise control over the connection.

Once all these objects are defined, the user can start the routing process by pressing the “Route” button.

### B. Results of the Tool

In order to illustrate results generated by the tool, assume the user has entered a layout as illustrated in Fig. 11. Additionally, assume that the user specified the connections A, B, C to have a channel width of $w = 100\,\mu\text{m}$, a minimal bending radius of $r = 100\,\mu\text{m}$, and a minimal spacing of $s = 50\,\mu\text{m}$, while the values for the connections D and E have been specified as $w = 150\,\mu\text{m}$, $r = 150\,\mu\text{m}$, and $s = 50\,\mu\text{m}$. Moreover, the user specified that the connection B should have a particular length of $l = 5000\,\mu\text{m}$, while the connection D should realize a hydrodynamic resistance $R_D = 3 \times 10^{12}\,\text{kg}/(\text{m}^4\text{s})$ (with a channel height of $h = 50\,\mu\text{m}$ and a fluid viscosity of $\mu = 1 \times 10^{-3}\,\text{kg}/(\text{ms})$).

Then, after clicking on the “Route” button, the proposed method instantly generates a layout and displays it to the



Port A

Position X: 1060.45 µm Y: 2828.05 µm | Direction X: 869.690 µm Y: -6.7418 µm

Port B

Position X: 2605.84 µm Y: 2855.39 µm | Direction X: -896.65 µm Y: -20.225 µm

Channel Specifications

Width 150 µm | Spacing 50 µm | Bend Radius 150 µm | Modify Length

Specify Length

Length 3707.32911848€ µm

or Resistance

Resistance 3000000000000 kg / (m^4 s) | Channel Height 50 µm

Viscosity 0.001 kg / (m s)

Fig. 11: Resulting online tool with drawing area (left) and control area (right)

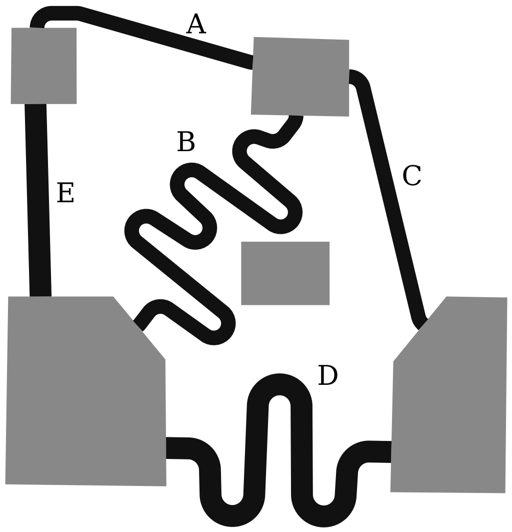
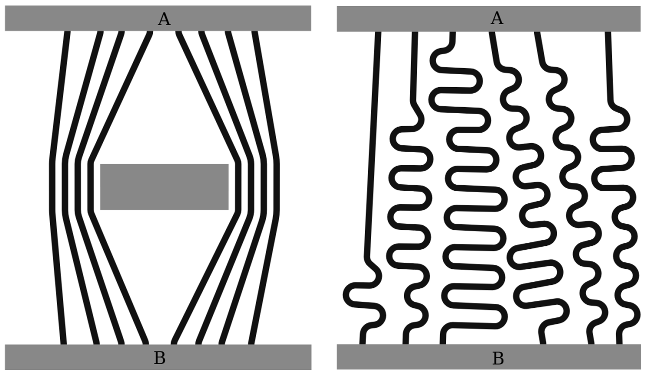


Fig. 12: Generated routing layout



(a) Multiple parallel channels (b) Length-constrained routing

Fig. 13

user as shown in Fig. 12. As can be seen, all channels were routed correctly with respect to the specified parameters and no cornered angles where produced. Moreover, the router automatically generated corresponding meander structures inside the channels B and D in order to match the defined length and hydrodynamic resistance, respectively. In case of the connection B, the router even placed the meander in such a way, that the channel does not cross the obstacle in the middle of the layout. Having that, the user can now download a corresponding SVG file in order to use for further production processes.

