Improving the Robustness of Microfluidic Networks

Gerold Fink*, Philipp Ebner*, Sudip Poddar*, and Robert Wille*‡
*Johannes Kepler University - Institute for Integrated Circuits, Linz, Austria
‡Software Competence Center Hagenberg GmbH (SCCH), Hagenberg, Austria
{gerold.fink, philipp.ebner, sudip.poddar, robert.wille}@jku.at
https://iic.jku.at/eda/research/microfluidics

***Abstract*—Microfluidic devices, often in the form of Lab-on-a-Chip (LoCs), are successfully utilized in many domains such as medicine, chemistry, biology, etc. However, neither the fabrication process nor the respectively used materials are perfect and, thus, defects are frequently induced into the actual physical realization of the device. This is especially critical for sensitive devices such as droplet-based microfluidic networks that are able to route droplets inside channels along different paths by only exploiting passive hydrodynamic effects. However, these passive hydrodynamic effects are very sensitive and already slight changes of parameters (e.g., in the channel width) can alter the behavior, even in such a way that the intended functionality of the network breaks. Hence, it is important that microfluidic networks become robust against such defects in order to prevent erroneous behavior. But considering such defects during the design process is a non-trivial task and, therefore, designers mostly neglected such considerations thus far. To overcome this problem, we propose a *robustness improvement process* that allows to optimize an initial design in such a way that it becomes more robust against defects (while still retaining the original behavior of the initial design). To this end, we first utilize a metric to compare the robustness of different designs and, afterwards, discuss methods that aim to improve the robustness. The metric and methods are demonstrated by an example and also tested on several networks to show the validity of the robustness improvement process.**

I. Introduction

Microfluidics has managed to minimize complex biochemical operations usually realized with unwieldy and expensive lab equipment to single chips at micro-scale level, commonly known as Lab-on-a-Chip (LoC) [1], [2]. Such LoCs allow to perform experiments in a parallel and automated fashion, while benefiting from high throughputs, small reagent volumes, fast reaction times and, in general, a high cost efficiency. As a result, they have been successfully utilized in several fields such as medicine, chemistry, biology, etc.

In general, microfluidic devices are rather sensitive systems, in which already slight changes of a parameter (e.g., width of a channel) may alter the behavior of the device and, hence, break the intended functionality. An example of such sensitive devices are droplet-based microfluidic networks [3], [4], [5], [6], [7] (which are a special type of LoCs) that operate by transporting small droplets through closed channels towards particular modules. These droplets usually contain some kind of biological or chemical reagents, which get processed inside these modules (e.g., heated, mixed, incubated, etc.). Additionally, the paths of the droplets are solely controlled by passive hydrodynamic effects, without using any active control elements such as valves. This routing ability allows to transport droplets to different destinations/modules and, by this, allows to perform different experiments.

However, these passive hydrodynamic effects are rather sensitive and, thus, slight changes of a parameter (e.g., channel width) can break the intended functionality of the network – rendering the device useless. This sensitivity is especially critical when an actual physical realization of a microfluidic network is produced. Usually, this can be achieved with multiple base materials (e.g., Polydimethylsiloxane (PDMS), Polymethylmethacrylate (PMMA), etc.) and through several fabrication techniques such as 3D-printing, soft-lithography, and milling processes. However, neither the fabrication processes nor the used materials are perfect and, thus, a physical realization of a microfluidic network will always suffer from inevitable defects [8]. For example, such defects are:

- **Fabrication tolerances:** Fabrication processes always induce fabrication tolerances which may lead to an unexpected behavior of the network [9].
- **Deformation and Swelling:** Certain materials such as PDMS may deform or swell under pressure-driven flows or specific solvents, which may have a direct impact on the behavior of such devices [10], [11].

Hence, it is important that such microfluidic networks are designed in such a way that they become robust against defects in order to prevent erroneous behavior. However, considering such defects during the design process is a rather complex and not straight-forward task, because designing a microfluidic network with a desired behavior is already a non-trivial problem on its own. As a result, designers commonly neglect such considerations, which frequently leads to a trial-and-error approach, i.e., the device gets fabricated and is then tested if it fulfills the desired functionality. If this is not the case, the design has to be revised and the whole process starts all over again – resulting in time-consuming and costly debugging loops.

