Dynamic Radial Placement and Routing in Paper Microfluidics

Joshua Potter[1], William H. Grover[2], and Philip Brisk[1]

***Abstract*—The low cost, simplicity, and ease of use of paper microfluidic devices have made them valuable medical diagnostics for applications from pregnancy testing to COVID-19 screening. Meanwhile, the increasing complexity of paper-based microfluidic devices is driving the need to produce new tools and methodologies that enable more robust biological diagnostics and potential therapeutic applications. A new design framework is being used to facilitate both research and fabrication of paper-based microfluidic biological devices to accelerate the investigative process and reduce material utilization and manpower. In this work we present a methodology for this framework to dynamically place and route microfluidic components in a non-discrete design space where fluid volume usage, surface area utilization, and the timing required to perform specified biological assays are accounted for and optimized while also accelerating the development of potentially lifesaving new devices.**

***Index Terms*—Paper Microfluidics, Placement and Routing, Continuous Placement**

I. Introduction

THE THREAT of a global pandemic on the scale of tens of millions of people infected and deceased like the 1918 Spanish flu [1] or a strain of influenza like H1N1 [2] has been of concern for nearly 100 years. Additionally, viral and non-viral diseases such as AIDS and tuberculosis also kill around 5 million people per year, 4.3 million die from respiratory infections, 2.9 million die from enteric infections, and 1 million die from malaria. The current COVID-19 pandemic has now underscored the need for rapid, inexpensive, and widely-available medical diagnostics.

Although such diseases are a threat to everyone, populations of lower socio-economic status – such as many third world countries – have an elevated risk of exposure and less diagnostic and therapeutic resources to treat such conditions [3]. Even in more prosperous countries there can be a potentially fatal gap in healthcare among those who have a dysfunctional political system, or predatory economic entities that have developed extensive governmental agency influence that can restrict needed resources from the general population. Meanwhile, efforts to identify and mitigate global threats of diseases [4], [5] are continually being developed and deployed wherever resources are limited and needs are greatest.

Low-cost and easy-to-use diagnostic technologies have the potential to positively impact healthcare outcomes in critical situations, and many researchers look to microfluidic technologies for its potential to achieve that impact. Microfluidic devices can reduce costs of materials through reducing their size in construction as well as lowering the volumes of reagents and fluids needed for testing. Moreover, they have the potential to enable doctors and researchers to perform diagnostic and treatments while in the field and reducing the need for time-consuming formal laboratory work as well as cutting the time to treatment and recovery of patients.

In 2004, the World Health Organization (WHO) specified the seven ASSURED criteria that it determines to be essential for point-of-care diagnostics in resource-limited settings [3] (Fig. 1). ***Paper Microfluidic Devices*** may come the closest to satisfying these criteria; they are versatile, inexpensive to produce, and easy to use. These advantages are why paper microfluidic devices are currently being used to detect COVID-19 antibodies in blood to determine if a patient was previously infected by the novel coronavirus SARS-CoV-2 [6].

A. Contribution

This paper introduces a design framework for paper microfluidic devices that will facilitate both research on design automation and subsequent fabrication studies. Similar to semiconductors, paper microfluidic devices feature components (akin to standard cells or IP blocks) connected by fluid transport channels (akin to wires). Placement and routing problems clearly exist, although at present, design rules have not been standardized, formal problem descriptions are lacking, and no heuristics have been published to date. The physical design of a paper microfluidic device must account for the underlying physics of passive fluid transport (e.g., wicking) [7], [8] and the physical properties of the paper substrate [9]. Lack of standardization suggests that design constraints akin to standard cells for paper microfluidic devices are unlikely to emerge in the foreseeable future.

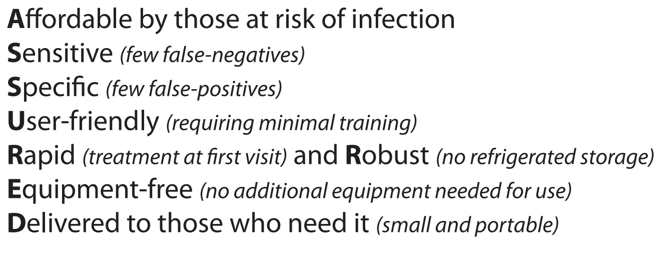
Affordable by those at risk of infection
Sensitive *(few false-negatives)*
Specific *(few false-positives)*
User-friendly *(requiring minimal training)*
Rapid *(treatment at first visit)* and **R**obust *(no refrigerated storage)*
Equipment-free *(no additional equipment needed for use)*
Delivered to those who need it *(small and portable)*

Fig. 1. The WHO's "ASSURED" criteria for point-of-care diagnostics in resource-limited settings.

[1] *Department of Computer Science and Engineering, University of California, Riverside, CA 92521 USA*
[2] *Department of Bioengineering, University of California, Riverside, CA 92521 USA; E-mail: wgrover@engr.ucr.edu*
Manuscript received ; revised.

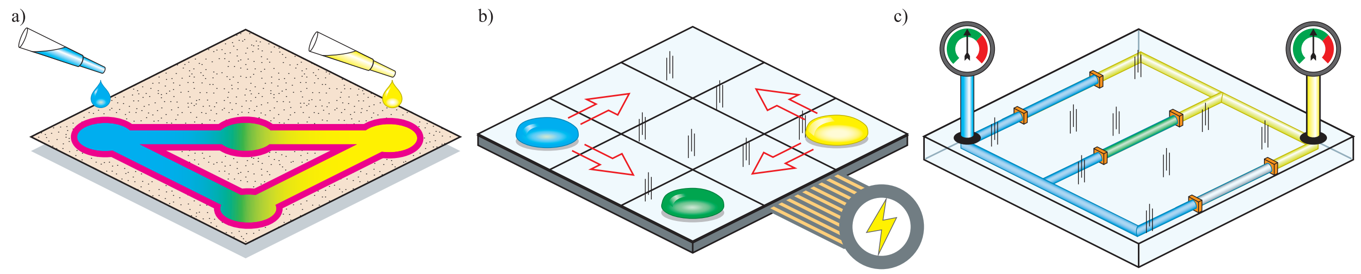

Fig. 2. *A mixing operation using various platforms* a) On paper, fluid and reagents are delivered via pipette and fluid travels using capillary action bounded by printed wax barriers (in pink). Fluid channels are non-discrete so travel can occur in any planar direction. b) Electro-wetting utilizes an electric power source that controls electrostatic pads below a hydrophobic surface material to induce droplet travel. Consequently, droplets may only mode orthogonally from pad to pad. c) Pressurized channels and valves (represented by orange blocks) force fluid through embedded channels in a block of material that may either be machined with the channels, or 3-D printed.