Another frequent problem in routing is determining multiple, mostly parallel, connections between components [38]. The concept of rubber-band routing is able to handle such cases in the same way since multiple channels may be attached to the same waypoints. Fig. 13a illustrates a solution for such a case (generated by the proposed tool), where several channels connect the components $A$ and $B$. At the same time, the depicted channels bend around an obstacle and therefore around its cornering waypoints. All of these channels have been specified with the same dimensions, such that each channel has a channel width of $w = 100\,\mu\text{m}$, minimal bending radius of $r = 100\,\mu\text{m}$, and a minimal spacing of $s = 100\,\mu\text{m}$. The channel length is not constrained in this example.

Furthermore, such problems may require each parallel channel to have a certain specified length, e.g., when the throughput time of fluids in each channel must be coordinated (or be equally long in a special case), creating a need for a length-constrained routing between components $A$ and $B$. Fig. 13b depicts such an example (again, generated with the proposed tool) with different channel lengths. Once again, all of these channels have been specified with the same dimensions, such that each channel has a channel width of $w = 100\,\mu\text{m}$, minimal bending radius of $r = 150\,\mu\text{m}$, and a minimal spacing of $s = 100\,\mu\text{m}$. However, the desired length for each channel is arbitrarily defined in advance – amounting to (left-to-right) $6000\,\mu\text{m}$, $9000\,\mu\text{m}$, $15\,000\,\mu\text{m}$, $10\,000\,\mu\text{m}$, $7500\,\mu\text{m}$, and $8000\,\mu\text{m}$. As expected, the channels

TABLE I: Benchmark data

| Case | Components | Channel Connections | | Runtime |
|---|---|---|---|---|
| | | any length | length-constr. | $[s]$ |
| 1 | 20 | 10 | 0 | 0.54 |
| 2 | 20 | 7 | 3 | 0.96 |
| 3 | 20 | 5 | 5 | 1.63 |
| 4 | 20 | 0 | 10 | 3.84 |
| 5 | 40 | 20 | 0 | 1.57 |
| 6 | 40 | 15 | 5 | 3.44 |
| 7 | 40 | 10 | 10 | 6.37 |
| 8 | 40 | 0 | 20 | 16.61 |
| 9 | 100 | 50 | 0 | 17.74 |
| 10 | 100 | 37 | 13 | 30.85 |
| 11 | 100 | 25 | 25 | 52.68 |
| 12 | 100 | 0 | 50 | 111.94 |

are routed in parallel with incorporated meander structures, but without violating mutual spacing constraints. Overall, these examples demonstrate the applicability of the proposed tool. Since the tool is publicly available at https://www.cda.cit.tum.de/research/microfluidics/channel_router/, the reader is kindly invited to try out further examples and scenarios.

### C. Discussion

While the two subsections above provided an intuition about the usage of and the results from the tool, we aim to complete this section with a discussion on the overall performance and applicability of the proposed method and resulting tool (covering what the method/tool is able to deliver, but also what it cannot provide yet).

We provide a set of larger benchmark cases which are also available for download from within our tool. As benchmark cases, we considered connections between arrays of components, both unconstrained and constrained in channel length, as well as mixed cases. For the cases with length-constrained channels, the lengths of these channels were increased by 20% with respect to the unconstrained cases (and therefore, the shortest connections). All other relevant geometric parameters of the channels (width, spacing, and minimum bend radius) were set to sensible fixed values. The results of our performance evaluations are summarized in Table I.