In this work, we are proposing an approach which, for the first time, aims to address this problem already at the design phase, i.e., before a first device is even fabricated. The goal is to evaluate a given design prior to fabrication and to automatically conduct changes which eventually makes the design more robust against defects such as mentioned above. Since this is a highly non-trivial task, we propose an entire robustness improvement process including a metric evaluating the current robustness, an analysis of the most sensitive components, and methods that actually modify the design to make it more robust against defects. The entire process is solely based on simulations and, thus, can be applied very early in the design process, even before a physical realization of the corresponding microfluidic device is fabricated – avoiding the trial-and-error scheme and costly iteration loops for re-design mentioned above. Moreover, all implementations are available at https://iic.jku.at/eda/research/microfluidics/robustness and will, for the first time, allow to improve the robustness of microfluidic networks.

The remainder of this work is structured as follows: First, we take a closer look at microfluidic networks, motivate the example used throughout the work, and cover the physical model which is utilized for the robustness improvement process in Sec. II. In Sec. III, we utilize a metric that is able to compare different designs regarding their robustness. Afterwards, we discuss methods and means that can be used to improve the robustness of an initial design in Sec. IV. The results of these methods for several networks are presented and discussed in Sec. V, before the paper is finally concluded in Sec. VI.

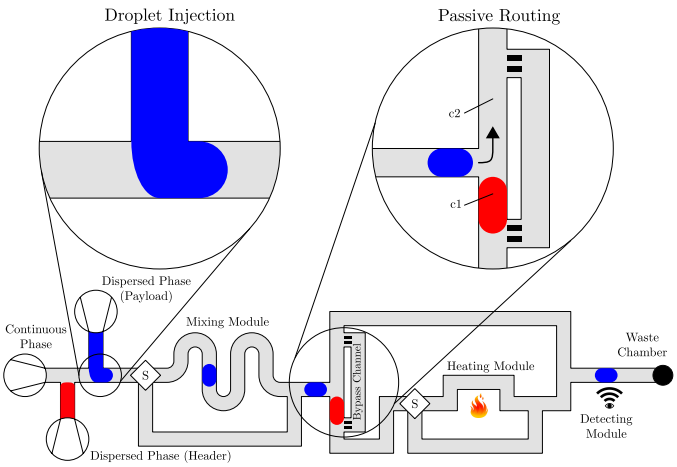

Fig. 1: Droplet-based microfluidic network.

II. Background

In this section, we briefly discuss the basic concept of microfluidic networks and the 1D analysis model, which provides the basis for the proposed robustness process. Additionally, we introduce a running example, which is used throughout this work to illustrate the proposed methods.

A. Microfluidic Networks

Droplet-based microfluidic networks [3], [4], [5], [6], [7] allow to route droplets through various modules (e.g., mixing, heating) in an arbitrary way and, thus, allow to conduct different experiments on a single microfluidic chip. In order to accomplish this task, such networks rely on basic concepts that are illustrated in Fig. 1. At first, droplets are formed by injecting the so-called dispersed phase through a T-junction into the continuous phase that acts as a transport fluid for the droplets. Here, two kinds of droplets are formed, namely payload droplets (blue) that contain some kind of bio/chemical samples, and header droplets (red) that do not contain any samples and are only used to route payload droplets along a desired path. The paths of the payload droplets can be controlled by microfluidic switches (in form of bifurcations), which solely rely on passive hydrodynamic effects as described in the following.

The successor channels c_1 and c_2 of the bifurcation shown in the top-right side of Fig. 1 have a certain hydrodynamic resistance, which mainly depends on channel geometry as well as the viscosity of the continuous phase. A droplet flowing towards such a bifurcation will always take the path with the lowest hydrodynamic resistance, which we assume to be the channel c_1 in this case (hence, this channel is also called the default channel). Moreover, a droplet increases the resistance of the channel it is currently in. Therefore, when a second droplet reaches the bifurcation while the first droplet is still inside channel c_1, then the second droplet will be routed into the non-default channel c_2, if the combined resistance of c_1 and the first droplet is greater than the sole resistance of channel c_2.

Overall, this means when a payload droplet should be routed towards a non-default channel inside a microfluidic switch, then it has to be ensured that a header droplet occupies the default channel at the moment the payload droplet reaches the bifurcation, which will eventually trigger the switching mechanism. This concept allows to route payload droplets along different paths inside the microfluidic network (realizing different experiments) only by exploiting passive hydrodynamic effects.

Moreover, it is usually not desired that header droplets flow through certain modules, since they should not interact with them. In order to prevent this, so-called droplet-by-size sorters [12] are utilized in front of each module, which are indicated by the diamond shaped symbols marked with an "S" inside Fig. 1. These sorters are able to distinguish between different volumes of droplets and, thus, allow to route the smaller payload droplets (denoted in blue) along different paths than header droplets with a bigger volume (denoted in red).