B. Technology Overview

The vast majority of papers published on design automation for microfluidics over the past 15-20 years have targeted two specific microfluidic technologies: electrowetting on dielectric (EWoD – often called "Digital Microfluidics") [10] and channel-based microfluidics featuring integrated microvalves which are controlled via external solenoid valves [11]. From the technological perspective, paper is fundamentally different.

Fig. 2 depicts fluid transport and mixing in these three technologies. In paper microfluidics (Fig. 2a) liquid expands in a radial pattern from the application point, in accordance with the theory of capillary action; the underlying physics is no different than using a commodity paper towel to mop up a fluid spill. Transport and routing of fluids within the paper is handled through various printed hydrophobic barriers, such as wax-based inks [12], [13], [14], [15]. The wax barrier impedes radial flow, but there is no external source beyond the substrate itself which pumps the fluid. This is significantly different than either the actuation mechanisms employed in other popular microfluidic technologies or the transport of electrical current in semiconductors, as the forces that are applied to fluids (or electrical currents) in the aforementioned technologies are inherently directional. In electrowetting microfluidics (Fig. 2b), the hydrophobic surface coupled with the pattern of electrodes that are actuated by an externally supplied voltage controls the direction of wetting (transport). In channel-based microfluidics (Fig. 2c), an external syringe pump creates a force which becomes directional due to channel geometry; the same is true of peristaltic pumping, which is internal to the chip, but is controlled by external solenoid valves.

The geometry of a paper microfluidic device determines the volume of the liquids and reagents that are required to successfully complete an assay. The time required for fluid to travel through a (portion of) the substrate can be constrained by both upper and lower bounds, depending on the assay: the upper bound may be due to evaporation, the rate of chemical interactions, and ultimately the amount of time that a person may be willing to wait for results or possible sample spoilage, while the lower bound is typically determined by the minimum time for chemical processes to complete. Further, the materials used in the device (substrates, inks, fluids, reagents, etc.) need to be limited to avoid waste in order to maintain low cost while still delivering efficient, effective, and accurate results.

C. Related Work

Many channel-based continuous fluid flow microfluidic systems have a linear layout in which fluid enters on one side of the chip and travels, under a pressurized flow, to the opposite side [16], [17]; in turn, many physical design algorithms targeting these technologies are based on a similar assumption [18], [19], [20]. We argue that physical design algorithms for paper microfluidics should work with, rather than against, the natural radial flow of fluid in a porous medium; conversely, linear layouts, while simple to generate, are poor choices for paper microfluidic devices; we confirm this argument experimentally.

Both pressure-driven flow through channels and capillary-force-driven flow through paper are convective flows. In the absence of any external force, such as gravity, fluid will flow equally in all directions (i.e., in a spherical direction in 3D or a radial direction in 2D). Channel barriers, which can be realized for paper microfluidic devices via wax printing, may constrain the otherwise uninhibited travel of fluid, but also introduce additional resistance to fluid flow [21]. Furthermore, evaporation, surface tension, and backpressure [22] place limits on the distance fluid can travel in a paper substrate.

Fig. 3 provides empirical evidence that linear layouts, i.e., those that might be generated by the algorithms described in Refs. [18], [19], [20] appropriately adapted for paper microfluidics, do not generate workable devices. We fabricated two linear paper microfluidic devices, as depicted in the top portion of Fig. 3.

Both devices are simple branching routes where fluid is delivered to a large source reservoir so that it will flow toward a control sink at the opposite end. In both devices, twenty-four channels branch off from the main artery to divert the fluid to twenty-four sinks. In one of the two devices, the branches lie at a 90° angle to the main artery; in the other, the design was modified to reduce the angle of fluid diversion when entering the channels. In both of these tests, fluid only reached six of the twenty-four sinks, despite the fact that the fluid delivered to the source exceeded the calculated volume for the device and consequently encountered barrier failure. The second device, with less extreme fluid diversion, offered at best a marginal improvement in the volume of fluid delivered to the sinks, and likewise experienced overflow.

The second set of paper microfluidic devices, which serve to

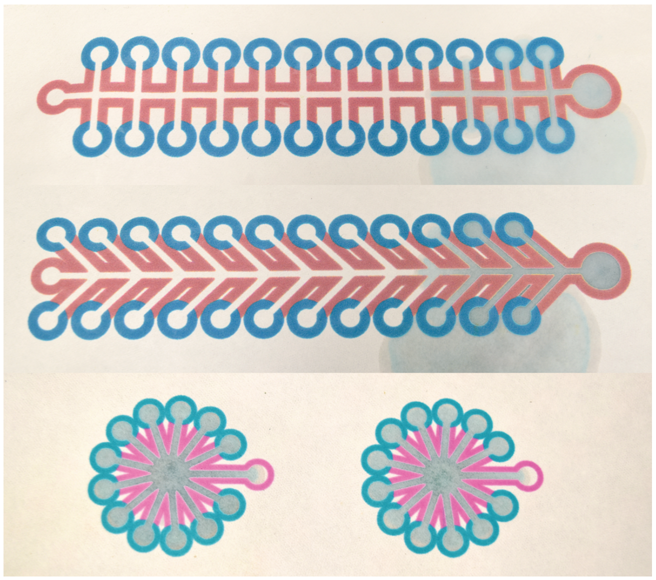

Fig. 3. *Linear vs. Radial Layouts* (Top): A linear paper microfluidic device layout with 24 test reservoirs (blue), one control reservoir (red), a large source reservoir, all with 2 mm barrier widths. The device had a calculated volume of 117.4 μL therefore 140 μL fluid was delivered to the source reservoir. After 8 min 50 secs, only four reservoirs were filled, two additional reservoirs were partially filled, and the 2mm barriers ultimately failed. (Middle): A second linear device was fabricated but the channels were angled to aid in fluid flow. The change in channels increased the device's calculated volume to 137.5 μL, and therefore 160 μL fluid was delivered to the second device. After 9 min 32 secs, four reservoirs were again filled, two more reservoirs partially filled, and the device barriers failed. (Bottom): Two radial device layouts were made with 12 reservoirs (blue) plus 1 control reservoir (magenta), and only 1 mm barrier width. Using radial channels to aid in fluid flow, this device had only a 54.9 μL calculated volume, and consequently 54 μL fluid delivered to the left device and 50 μL to the right device. After 3 min 36 secs all 24 reservoirs filled completely and fluid reached both control reservoirs thereby successfully running to completion. The total fluid delivered to both bottom devices is less than the amount of fluid delivered to the top and middle devices, allowing two sets of tests to be performed using less fluid than the single test and without failure, demonstrating the advantage of a radial layout.

motivate this paper, are shown in the bottom portion of Fig. 3. In these "radial layouts," the source reservoir is placed at the center, and twelve sink reservoirs and one control reservoir were placed equidistant from the source. The radial layouts exploit tendency of fluid to flow in an expanding circle from the point of delivery, while the two linear devices shown on top aim to counteract the fluid's flow. In this experiment, the two radial layouts' twenty-four reservoirs were able to successfully fill, with less fluid, less paper area, and in less time than the two linear devices shown at the top of the figure.