Despite the good performance of the tool, it might happen that the algorithm cannot determine a solution at all. In fact, after all, the method is not complete, i.e., can eventually not prove whether a solution exists. While this is one of the reasons for the efficient run-time performance (being complete would require a much larger search space traverse), this might look like a serious disadvantage at the first glance. However, if the method returns with no result, it is usually sufficient to re-arrange the components slightly differently and re-run the tool. This is especially true when a connection should realize a certain length but the distance between the corresponding components is too long, so not even a straight channel would satisfy the length constraint. In the current version, such connections are marked as red (indicating an error), but we will provide a more comprehensive error description in future implementations in order to prevent these failures. After all, we were always able to eventually determine a result after such small re-arrangements. Since this additionally, did not harm the design task or the validity of the solution, the proposed tool remained efficient. In fact, complex routing layouts could be generated for various designs within a few moments and in an easy-to-use fashion.

## VI. Conclusion

In this work, we presented a routing tool for channel-based microfluidic devices, i.e., devices where components must be connected by channels according to a certain specification. Since current routing algorithms are frequently based on wire routers from the traditional electrical domain, they often do not satisfy the requirements needed in microfluidics, such as a minimal bending radius for channel bends or a length constraint for particular channels. Moreover, the accessibility of these routers is focused on users with an EDA-background, but are mostly not suitable for designers of microfluidic devices. We developed a router that addresses these shortcomings and, additionally, implemented the router as a user-friendly tool. The tool is available online and can be accessed by visiting https://www.cda.cit.tum.de/research/microfluidics/channel_router/.

## Acknowledgments

This work has partially been supported by the FFG project AUTOMATE (project number: 890068) as well as by BMK, BMDW, and the State of Upper Austria in the frame of the COMET Programme managed by FFG.

## References

[1] G. M. Whitesides, “The origins and the future of microfluidics,” *Nature*, vol. 442, no. 7101, pp. 368–373, 2006.

[2] D. Mark, S. Haeberle, G. Roth, F. von Stetten, and R. Zengerle, “Microfluidic Lab-on-a-Chip platforms: requirements, characteristics and applications,” *Chemical Society Reviews*, vol. 39, no. 3, pp. 1153–1182, 2010.

[3] P. S. Dittrich and A. Manz, “Lab-on-a-chip: microfluidics in drug discovery,” *Nature Reviews Drug Discovery*, vol. 5, no. 3, p. 210, 2006.

[4] S.-Y. Teh, R. Lin, L.-H. Hung, and A. P. Lee, “Droplet microfluidics,” *Lab on a Chip*, vol. 8, pp. 198–220, 2008.

[5] P. K. Sorger, “Microfluidics closes in on point-of-care assays,” *Nature biotechnology*, vol. 26, no. 12, pp. 1345–1346, 2008.

[6] T.-M. Tseng, M. Li, D. N. Freitas, T. McAuley, B. Li, T.-Y. Ho, I. E. Araci, and U. Schlichtmann, “Columba 2.0: A co-layout synthesis tool for continuous-flow microfluidic biochips,” *Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 8, pp. 1588–1601, 2017.

[7] X. Huang, T.-Y. Ho, W. Guo, B. Li, and U. Schlichtmann, “Minicontrol: Synthesis of continuous-flow microfluidics with strictly constrained control ports,” in *Design Automation Conference*. IEEE, 2019, pp. 1–6.

[8] A. Grimmer, P. Frank, P. Ebner, S. Häfner, A. Richter, and R. Wille, “Meander designer: Automatically generating meander channel designs,” *Micromachines – Journal of Micro/Nano Sciences, Devices and Applications*, vol. 9, no. 12, 2018.

[9] A. Grimmer, W. Haselmayr, and R. Wille, “Automated dimensioning of Networked Labs-on-Chip,” *Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 2018.

[10] ——, “Automatic droplet sequence generation for microfluidic networks with passive droplet routing,” *Comput.-Aided Des. Integr. Circuits Syst.*, 2018.

[11] G. Fink, M. Hamidović, W. Haselmayr, and R. Wille, “Automatic design of droplet-based microfluidic ring networks,” *Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 2020.