Example 1. *In the following, we will consider a microfluidic network composed of 8 modules and 34 channels as running example. This network is conceptually equal to Fig. 1, i.e., multiple modules are interconnected through channels and microfluidic switches. Moreover, the network is able to conduct three different experiments, i.e., a payload droplet can be subject to one of three different sequences of microfluidic operations (realized by the 8 modules). While the exact design of the network is not relevant to demonstrate the robustness improvement process, we completely disclose it and its specification online at https://iic.jku.at/eda/research/microfluidics/robustness.*

B. 1D Analysis Model

In order to work with the concepts reviewed above, the one-dimensional (1D) analysis model [13], [14] can be utilized. The model itself can be applied when a fully developed and laminar flow (usually at low Reynolds numbers) [9] occurs, which is almost always the case in microfluidic devices. Then the flow inside channels can be described by *Hagen-Poiseuille's law* [15], [13]

$$\Delta P = Q \cdot R, \tag{1}$$

where Q is the volumetric flow rate, ΔP the pressure gradient, and R the hydrodynamic resistance. This resistance depends on the geometry of the channel (i.e., length l, width w, and height h) as well as the dynamic viscosity of the continuous phase μ_c For rectangular channels with a section ratio $h/w < 1$, this resistance can be determined by [16]

$$R = 12 \left[1 - \frac{192\,h}{\pi^5\,w} \tanh\left(\frac{\pi\,w}{2\,h}\right)\right]^{-1} \frac{\mu_c\,l}{w\,h^3}\ . \tag{2}$$

Additionally, droplets increase the resistance of the channel they are currently in by an amount that is directly proportional to the volume (or length l_d) of the droplet

$$R_d = b\ 12 \left[1 - \frac{192\,h}{\pi^5\,w} \tanh\left(\frac{\pi\,w}{2\,h}\right)\right]^{-1} \frac{\mu_c l_d}{w\,h^3}\ . \tag{3}$$

Here b is a factor that indicates how much more the droplet increases the resistance of the channel segment it occupies and is about $2-5$ times according to [17].

Moreover, Hagen-Poiseuille's law in the microfluidic domain is equivalent to Ohm's law in the electrical domain. This means, a microfluidic network can be easily converted to an equivalent electrical resistor network. Therefore, the pressure drops and flow rates inside the channels can be calculated by applying well-known methods for electrical circuits. This eventually allows to efficiently simulate the behavior of microfluidic networks as proposed in [18], [19], [13], [20], [21].

III. Robustness of Microfluidic Networks

In this section, we first motivate the necessity of improving the robustness of microfluidic networks and the advantages that come along with it. Afterwards, we discuss how the robustness of a microfluidic network can be quantified and demonstrate this on the running example. By this, we motivate the robustness method proposed in the following section.

A. Motivation

Controlling the fluid flow inside a microfluidic network only by the hydraulic resistances of the channels and injection velocities is a delicate task. Already slight changes in the channel's resistances may impair the routing of single droplets and, hence, break the correct behavior of the network. This becomes even more severe considering that, additionally, neither the fabrication process for microfluidic networks nor the respectively used materials are perfect. As a result, all sorts of defects can occur inside a microfluidic network, which can alter the behavior, even in such a way, that the fabricated device becomes useless. Considering such defects during the design process is a rather complex and not straight-forward task and, thus, designers mostly neglected such considerations thus far – frequently leading to the production of erroneous microfluidic networks.

In the remainder of this work, we aim at addressing this problem, by proposing a robustness improvement process that allows to optimize an initial design in such a way that it becomes robust against such defects (while still retaining the original behavior of the initial design). To accomplish this, we first utilize a *robustness metric* that allows to measure the impact of defects on the behavior of a device. This metric can then be used to compare different designs regarding their robustness and, by this, quantify which design is more reliable when defects occur. Then, the improvement process itself can be broken down into three steps (which are described in the following section): (1) obtain the robustness metric value of the initial design, (2) apply particular methods that are able to improve the robustness of the initial design, (3) validate the improvement by comparing the robustness metric of the initial and the improved design.

B. Robustness Metric

In order to properly define the *robustness* of a microfluidic network, a corresponding metric must be able to measure the impact of defects on the (intended) behavior of the device. Recently, a corresponding robustness metric has been proposed in [22] which would serve this purpose and, hence, is considered in the following[1]. Here, a Multi-Defect-Model is proposed in which, as the name suggests, multiple defects are randomly "injected" into the microfluidic network and, afterwards, their effect on the behavior of the device is analyzed. Based on that, the robustness of the device is quantified by a single value.