The two devices shown on the bottom of Fig. 3 are smaller and have shorter channel lengths than the two shown on the top. As a matter of principle, similar device geometries could be laid out by appropriately adapting optimal or near-optimal physical design algorithms for continuous flow microfluidic chips that optimize these metrics [23], [24], [25], [26]. Fig. 4 illustrates one key difference between these algorithms and the approach presented here: existing physical design algorithms abstract away each component with a rectangular bounding box, and impose physical layout constraints that bounding boxes cannot overlap, and that fluid channels cannot intersect bounding boxes unless they connect directly to an I/O port of the corresponding component. In contrast, the physical layout algorithm presented here detects component overlap based on component geometry, which is more accurate than the conservative bounding-box approach. As shown in Fig. 4 this yields tighter layouts and shorter routing channels.

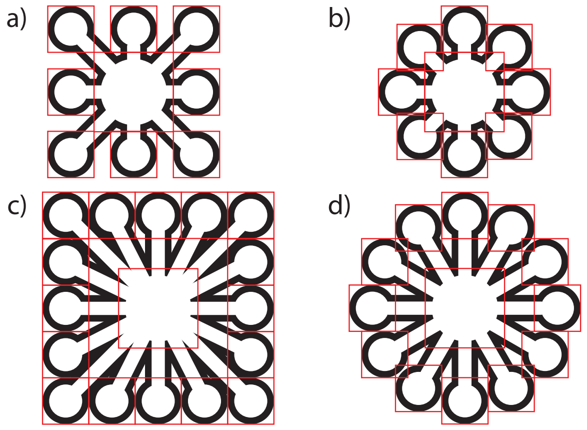

Fig. 4. Using actual geometry versus abstract bounding boxes can significantly reduce fluid area and consequently materials and sample usage. Layout a) using bounding boxes for placement of components has a calculated volume of $284mm^2$ while b) has a calculated volume of $268mm^2$ yielding a 6% improvement over the bounds dominated version. Layout c) has a calculated volume of $618mm^2$, d) has a calculated volume of $482mm^2$ resulting in a 22% reduction of fluid area versus the bounds dominated layout.

The radial layout method presented here takes inspiration from the field of graph drawing. A radial tree (also called a radial map) draws a rooted tree by placing the root at the center of a circle and expanding the tree such that the levels are drawn on concentric circles [27], [28], [29]. We have observed that the layouts produced by our algorithms for tree-shaped netlists do not resemble radial trees, but instead shares some principle similarities to H-trees [30], which were used in early multiprocessor interconnection networks [31] as well as VLSI clock tree routing [32], [33]. Radial tree drawing generalizes to radial graph drawing [34], in which the vertices of a graph are drawn on a set of concentric circles; while our approach can place and route a netlist corresponding to any graph, we do not impose any constraints comparable to radial graph drawing. Additionally, our approach to channel route employs probes, taking inspiration from grid-based maze routers developed in the late 1960s [35], [36].

II. Background

A. Design Automation Challenges

In principle, automated design of paper microfluidic devices – and the individual components that are used to construct them – takes inspiration from semiconductor design automation; however, there are also many important differences. Along with the aforementioned physical limitations of passive-flow fluid delivery, Paper microfluidic device design automation must account for non-discrete and non-linear geometries. Components can be located anywhere within the device, unlike traditional circuit placement which is restricted by grid-oriented standard cells. Components may have non-polygonal geometries, such as Bézier curves (Fig. 6) which complicates the validation of placement legality. Moreover, components may consume 2D space across multiple device layers, while fluid transport, in many cases, crosses substrate boundaries. Thus, straightforward adaptations of existing semiconductor physical design algorithms are inappropriate for paper microfluidic devices, and design tools and methodologies to address these challenges are needed.

B. Paper Microfluidic Design Practice

Paper microfluidic devices are presently designed by hand using software such as AutoCAD® or Adobe Illustrator®. Each design is essentially a “hard-coded” device (analogous to an application-specific integrated circuit). During the development phase, the designer must create multiple device variations to compare performance and accuracy and to ensure that the device remains usable under varying environmental conditions. Under the current paradigm, each variation must be designed by hand, which is time-consuming, labor-intensive, and prone to inaccuracy. These issues also limit the complexity of the biochemical reactions for which a realistic paper microfluidic device can be designed.

III. Design Automation Framework

This paper utilizes a design framework [37] that paper microfluidic device developers can use to prototype, dynamically generate, and test new designs (Fig. 5); at the time of publication, this framework did not feature algorithms for automatic placement and routing, we introduce, for this first time, here. The framework provides the capability to reliably reproduce devices streamlined for *in-situ* fabrication. Moreover, the framework is designed to integrate with tools to test and analyze each design in order to enable automated or semi-automated paper microfluidic device design space exploration in the future. Different device variations may be designed to account for the effects of environmental conditions, impact on physical substrates, and dynamic fluid

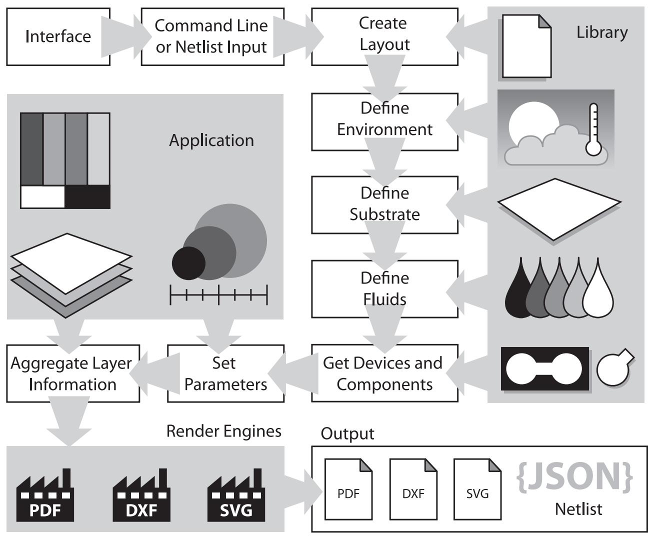

Fig. 5. Paper microfluidic device design framework overview.

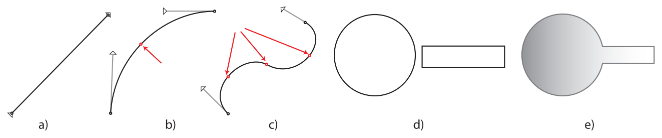

Fig. 6. A Bézier curve is defined by 4 points in space: start and end points and 2 control points that define a parametric curve. The control points may or may not be on the curve itself. The curve may contain 2-5 critical points where it potentially changes direction, and possibly an inflection point where the curve changes direction. a) When the start and end points are also control points, the Bézier curve degenerates to a straight line. b) A curve defined by two control points, indicated by lines connecting them to the start and end point, and one maximum point (indicated by a red arrow. c) A curve with one maximum and one minimum point, along with inflection point between them where the curve changes direction (all three points indicated by red arrows. d) A series of Bézier curves defines a path; when the path starts and ends at the same point, a closed path creates a shape. e) Simple shapes can be joined to create more complex shapes.

conditions, providing the designer with a greater understanding of how these physical factors influence accuracy under test.