[12] G. Fink, T. Mitteramskogler, M. A. Hintermüller, B. Jakoby, and R. Wille, "Automatic design of microfluidic gradient generators," *IEEE Access*, vol. 10, pp. 28 155–28 164, 2022.
[13] C.-X. Lin, C.-H. Liu, I.-C. Chen, D. Lee, and T.-Y. Ho, "An efficient bi-criteria flow channel routing algorithm for flow-based microfluidic biochips," in *Design Automation Conference*, 2014, pp. 1–6.
[14] K. Hu, T. Dinh, T. Y. Ho, and K. Chakrabarty, "Control-layer routing and control-pin minimization for flow-based microfluidic biochips," *Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. PP, no. 99, 2016.
[15] Q. Wang, H. Zou, H. Yao, T.-Y. Ho, R. Wille, and Y. Cai, "Physical co-design of flow and control layers for flow-based microfluidic biochips," *Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 6, pp. 1157–1170, 2017.
[16] Y. Zhu, B. Li, T.-Y. Ho, Q. Wang, H. Yao, R. Wille, and U. Schlichtmann, "Multi-channel and fault-tolerant control multiplexing for flow-based microfluidic biochips," in *Int'l Conf. on Computer-Aided Design*, 2018.
[17] X. Chen and C. L. Ren, "A microfluidic chip integrated with droplet generation, pairing, trapping, merging, mixing and releasing," *RSC Advances*, vol. 7, no. 27, pp. 16 738–16 750, 2017.
[18] X. Li, D. R. Ballerini, and W. Shen, "A perspective on paper-based microfluidics: Current status and future trends," *Biomicrofluidics*, vol. 6, no. 1, p. 011301, 2012.
[19] J. Guerrero, Y.-W. Chang, A. A. Fragkopoulos, and A. Fernandez-Nieves, "Capillary-based microfluidics—coflow, flow-focusing, electro-coflow, drops, jets, and instabilities," *Small*, vol. 16, no. 9, p. 1904344, 2020.
[20] P.-H. Yuh, C.-L. Yang, and Y.-W. Chang, "BioRoute: A network-flow-based routing algorithm for the synthesis of digital microfluidic biochips," *Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 11, pp. 1928–1941, 2008.
[21] O. Keszocze, R. Wille, and R. Drechsler, "Exact routing for digital microfluidic biochips with temporary blockages," in *Int'l Conf. on Computer-Aided Design*, 2014, pp. 405–410.
[22] O. Keszocze, R. Wille, K. Chakrabarty, and R. Drechsler, "A general and exact routing methodology for digital microfluidic biochips," in *Int'l Conf. on Computer-Aided Design*, 2015, pp. 874–881.
[23] O. Keszocze, Z. Li, A. Grimmer, R. Wille, K. Chakrabarty, and R. Drechsler, "Exact routing for micro-electrode-dot-array digital microfluidic biochips," in *Asia and South Pacific Design Automation Conference*, 2017.
[24] Y.-S. Su, T.-Y. Ho, and D.-T. Lee, "A routability-driven flow routing algorithm for programmable microfluidic devices," in *Asia and South Pacific Design Automation Conference*. IEEE, 2016, pp. 605–610.
[25] A. Grimmer, B. Klepic, T.-Y. Ho, and R. Wille, "Sound valve-control for programmable microfluidic devices," in *Asia and South Pacific Design Automation Conference*, 2018.
[26] A. Finch, K. Mackenzie, G. Balsdon, and G. Symonds, "A method for gridless routing of printed circuit boards," in *Design Automation Conference*. IEEE, 1985, pp. 509–515.
[27] C. Ababei, Y. Feng, B. Goplen, H. Mogal, T. Zhang, K. Bazargan, and S. Sapatnekar, "Placement and routing in 3d integrated circuits," *IEEE Design & Test of Computers*, vol. 22, no. 6, pp. 520–531, 2005.
[28] J. You, L. Flores, M. Packirisamy, and I. Stiharu, "Modeling the effect of channel bends on microfluidic flow," *IASME Trans*, vol. 1, no. 1, pp. 144–151, 2005.
[29] K. Yang, H. Yao, T.-Y. Ho, K. Xin, and Y. Cai, "Aarf: any-angle routing for flow-based microfluidic biochips," *Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 12, pp. 3042–3055, 2018.
[30] A. Grimmer, Q. Wang, H. Yao, T.-Y. Ho, and R. Wille, "Close-to-optimal placement and routing for continuous-flow microfluidic biochips," in *Asia and South Pacific Design Automation Conference*, 2017, pp. 530–535.
[31] S.-S. Chen, J.-J. Chen, C.-C. Tsai, and S.-J. Chen, "Automatic router for the pin grid array package," *IEE Proceedings-Computers and Digital Techniques*, vol. 146, no. 6, pp. 275–281, 1999.
[32] H. Yao, T.-Y. Ho, and Y. Cai, "PACOR: practical control-layer routing flow with length-matching constraint for flow-based microfluidic biochips," in *Design Automation Conference*, 2015, p. 142.
[33] G. Fink, P. Ebner, and R. Wille, "Comprehensive and accessible channel routing for microfluidic devices," in *Design, Automation and Test in Europe (DATE)*, 2022.
[34] D. Staepelaere, J. Jue, T. Dayan, and W.-M. Dai, "Surf: Rubber-band routing system for multichip modules," *IEEE Design & Test of Computers*, vol. 10, no. 4, pp. 18–26, 1993.
[35] T. Dayan, *Rubber-band based topological router*. University of California, Santa Cruz, 1997.
[36] W. W.-M. Dai, T. Dayan, and D. Staepelaere, "Topological routing in surf: Generating a rubber-band sketch," in *Proceedings of the 28th ACM/IEEE Design Automation Conference*, 1991, pp. 39–44.
[37] R. L. Burden and J. D. Faires, "Numerical analysis 8th ed," *Thomson Brooks/Cole*, 2005.
[38] T. Yan and M. D. Wong, "Bsg-route: A length-matching router for general topology," in *2008 IEEE/ACM International Conference on Computer-Aided Design*. IEEE, 2008, pp. 499–505.