To this end, however, we obviously need a precise definition of what is meant by the "(intended) behavior of the device". This is done by specifying (measurable) objectives that allow to check if the network works as intended or not. For example, such objectives can be:

- The ratio between two flow rates must have a certain value.
- A droplet has to follow a desired path inside a microfluidic network.
- A droplet has to stay inside a trap and must not be squeezed through any gap.
- The time a droplet needs to pass a specific channel must be beneath/above a given value.

If the objectives are satisfied, then the network behaves as intended; otherwise, the designer knows that the injected defects caused the network functionality to break. Of course, the more objectives the more accurate the behavior can be described and, thus, the better the optimization towards a robust network works.

[1]However, please note that the methods proposed in this work are also applicable for any other robustness metric.

TABLE I: Results of robustness metric

Network	σ	N	N_{success}	p_{success}
8 modules 34 channels	0.005	1200	665	55.42%

Having the objective, the question remains how to evaluate whether those are satisfied or not, after injecting the defects. Since doing that by actually fabricating a device is not feasible (would be way too costly and time-consuming when this has to be done multiple times), simulations are used instead (e.g., utilizing the 1D analysis model described in Sec. II-B). Overall, this yields the following process of determining the robustness of a microfluidic network[2]:

1) Inject multiple random defects into the network, where the magnitude of the defects depends on a normal Gaussian distribution with a certain standard deviation.
2) Simulate the defective network and check if all objectives are still satisfied. If so, mark the simulation as *success*.
3) Repeat the first two steps N times, which also results in a number N_{success} that corresponds to the number of simulations marked as *success*.
4) Then, the ratio $p_{\text{success}} = N_{\text{success}}/N$ represents how likely it is to get a correctly working network with the corresponding standard deviation of the normal Gaussian distribution. This ratio serves as value for the robustness metric.

Example 2. *Applying the robustness metric reviewed above to the running example introduced in Ex. 1, we get the results shown in Tab. I, where Fig. 2 illustrates the course of p_{success} with respect to the number of simulations. The second column σ shows the standard deviation of the normal Gaussian distribution for the defects that were randomly injected into the network. N shows the total number of simulations that were conducted, while N_{success} represents the number of simulations which still satisfied the objectives. More precisely, the objectives were satisfied when each of the desired experiments could be processed, i.e., when the payload droplets could be routed into the correct modules. The actual value indicating the robustness is represented by p_{success}, which is the ratio (given in percentage) between N_{success} and N. This value now indicates, that only about 55.42% of the manufactured devices would work correctly when defects (e.g., from the fabrication process) occur that have a standard deviation of $\sigma = 0.005$.*

This robustness metric itself can already be a very helpful value during the design process, because it allows to estimate how likely it is that defects make a fabricated device useless and if modifications/optimizations of the design should be conducted. However, we will use this value rather as a reference, in order to compare the initial design with the improved design obtained by the methods introduced in the next section. Furthermore, note that p_{success} heavily depends on the used standard deviation of the normal Gaussian distribution and, thus, we have to ensure that the robustness metrics are computed with the same standard deviation when comparing the initial with the improved design. Moreover, as shown in Fig. 2 a high number of simulations is required to get a "correct" value of p_{success}. As a result, the computation of the robustness metric value is rather expensive, which is especially critical for larger networks and has to be considered in methods that are presented later.

[2]Note that this is similar to the robustness evaluation of conventional circuits and systems as done, e.g., in [23], [24], [25].

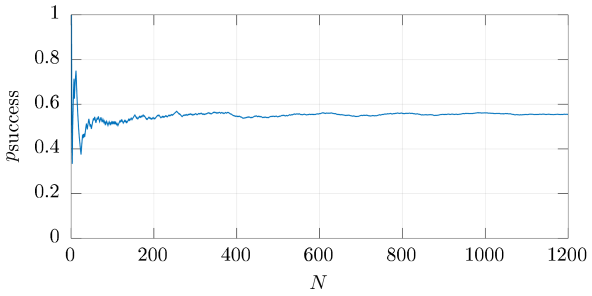

Fig. 2: p_{success} depending on the number of simulations N.

IV. Improving Robustness

In the previous section, we discussed how to quantify the robustness of a microfluidic network and utilized a metric that allows to compare the robustness of different designs. In this section, we now describe methods to improve the robustness of a given design. Basically, this can be achieved by adjusting several parameters in the "right way". Hence, we first discuss which parameters should be considered and, afterwards, we describe the methods that change these parameters in such a way that the robustness of a given design increases.