The framework includes a library of paper microfluidic device components, which are reusable objects that can be rapidly assembled into netlists to form new diagnostic devices. Components are abstract elements that must reside in one and only one device, but may connect to multiple devices. A component definition may include functionality in terms of fluidic actions and abstract dynamics, such as mixing, transport, timing, etc. Each ***component*** $c = \langle \rho_1, \rho_2, \ldots, \rho_n \rangle$ is physically defined by one or more geometric ***primitives***. A primitive ρ_i is constructed from one or more ***paths*** P_i, each of which using one or more ***Bézier curves*** (Fig. 6). Bézier curves are parametric arcs defined with start and end points and “handle” points that constrain the curve. The curves may have 2-5 ***critical points***: 1) the start, 2) the end, 3) local maximum, 4) local minimum, and 5) an inflection point where the curve can change from concave to convex. The placement phase of our algorithm processes the critical points, as opposed to enumerating all sides, angles, and curves, in order to measure how close objects are placed to one another and to determine whether or not overlap occurs.

A ***device*** $D = \langle c_1, c_2, \ldots, c_n \rangle$ consists of at least one component, and encapsulates the desired actions and parameters needed to characterize its behavior. To create a device, individual components may be scaled or rotated as needed. Large devices may be specified hierarchically in terms of smaller devices, facilitating the concatenation of multiple assays in sequence or in parallel.

A ***netlist*** $N = \langle c_1, c_2, \ldots, c_n \rangle$ is a queue of components (Fig. 7) which determines the order in which components will be placed. Both components and devices may contain ***ports***, which define an interface for fluid transport. For example, a port on device D_1 may connect to a port on component c_1 $\{P_{D_1} \leftarrow P_{c_1}\}$; similarly, a port on device D_2 may connect to two ports on components c_2 and c_3 respectively $\{P_{D_2} \leftarrow P_{c_2,c_3}\}$. The physical location of a port within a component is defined as part of the component’s specification. The physical location of a port within a larger device is not known until the physical location of the component containing that port has been placed within the device.

TABLE I
VARIABLES USED IN THE ALGORITHMS

Vars	Description
c	Component to be placed
cf	The cost factor of Component c placed in relation to one or more previously-placed Devices D
Chs	A set of Channel Components
ch	A Channel Component
D	A Device that contains one or more Components that have been been placed and possibly routed
Δ_δ	The set of minima and maxima for each dimension
I	A set of intersections
L	A Layout containing one or more fully laid-out Devices
N	The netlist; a set of Components to be placed ordered as collections of sources to sinks
O	The path that serves as an outline to a Device or Component
p	A point (x, y)
P	A Path object consisting of one or more Segments s
P_D	A sub-path of the outline of D which consists of all its critical points and the angle about the center each are located
P_c	Similar to P_σ but for a component c
$Page$	Contains the finished Layout L along with information about substrate properties
Φ	Port variable that describes the input (x_i, y_i) and output locations (x_o, y_o) between Components and/or Devices
Q_ρ	A priority queue consisting of the x, y coordinates and rotation θ of Component c to be placed that is sorted on cost factor cf
s	A Segment object defined by a Bézier curve
Θ	The aperture $\langle\theta_\alpha, \theta_\omega\rangle$ of exposure between a component c that is the starting angle θ_α and the ending angle θ_ω about its center point
$\theta_{\alpha,\omega}$	The start and stop angles of an object that defines an arc angle
α	the starting value
ω	the ending value

A ***layout*** $L = \langle D_1, D_2, ...D_n\rangle$ (Fig. 7) contains all devices residing on one or more ***pages*** and also defines environmental variables such as substrate type, composition, and size, temperature, humidity, and other variables as defined by the end user. Once a layout is completed, the framework renders the device using established file formats, such as PDF, DXF, and SVG.

IV. Physical Design Algorithm

Our approach to paper microfluidic physical design is to work radially outward from source fluid reservoirs to sink reservoirs while seeking to maintain the minimum distance that fluid must travel. Components are placed one-at-a-time. Potential locations for each new component are enumerated by a 360° sweep, motivated by the way that a radar screen displays information. At each potential location, the component may also be rotated 360° to best fit the component into the subset of the device layout that has been generated thus far.

In this manner, a listing of potential placement locations is sorted by how closely they abut the existing layout, while minimizing any desired parameters such as shortest critical path, fluid volume, time to complete, etc., as secondary criteria. A route is computed for each potential location, based on the premise that the closest positions are likely to have the shortest routes, although no such claim can be guaranteed in the general case. If a suitable route is found, the component is permanently placed and connected to the

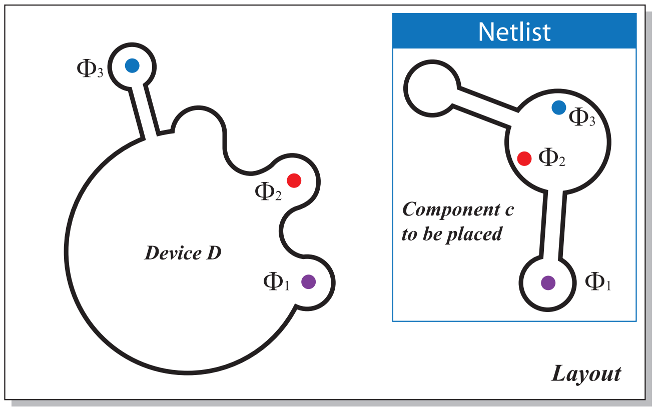

Fig. 7. The Radar algorithm generates a ***layout*** of one or more ***devices*** constructed from a ***netlist*** of ***components***. *Components are selected one-by-one for placement; after each component is placed, it is connected to the existing layout by **channels** that route fluid throughout the layout.*

layout. The algorithm terminates preemptively if all other candidate placement locations are not routable; it terminates successfully when successful placement locations are found for all components.