**Philipp Ebner** received his Master's degree in computer science from the Johannes Kepler University Linz, Austria, in 2021. Currently, he is a Ph.D. student at the Institute for Integrated Circuits at the Johannes Kepler University. His main research interest is design automation for microfluidics.

**Gerold Fink** received the Master's degree in mechatronics from the Johannes Kepler University Linz, Austria, in 2019. Currently, he is a Ph.D. student at the Institute for Integrated Circuits at the Johannes Kepler University. His research area focuses on simulations and design automations for microfluidic networks.

**Robert Wille** (M'06–SM'15) is a Full and Distinguished Professor at the Technical University of Munich, Germany, and Chief Scientific Officer at the Software Competence Center Hagenberg, Austria. He received the Diploma and Dr.-Ing. degrees in Computer Science from the University of Bremen, Germany, in 2006 and 2009, respectively. Since then, he worked at the University of Bremen, the German Research Center for Artificial Intelligence (DFKI), the University of Applied Science of Bremen, the University of Potsdam, and the Technical University Dresden. From 2015 until 2022, he was Full Professor at the Johannes Kepler University Linz, Austria, until he moved to Munich. His research interests are in the design of circuits and systems for both conventional and emerging technologies. In these areas, he published more than 350 papers and served in editorial boards as well as program committees of numerous journals/conferences such as TCAD, ASP-DAC, DAC, DATE, and ICCAD. For his research, he was awarded, e.g., with Best Paper Awards, e.g., at TCAD and ICCAD, an ERC Consolidator Grant, a Distinguished and a Lighthouse Professor appointment, a Google Research Award, and more.