A. Considered Parameters

The aim of the methods described later in this section is to optimize certain parameters in such a way that the impact of defects is as small as possible, while all objectives of the network are still satisfied (i.e., the initial behavior is retained). While such parameters are often the ones where defects are likely to occur (e.g., channel geometries), the following methods are not only limited to those parameters. In fact, nearly all parameters which are used in the simulation (e.g, channel dimensions, fluid properties, etc.) can be considered for improving the robustness, which gives a great flexibility to the user.

Since parameter values usually have some kind of restriction (e.g., a channel should not be too short/long), we define a lower and upper boundary for each parameter (cf. b_{lower} and b_{upper} in Fig. 3) that can be specified by the user. These boundaries define the search space for the optimization, but also the degree of freedom that can be used to improve the robustness.

Example 3. *Let's consider again the running example. Before we apply any methods that improve the robustness, we have to define which parameters can be considered for the optimization. Let's assume the lengths of the 16 channels which connect the different modules with the corresponding switches are the only parameters of the realization process that offer the potential of improving the robustness*[3]*. This is because they have a great impact on the behavior of the network and, by this, can greatly improve the robustness of the network when they are dimensioned correctly. Besides this, other parameters such as channel widths and heights are often predefined values by the designer that should not be changed at all, since they usually work as some kind of boundary conditions for the network design. Furthermore, we will choose the lower and upper boundary for each channel length as* 50% *and* 200% *of the initial length, respectively.*

B. Sensitivity Analysis

The amount of considered parameters is crucial during the robustness improvement process, since a high number of parameters leads to a complex multi-dimensional optimization problem that can quickly become infeasible to solve. To overcome this problem, we introduce a so-called sensitivity analysis to limit the focus on parameters that tend to increase the robustness the most. This analysis sorts the parameters by their sensitivity, i.e., the impact they have on the behavior of the network when they get modified. A high sensitivity indicates that a parameter has a strong impact on the network and can easily break the intended behavior when the value of the parameter gets slightly changed, for example, as a result of a fabrication error. In contrast, a parameter with a low sensitivity does not have a great effect on the network itself and, thus, should also be robust against changes due to defects. By focusing on the most sensitive parameters during the robustness improvement, we ensure that the optimization problem stays manageable, while still aiming for a more robust design of the network.

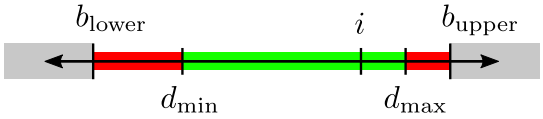

Fig. 3: Variables of a single parameter.

The analysis itself can be conducted as follows, where Fig. 3 provides a corresponding illustration for all mentioned values:

1) For each considered parameter obtain the minimal and maximal defect value (d_{min} and d_{max}), where the objectives of the network are still satisfied, i.e., the network behaves as expected (cf. green range in Fig. 3).
2) Compute the sensitivity S of each parameter according to the following equation, where i is the initial value of the parameter:

$$S = \frac{i}{i - d_{\text{min}}} + \frac{i}{d_{\text{max}} - i} \tag{4}$$

3) Sort the parameters in a descending order with respect to their sensitivity value.

The first step in this analysis is the computationally most expensive task and requires to conduct multiple simulations in order to determine the defect limits (usually done by a binary search algorithm). Note that all other parameters are set to their initial value, while the defect limits of a certain parameter are obtained.

Furthermore, Eq. 4 ensures that S is normalized by the initial value of the parameter and gets quickly larger, the closer the minimal and maximal defect values are to the initial value, which eventually allows to obtain the most sensitive parameters.

Example 4. *Applying the sensitivity analysis proposed above to the 16 channels of the running example yields the sorted parameters shown in Tab. II (for convenience only the 7 most sensitive parameters are listed). The table clearly shows that the parameter* $c7.l$ *(i.e., length of channel* $c7$*) is the most sensitive one and, thus, methods that want to increase the robustness of the network should definitely have a focus on this parameter.*

Based on the results of the sensitivity analysis, now actually improving the robustness can start. To this end, two methods with different approaches are proposed in the following. The Single-Parameter-Variation method tries to improve the robustness metric value in an indirect way, hence, it does not have to determine a computationally expensive robustness metric. In contrast, the Downhill-Simplex method aims to improve the robustness metric directly and, thus, usually has a higher run-time.

[3]Note that not all channels connect modules and switches (some are part of the switches themselves) and, thus, only 16 out of 34 channels of the running example are considered.