Algorithm 1 The Radar Place and Route Algorithm

```
1:  function RADAR($N$)
2:      $D \leftarrow$ new Device()
3:      $L \leftarrow$ new Layout($D$)
4:      while $c \leftarrow N$.pop()
5:          $Q_\rho \leftarrow$RADARPLACE($D \in L, c$)
6:          if $Q_\rho.empty()$
7:              return $L$
8:          else
9:              while $!Q_\rho.empty()$
10:                 $c.(x, y, r) \leftarrow Q_\rho.pop()$
11:                 if !RADARROUTE($D, c$)
12:                     return $L$
13:         MERGE($L, c, Chs$)
14:         $L.center \leftarrow Page.center$
15:         if $L.w > Page.width$ || $L.h > Page.height$
16:             return $L$
17:     return $L$
```

A. *The* Radar *Algorithm*

The Radar algorithm (Alg. 1, Fig. 7) takes as input a netlist of components and, optionally, a buffer value, which is the minimum allowable distance between components after placement. At the onset of the algorithm, the layout L is initialized with the environmental parameters determined by the user and an empty device to be constructed from the netlist. Table I lists and briefly describes all variables and data structures used.

The algorithm does not concern itself with the viability or the nature of the component during placement, but does check for whether or not the component is a source, sink, or internal, to determine whether or not routing needs to be performed. A source is a component with no input channels, and a sink is a component with no output channels; the netlist can have any number of source and sinks.

At the start of each pass, the next component $c \in N$ is popped from the netlist and attempted to be placed and routed. A priority queue Q_ρ (Table I) is initialized to hold the set of potential locations and orientations for c that RADARPLACE will attempt to generate when called. Q_ρ is ordered by increasing ***cost factor***, a value calculated for each candidate position that is calculated using a user-defined method. The default approach is to only consider total surface area or fluid volume but more complex methods can be considered, depending on the user's choice of optimization criteria. Should the list be empty, no suitable placement was found and the algorithm terminates. Otherwise, c is set to the calculated values of each candidate placement in Q_ρ until either a valid routing for c is discovered, or Q_ρ is emptied, resulting in a failed routing. The final placement and route of each component is then merged into L using the MERGE function for the next pass. The device layout is then centered in the page layout and evaluated to determine if extending beyond the page dimensions and returned if exceeded with an error. If failure occurs at any point (Fig. 13), or N is emptied and the algorithm completed, the finished or currently constructed layout L is returned and flagged with any appropriate errors.

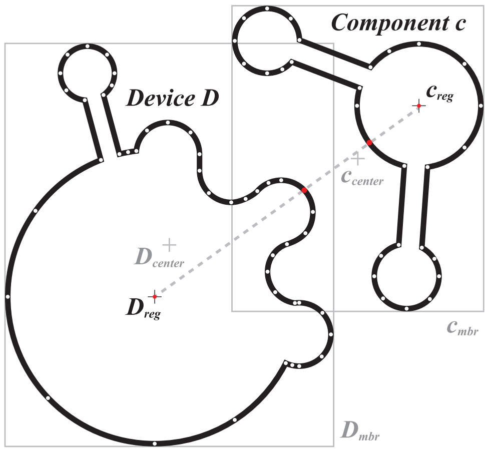

Fig. 8. A component c whose placement and orientation will be determined relative to a device D comprising one or more previously-placed components. The objective is to minimize the sum of the distances between each pair of critical points in c and D, which also aligning with either the center points or the registration points of each object as chosen by the user. The two center points c_{center} and D_{center} are defined as the mid-point vertically and horizontally of the minimum bounding rectangles, c_{mbr} and D_{mbr}, respectively. The registration points c_{reg} and D_{reg} are defined as local origin points $(0,0)$ for which all measurements within D are calculated and also where the location of D in L is determined. In this example, c and D have 29 and 44 critical points, respectively, resulting in 1276 distance measures for ***each*** candidate location.

B. The RADARPLACE *Algorithm*

RADARPLACE (Alg. 2) generates an ordered positional and orientation queue Q_ρ for evaluation of potential placements for component c in layout L sorted on COSTFUNCTION results. The sets P_D and P_c contain tuples of critical points of the Bézier paths for $D \in L$ and c respectively. The ***aperture*** $\Theta\langle\theta_\alpha, \theta_\omega\rangle$ is defined to be the start and end angles to be checked for either D or c and is initialized to a full 360° sweep. P_D obtains the results of scanning D performing a full sweep of its border using an algorithm called CURVESCAN (Alg. 3 and Section IV-B1). CURVESCAN returns the set of critical points that c will be compared against during the **for** loop spanning lines 7-25 of Alg. 2 and depicted in Fig. 8.

Algorithm 2 The RADAR Place Algorithm

```
 1: function RadarPlace($D, c$)
 2:     $Q_\rho \leftarrow \emptyset$
 3:     $\Theta\langle\theta_\alpha, \theta_\omega\rangle \leftarrow (0°, 360°)$
 4:     $P_D \leftarrow$ CurveScan($D_{outline}, \Theta$)
 5:     for $p_i \in P_D$
 6:         $c.(x, y) \leftarrow p_i.(x, y)$
 7:         $c.moveby$(Collide($D, c$))
 8:         if $c.connected$
 9:             $\Phi_D \leftarrow D.ports.at(\lceil D.ports.size/2\rceil)$
10:             $\Phi_c \leftarrow c.ports.at(\lceil c.ports.size/2\rceil)$
11:             $c.rotate$(GetPortAngle($\Phi_D, \Phi c$))
12:             $c.moveby$(Collide($D, c$))
13:             $\Theta\langle\theta_\alpha, \theta_\omega\rangle \leftarrow$ ViewWindow($D, c$)
14:         else
15:             $\Theta\langle\theta_\alpha, \theta_\omega\rangle \leftarrow$ ViewWindow($D, c$)
16:         $P_c \leftarrow$ CurveScan($c_{outline}, \Theta$)
17:         for $p_j \in P_c$
18:             $\theta_\tau \leftarrow$ GetAngle($D.center, c.center$)
19:             $\theta_\sigma \leftarrow$ GetAngle($p_i.(x, y), c.center$)
20:             $c.rotate((\theta_\sigma - \theta_\tau), c.center)$
21:             $c.moveby$(Collide($D, c$))
22:             if Collide($D, c$) == 0
23:                 $Q_\rho.add(\langle c.(x, y), c.rotation,$ Cost(D,c)$\rangle$
    )
24:     return $Q_\rho$
```

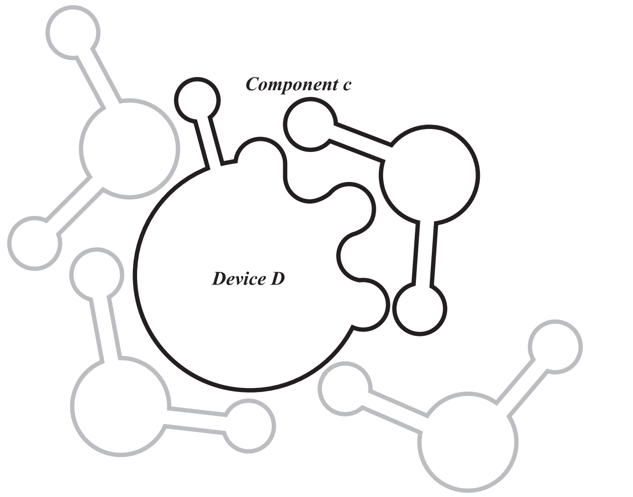

Fig. 9. When there are no ports to be connected, RADARPLACE rotates the component c up to 360° for each candidate location to compare all critical points of c to all critical points of placed devices $D \in L$.