TABLE II: Results of sensitivity analysis

Parameters	*c7.1*	*c23.1*	*c12.1*	*c21.1*	*c1.1*	*c10.1*	*c19.1*
Sensitivity	39.78	21.32	17.27	12.85	12.68	8.16	7.87

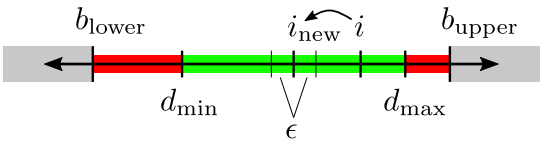

Fig. 4: Concept of Single-Parameter-Variation method.

C. Single-Parameter-Variation Method

The Single-Parameter-Variation method is a rather simple method, but is also quite fast, since no actual value of the robustness metric has to be calculated and, thus, the number of conducted simulations is kept quite small. Basically this method focuses only on a single parameter at a time in order to improve the robustness of the overall network. The following steps describe the working principles of the method in more detail, while Fig. 4 provides a corresponding illustration:

1) Conduct a sensitivity analysis for all considered parameters.
2) Loop through the sorted parameter list and perform the following tasks:
   - Skip the parameter (and consider the next one) if the initial value is within a tolerance range ϵ, that lies around the midpoint of the minimal and maximal defect values, i.e., if the following equation is satisfied:
   $$\left| \frac{i - d_{\text{min}}}{d_{\text{max}} - d_{\text{min}}} - 0.5 \right| < \epsilon \tag{5}$$
   - Otherwise, set the new initial value of the parameter to the midpoint of the minimal and maximal defect values, according to:
   $$i_{\text{new}} = \frac{d_{\text{min}} + d_{\text{max}}}{2} \tag{6}$$
3) Continue with Step 1 if no termination criterion is reached, such as:
   - Maximal number of iterations was reached.
   - The n most sensitive parameters were skipped, where n can be defined by the user.

This method ensures that parameters, that do not already lie close to the midpoint of their minimal and maximal defect values, are set exactly to this midpoint (cf. Fig. 4). As a result, the network becomes more robust, since the considered parameters can now alter in a wider range before the desired network functionality breaks. Moreover, it is often sufficient when not all parameters are improved but rather only the n most sensitive parameters (where n can be specified by the user), because parameters that do not have a high sensitivity usually contribute very little to the overall robustness of the network.

D. Downhill-Simplex Method

This approach uses the Downhill-Simplex method [26] (also called Nelder-Mead-Simplex) in order to improve the robustness of a given design. This method is generally used to optimize n-dimensional cost functions ($f : \mathbb{R}^n \rightarrow \mathbb{R}$) where no derivative is known. Basically it uses $n + 1$ test points in form of a simplex, where the cost function values of these test points are compared in order to approximate the tendency of the values and the gradient towards the optimum. By replacing "bad" test points in a clever way with better ones the Downhill-Simplex method progresses and the simplex moves along the parameter space until it finds a proper optimum.

In our case, the cost function corresponds to the robustness metric and the multi-dimensional space is defined by the parameters that are considered for the optimization. Hence, compared to the Single-Parameter-Variation method the Downhill-Simplex algorithm tries to improve the robustness of the network (i.e., the cost function) in a direct way. As a result, this method has to compute multiple values of the robustness metric to find a proper maximum, which is very expensive in terms of computational time (especially for larger networks and many considered parameters).

To overcome this problem, we conduct a sensitivity analysis again, before we start the Downhill-Simplex method, where only the most sensitive parameters are taken into account for the optimization, so the problem stays manageable. How many sensitive parameters are considered can be specified by the user and usually depends on the computational time of the robustness metric and, by this, on the size of the network. The implementation of the Downhill-Simplex algorithm is already available for Java inside the *Math* component of the *Apache Commons* project (http://commons.apache.org/proper/commons-math/). We initialized the algorithm with default values and build the start simplex around the initial parameters. Moreover, we used bounded functions to guarantee that the method searches inside the lower and upper bound of the parameters.

V. Application & Experiments

With the methods proposed above (i.e., a metric allowing one to measure the robustness of the current network, a sensitivity analysis allowing to evaluate which components should be modified to improve robustness, as well as the proposed methods to actually conduct changes for this purpose), for the first time, a complete process for improving the robustness of microfluidic networks during the design phase is available. This section shows how to apply it and how it improves robustness; using again the running example as a use case as well as several other designs that differ in the number of modules and channels. In addition to that, all network designs as well as the implementations are available at https://iic.jku.at/eda/research/microfluidics/robustness and can be used to improve the robustness of other devices as well (or to conduct further evaluations).