1) CURVESCAN*:* The CURVESCAN algorithm (Alg. 3) generates a sub-curve comprising a set of points contained in a Bézier path O bounded radially by a start/ending angle Θ, and

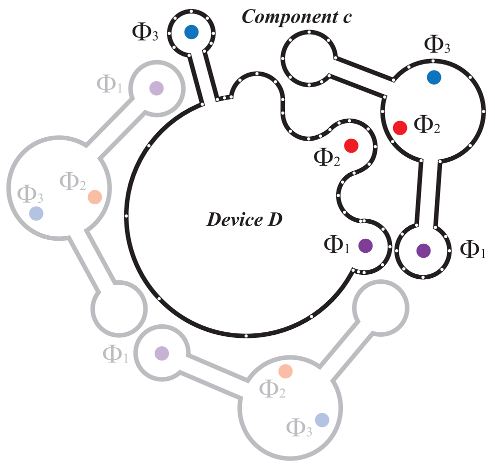

Fig. 10. When there are ports to be routed, RADARPLACE restricts the rotation of c to ensure that the ports are oriented toward the points in D to which they connect.

returns the sub-path outline of the object bounded by those angles for analysis, as illustrated in Fig. 11. Each segment s within the sub-path is examined to determine if it falls within the aperture Θ, which is defined as the angle formed between the edges that are reachable from the source object to destination object, similar to the visible face of the moon when viewed from earth. If so, the curve and any connecting curves are added to sub-path P if not already present in P. CURVESCAN increments the angle by the $radianInterval$ formed by the last curve added minus the current angle and terminates after examining all segments in O and returns P within the aperture Θ.

Algorithm 3 The CURVESCAN Algorithm

```
1: function CURVESCAN($O, \Theta_{\alpha,\omega}$)
2:     $P \leftarrow \emptyset$
3:     if $\Theta_\alpha$ undefined
4:         $\Theta_\alpha \leftarrow 0$
5:     if $\Theta_\omega$ undefined
6:         $\Theta_\omega \leftarrow 360$
7:     while $\Theta_\alpha < \Theta_\omega$
8:         $line \leftarrow (O.center(x,y), \Theta_\alpha)$
9:         $I \leftarrow$ GETINTERSECTIONS($O, line$)
10:        for $i \in I$
11:            $s_{temp} \leftarrow s_i$ intersected farthest from $O.center$
12:            if ( $!s_{temp} \in P$ )
13:                $P$.add($s_{temp}$)
14:                if ( $!( (s \in O_{i-1}) \in P )$ )
15:                    $P$.add($s \in O_{i-1}$ to $s_{temp}$ )
16:        $\Theta_\alpha \leftarrow \Theta_\alpha + radianInterval$
17:    return $P$
```

2) Placing Components: At the start of each iteration (Line 7 of Alg. 2), c is initialized so that it is placed such that the center point of c is set to the same coordinates as critical point p_i to initialize the location of c to have some overlap with the already placed items in L. The COLLIDE function computes the amount of overlap between c and D. If the amount of

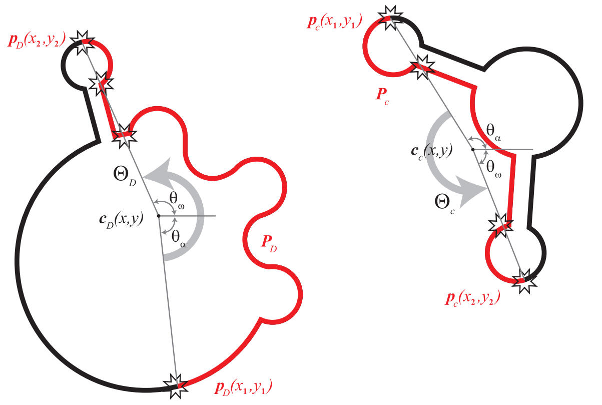

Fig. 11. CURVESCAN retrieves a sub-path P_D from a device's outline or P_c from a component's outline based on a start and end angle $(\theta_\alpha, \theta_\omega)$, which were determined by VIEWWINDOW (Alg. 3 and Fig. 12). The start and end points are calculated in standard polar orientation in a counter-clockwise manner with the rotational origin at the center point of the outline and 0° aligning with the standard x-axis orientation. The right component's sub-curve is determined by the furthest point intersecting with the start angle $P_c(x_1, y_1)$ (e.g. 110°) and end angle $P_c(x_2, y_2)$ (e.g. 270°). The left sub-curve begins earlier in the rotation (e.g. 290°) but is internally converted to a negative value (e.g. -70°) to return the proper sub-curve from points $P_D(x_1, y_1)$ to $P_D(x_2, y_2)$.

overlap is non-zero, then this information is used to move c to eliminate overlap currently in L. For example, should COLLIDE return $\{15.0, -5.0\}$, c would be moved 15 units to the right along the x-axis, and 5 units down the y-axis.

Depending on whether c is connected to the current layout or not – meaning a new source – the algorithm then calculates the aperture of points to be considered from D and c when calculating the distance between each candidate location to place c to the closest non-overlapping object in L. If c has no ports to connect (Fig. 9), all points of c are compared against all points of all devices in L. The base case assumes c is connected and therefore retrieves the middle connecting port pair(s) from D and c based on their indexed location in an internal $ports$ array in c. Port specification is determined by the user prior to algorithm execution and therefore the order of indexing is fixed. The component c is rotated to align the source in D to the sink in c for the purpose of minimizing channel crossing in the routing phase, as shown in Fig. 10 By aligning the middle ports, the most direct ports would be connected initially and then connected to each side in turn. In each case, VIEWWINDOW is called to obtain the rotational angle of arc aperture (Table I) for both D and c and is assigned to Θ for passing into CURVESCAN (Alg. 3). CURVESCAN then returns the points P_c on c that are to be considered in the inner **for** loop.

3) Determining Aperture with VIEWWINDOW*:* It is necessary to calculate the viewing aperture from the bounds of one component to the bounds of another component determined from their respective centers as shown in Fig. 12. VIEWWINDOW takes in two objects – devices or components – and begins by identifying the angle θ_τ between their centers $c_\sigma.(x.y)$ and $c_\tau.(x.y)$. θ_λ and θ_ρ are the perpendicular angles to that angle which are used to find the outermost intersections points to find the start and end points $p_\sigma.(x, y)$ and $p_\tau.(x, y)$

of the profile on each component. The angle formed between those points for each component $\Theta.\alpha$ and $\Theta.\omega$ is then returned in the aperture Θ.

Algorithm 4 The VIEWWINDOW Algorithm

```
1: function VIEWWINDOW($c_\sigma, c_\tau$)
2:     $\Theta.(\alpha, \omega) \leftarrow (0, 0)$
3:     $\theta_\tau \leftarrow$ GETANGLE($c_\sigma.(x, y)$, $c_\tau.(x, y)$)
4:     $\theta_\lambda \leftarrow \theta_\tau + 90°$
5:     $\theta_\rho \leftarrow \theta_\tau - 90°$
6:     $p_\sigma.(x_1, y_1) \leftarrow$ INTERSECT($c_\sigma$, $\theta_\lambda$)
7:     $p_\tau.(x_1, y_1) \leftarrow$ INTERSECT($cC_\tau$, $\theta_\lambda$)
8:     $\Theta.\alpha \leftarrow$ GETANGLE($p_\sigma.(x_1, y_1)$,$p_\tau.(x_1, y_1)$)
9:     $p_\sigma.(x_2, y_2) \leftarrow$ INTERSECT($c_\sigma$, $\theta_\rho$)
10:    $p_\tau.(x_2, y_2) \leftarrow$ INTERSECT($c_\tau$, $\theta_\rho$)
11:    $\Theta.\omega \leftarrow$ GETANGLE($p_\sigma.(x_2, y_2)$,$p_\tau.(x_2, y_2)$)
12:    return $\Theta$
```

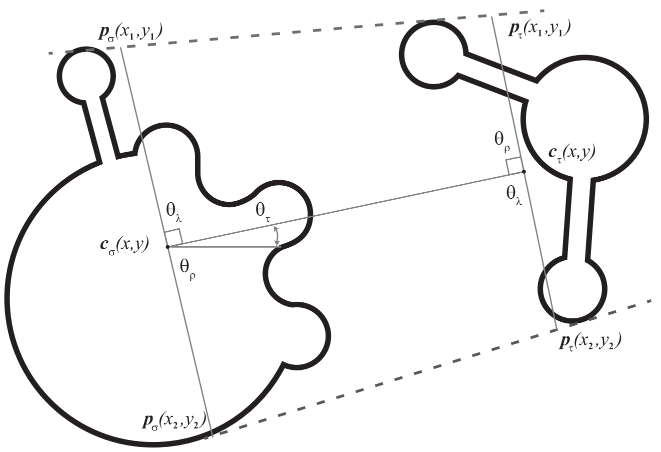

Fig. 12. VIEWWINDOW identifies the profiles of the devices D in layout L that would face component c when drawing lines from the outer extrema of each device. This determines the portion of the objects' profiles that need to be considered when calculating the closeness of the objects to one another. Devices and components do not have to be connected to anything currently in the layout.

4) Comparing Critical Points: For each point pair, c is rotated to align p_j and p_i and their centers of rotation (Fig. 8). The rotation angle is the difference between the angles formed from the line connecting the centers of the place devices in L and c, as well as the line connecting p_i and the center of c. c is then moved so that its center point is equal to the current critical point p_i, which ensures that overlap will occur. Next, c is shifted by the amount of overlap to a location expected to be outside the bounds of all devices in L. Should the new location also result in an overlap as shown in Fig. 13, then the location is discarded and the loop continues. If there is no overlap, the location, orientation, and cost factor of the location is added to the priority queue. Once all desired points have been evaluated, Q_ρ is returned.

5) Evaluating the Cost of Potential Locations: The placement phase of the algorithm requires a method of determining what is "best" when identifying potential placements for components. The COST function in Alg. 2 is an abstracted method call that returns a numeric value which determines its rank in

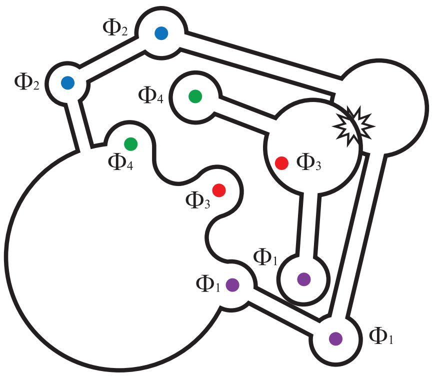

Fig. 13. In this example, RADARPLACE is unable to place c without colliding with already placed components in L and the candidate location is discarded from consideration.

the priority queue which sorts the candidate placements by that rank. Paper microfluidic devices have several mitigating factors that affect performance of the fluid in the substrate and depending on the specific application the end user is targeting, the relative importance of each of these factor may alter what constitutes an optimal and/or effective device. Several mathematical models exist [38] that characterize various physical properties that an end user may want to employ in determining a cost factor for a component location. The Lucas-Washburn model (eq. 1) applies to one-dimensional flow and calculates length of travel of a fluid over a particular time – which is useful when seeking a method to minimize channel lengths

$$l = k\sqrt{\frac{\sigma}{\mu}t} \tag{1}$$

where "k is a proportional constant, σ the surface tension and μ the viscosity of liquid, and t time. The proportional constant k depends on the material properties of the porous medium including the pore diameter, contact angle between the liquid and porous medium, and tortuosity of the porous medium." [39] Alternatively, another useful model [7] that accounts for capillary action and the fluid interaction with channel boundaries might be

$$l(t) = \alpha\sqrt{\left(1 + \beta\frac{d}{\phi^{\frac{1}{3}}w}\frac{\cos\theta_b}{\cos\theta}\right)\frac{\sigma}{\mu}t} \tag{2}$$

where α is an empirical co-efficient based on experimental results, β is a correction co-efficient, W is the channel width, θ_b is the contact angle with the barriers, θ is the contact angle with the substrate, ϕ represents the porosity value of the substrate, σ the vapor-liquid interfacial tension, and d the diameter of the capillary tubes in the medium. The dynamics of modelling and optimizing cost factors in a paper microfluidic setting can greatly increase the algorithmic complexity and computational overhead. While, the cost functions can be determined by the paper microfluidic device designer based on the desired metrics for optimization, the default approach, which is presented here, utilizes the paper area, and, by extension fluid volume, as the primary objective. Specifically, our approach tries to minimize the distance between placed components, and calculates how close they are to one another, as discussed in the following subsection.