At first, we compute the initial value for the robustness metric $p_{\text{success}}^{(\text{initial})}$ for each network[4]. Afterwards, the two proposed methods are applied, where only the channel lengths between the switches and modules are considered as the parameters that should be optimized. Finally, we compute the new value for p_{success} and compare the improvement with the initial design. The results are summarized in Tab. III, where each column represents a different network with the corresponding number of modules and channels. The second row σ indicates the standard deviations of the defects that were used to compute the value of the initial and improved robustness metrics.

As described in Sec. IV-C, the variables for the Single-Parameter-Variation method are the tolerance range ϵ and the maximal number of sensitive parameters n, that should be skipped for the termination criterion of the method. Moreover, the number of conducted iterations I as well as the overall computational time t are also shown. For the Downhill-Simplex method, the number of cost function calls I (i.e., the number of robustness metric computations) and the number of the n most sensitive parameters that should be considered are given for each network, while t indicates the overall computational times again.

[4]Please note, that different values of σ are used for the networks in order to get proper values for $p_{\text{success}}^{(\text{initial})}$.

TABLE III: Benchmarks

	Networks		
Modules/Channels	8/34	10/67	15/101
σ	0.005	0.002	0.002
$p_{\text{success}}^{(\text{initial})}$	55.42%	40.91%	46.07%
Single-Parameter-Variation:			
ϵ	0.05	0.05	0.05
n	16 (all)	8	8
I	24	30	10
t	31 s	21 min	26 min
p_{success}	64.86%	74.81%	59.10%
Downhill-Simplex:			
n	16 (all)	5	3
I	109	38	22
t	9 min	32 min	96 min
p_{success}	63.86%	53.50%	62.20%

As shown in the table, both methods improve the robustness metric p_{success} of all networks, which confirms the validity of the proposed robustness improvement process. While both methods provide nearly the same improvements for the first ($\sim 8-9\%$) and third ($\sim 13-16\%$) network, the Single-Parameter-Variation method is able to improve the robustness of the second network drastically of about 34% (here, the increase of the Downhill-Simplex-Method is also very good with about 13%). As expected, the run-time of both methods significantly increases with the size of the network (and also with the number of considered parameters n). This behavior is also known in the design of conventional circuits/systems, where long run-times of methods that increase the robustness are not uncommon, even for moderate sized systems [23], [24], [25]. Especially the Downhill-Simplex method has considerable performance issues when dealing with larger networks, because it has to obtain the computational expensive robustness metric multiple times. To this end, we limited n for the larger two networks in order to get reasonable computational times. As the results confirm, this is sufficient to gain a considerably better robustness, since the sensitive parameters are usually the ones which tend to increase the robustness the most.

Overall, designing robust microfluidic networks remains a hard task, since these systems are very sensitive and controlling the droplet movement only by hydraulic resistances and injection times remains tedious. However, as the results confirm, the process proposed in this work provides a good step forward to achieve this goal.

VI. Conclusion

In this work, we proposed a robustness improvement process which allows to optimize an initial design of a microfluidic network in such a way, that it becomes robust against defects. This is important, because defects are likely to occur during the fabrication process or due to swelling/deformation of the used material under pressure-driven flows and can result in erroneous behavior of the microfluidic networks. To this end, we first utilized a metric that is able to compare the robustness of two different designs. Afterwards, we discussed methods and means which aim to improve the robustness of a given design and tested them on several microfluidic networks. The results showed, that the process effectively improves the robustness of microfluidic networks and, by this, breaks costly debugging loops for re-designs. This finally reduces manufacturing costs and times, since it can be applied early in the design process even before a first device is fabricated.

Acknowledgments

This work has partially been supported by the Linz Institute of Technology and by BMK, BMDW, and the State of Upper Austria in the frame of the COMET Programme managed by FFG.

References

[1] 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.

[2] 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.

[3] E. De Leo, L. Galluccio, A. Lombardo, and G. Morabito, "Networked labs-on-a-chip (NLoC): Introducing networking technologies in microfluidic systems," *Nano Communication Networks*, vol. 3, no. 4, pp. 217–228, 2012.

[4] E. De Leo, L. Donvito, L. Galluccio, A. Lombardo, G. Morabito, and L. M. Zanoli, "Communications and switching in microfluidic systems: Pure hydrodynamic control for networking Labs-on-a-Chip," *Trans. on Communications*, vol. 61, no. 11, pp. 4663–4677, 2013.

[5] L. Donvito, L. Galluccio, A. Lombardo, and G. Morabito, "μ-NET: a network for molecular biology applications in microfluidic chips," *Trans. on Networking*, 2015.

[6] 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.

[7] A. Grimmer, W. Haselmayr, A. Springer, and R. Wille, "Design of application-specific architectures for Networked Labs-on-Chips," *Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 1, pp. 193–202, 2018.

[8] S. K. Sia and G. M. Whitesides, "Microfluidic devices fabricated in poly (dimethylsiloxane) for biological studies," *Electrophoresis*, vol. 24, no. 21, pp. 3563–3576, 2003.

[9] F. Jousse, G. Lian, R. Janes, and J. Melrose, "Compact model for multiphase liquid–liquid flows in micro-fluidic devices," *Lab on a Chip*, vol. 5, no. 6, pp. 646–656, 2005.

[10] M. Kim, Y. Huang, K. Choi, and C. H. Hidrovo, "The improved resistance of PDMS to pressure-induced deformation and chemical solvent swelling for microfluidic devices," *Microelectronic Engineering*, vol. 124, pp. 66–75, 2014.

[11] J. N. Lee, C. Park, and G. M. Whitesides, "Solvent compatibility of poly (dimethylsiloxane)-based microfluidic devices," *Analytical Chemistry*, vol. 75, no. 23, pp. 6544–6554, 2003.

[12] Y.-C. Tan, Y. L. Ho, and A. Lee, "Microfluidic sorting of droplets by size," *Microfluidics and Nanofluidics*, vol. 4, no. 4, pp. 343–348, 2008.

[13] K. W. Oh, K. Lee, B. Ahn, and E. P. Furlani, "Design of pressure-driven microfluidic networks using electric circuit analogy," *Lab on a Chip*, vol. 12, no. 3, pp. 515–545, 2012.

[14] M. Schindler and A. Ajdari, "Droplet traffic in microfluidic networks: A simple model for understanding and designing," *Physical Review Letters*, vol. 100, no. 4, p. 044501, 2008.

[15] H. Bruus, *Theoretical microfluidics*. Oxford university press Oxford, 2008, vol. 18.

[16] M. J. Fuerstman, A. Lai, M. E. Thurlow, S. S. Shevkoplyas, H. A. Stone, and G. M. Whitesides, "The pressure drop along rectangular microchannels containing bubbles," *Lab on a Chip*, vol. 7, no. 11, pp. 1479–1489, 2007.

[17] T. Glawdel and C. L. Ren, "Global network design for robust operation of microfluidic droplet generators with pressure-driven flow," *Microfluidics and Nanofluidics*, vol. 13, no. 3, pp. 469–480, 2012.

[18] A. Biral and A. Zanella, "Introducing purely hydrodynamic networking functionalities into microfluidic systems," *Nano Communication Networks*, vol. 4, no. 4, pp. 205–215, 2013.

[19] A. Grimmer, M. Hamidović, W. Haselmayr, and R. Wille, "Advanced simulation of droplet microfluidics," *Journal on Emerging Technologies in Computing Systems*, 2019.

[20] A. Grimmer, X. Chen, M. Hamidović, W. Haselmayr, C. L. Ren, and R. Wille, "Simulation before fabrication: a case study on the utilization of simulators for the design of droplet microfluidic networks," *RSC Advances*, vol. 8, pp. 34 733–34 742, 2018. [Online]. Available: http://dx.doi.org/10.1039/C8RA05531A

[21] G. Fink, P. Ebner, M. Hamidović, W. Haselmayr, and R. Wille, "Accurate and efficient simulation of microfluidic networks," in *Asia and South Pacific Design Automation Conference*, 2021, pp. 85–90.

[22] G. Fink, A. Grimmer, M. Hamidović, W. Haselmayr, and R. Wille, "Robustness analysis for droplet-based microfluidic networks," *Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 2019.

[23] U. Krautz, M. Pflanz, C. Jacobi, H.-W. Tast, K. Weber, and H. T. Vierhaus, "Evaluating coverage of error detection logic for soft errors using formal methods," in *Proceedings of the Design Automation & Test in Europe Conference*, vol. 1. IEEE, 2006, pp. 1–6.

[24] L. Doyen, T. A. Henzinger, A. Legay, and D. Nickovic, "Robustness of sequential circuits," in *2010 10th International Conference on Application of Concurrency to System Design*. IEEE, 2010, pp. 77–84.

[25] S. Huhn, S. Frehse, R. Wille, and R. Drechsler, "Determining application-specific knowledge for improving robustness of sequential circuits," *Trans. on Very Large Scale Integration Systems*, vol. 27, no. 4, pp. 875–887, 2019.

[26] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The computer journal*, vol. 7, no. 4, pp. 308–313, 1965.