Algorithm 5 The HOWSNUG Algorithm

```
1: function HOWSNUG($D, c$)
2:     $\Theta \leftarrow$ VIEWWINDOW($D, c$)
3:     $P_D \leftarrow$ CURVESCAN($D, \Theta$)
4:     $P_c \leftarrow$ CURVESCAN($c, \Theta$)
5:     $(x_2, y_2) =$ MAX($P_D.p_1, P_D.p_n, P_c.p_1, P_c.p_n$)
6:     $(x_1, y_1) =$ MIN($P_D.p_1, P_D.p_n, P_c.p_1, P_c.p_n$)
7:     $Box.area = (x_2 - x_1) * (y_2 - y_1)$
8:     $P_D.area \leftarrow$ BÉZIERAREA($P_D, box.(x_1, y_1)$)
9:     $P_c.area \leftarrow$ BÉZIERAREA($P_c, box.(x_2, y_2)$)
10:    $SnugFactor \leftarrow (Box.area - P_D.area - P_c.area)$
11:    return $SnugFactor$
```

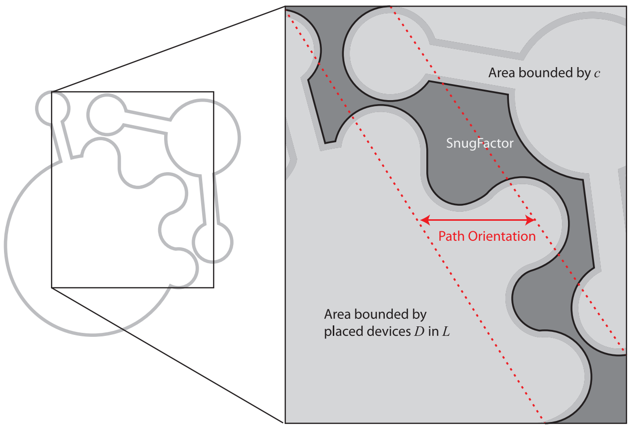

Fig. 14. The snugness factor of two curves, as computed by HOWSNUG. The snugness factor is defined to be the area between the two curves within a bounding box. The path orientation (in red) cannot be vertical; if this occurs, both paths are rotated 90° prior to computing the snugness factor.

6) Quantifying "Closeness" with the SNUGNESS FACTOR*:* The concept of snugness when evaluating relative placement of components is defined as the minimal amount of space between 2 objects. The HOWSNUG algorithm (Alg. 5 takes in two Bézier paths and calculates the area between them which represents the concept of snugness between components. Using the components passed in, VIEWWINDOW is called to determine the aperture of exposure between the components which is then passed into CURVESCAN which returns the sub-path from the outline of each component which forms the paths where the area between is calculated.

Figure 14 illustrates computation of the snugness factor. The first step is to generate the smallest bounding box that contains both curves. The start and end points of both sub-curves P_D and P_c are compared to find the maximum and minimum points of the bounding box and the $Box.area$ is calculated using these points. Second, the orientation of the curves is determined; if the orientation is vertical, the bounding box and curves are rotated 90°. Third, the area under both curves, but within the bounding box, is calculated using a standard polygon decomposition method [40]; let $P_D.area$ and $P_c.area$ denote these areas. Then the snugness factor is computed as $Box.area - P_D.area - P_c.area$.

7) MINMAXDELTA*:* Designers of real-world paper microfluidic devices often need to *minimize* some values while *maximizing* others. For example, consider a home pregnancy test, probably the most common example of paper microfluidics. These tests consist of a strip of paper; urine is applied to one end, and capillary action transports the urine past two or more test lines to an absorbent pad on the other end of the strip. The test lines change color when exposed to human chorionic gonadotropin (HCG), a protein present in the urine of pregnant women. It is advantageous to *maximize* the amount of urine that passes through the test lines, because more urine means more HCG detected (and therefore a more-pronounced color change and an easier-to-read test result). Designers accomplish this by maximizing the size of the absorbent pad, which functions as a pump to drive urine flow past the test lines. It is also advantageous to *maximize* (up to a practical limit) the run time of the assay (the amount of time spent flowing urine past the lines). Simultaneously, it is advantageous to *minimize* the size of the paper strip, since it only serves to conduct urine through the test lines, and minimizing overall device size reduces costs associated with device fabrication, packaging, and shipping.

The function MINMAXDELTA supports maximization and minimization optimizations like these. The function computes a running tally of the minimum and maximum differences in value for each dimension among a set of dimensions passed in as a parameter. It returns the appropriate minimum or maximum value in either the positive or negative directions along each dimension. For example, assume we have a running tally of surface area (that we want to minimize) and runtime (that we want to maximize) $S_\triangle = \{10.0_{min}, 15_{max}\}$. The algorithm has determined there is a potential placement with a value of $T_\triangle = \{9.0_{min}, 10_{max}\}$ and when fed into MINMAXDELTA the resulting values are $S_\triangle = \{9.0_{min}, 15_{max}\}$. Alternatively if $T_\triangle = \{21_{min}, 10_{max}\}$, then $S_\triangle$ would not change; or if $T_\triangle = \{5_{min}, 20_{max}\}$ then we would have $S_\triangle = \{5_{min}, 20_{max}\}$.

C. RADAR *Route*

RADARROUTE (Alg. 6) is invoked when a candidate location has been identified for a component c and at least one of c's ports has been matched with corresponding port on the device D in the current layout L. RADARROUTE attempts to route one or more channels to deliver fluid from D to c. The number of port pairs between c and D determines the number of routes required.

The algorithm initializes sets to hold Channels Chs constructed during the algorithm and the intersections of the probes I (Line 2). The $buffer$ value is set to $1/2$ the resultant width of the channel to be constructed (Line 3). In Line 4, the number of port pairs is determined to be odd or even and then the middle index value of the port pairs is chosen (Line 5) and is used as the initial iterative values for i and j (Line 6). Finally, if the port pair count is even, j is incremented to the next port pair index value (Line 7). The example in Fig. 15 shows three port pairs $\{\Phi_1, \Phi_2, \Phi_3\}$ to be connected by three channels. The port pairs are ordered for routing from the middle toward the two perimeter of the device. As shown in Fig. 16, the routing order is $\{\Phi_2, \Phi_1, \Phi_3\}$.

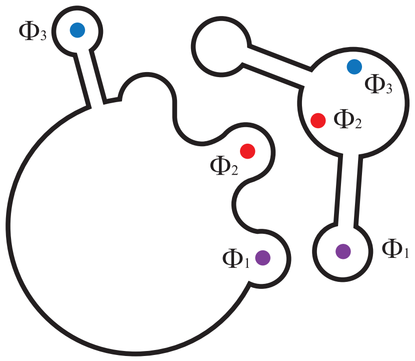

Fig. 15. RadarRoute starts with component c already placed and 3 port pairs to be connected: Φ_1, Φ_2, and Φ_3. Connections are routing starting with the middle port pair Φ_2, followed by Φ_1 and Φ_3 to reduce the likelihood that routes cross.

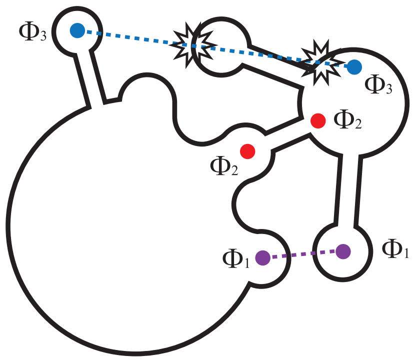

Fig. 16. The probe generated for port pair Φ_2 is unobstructed, so the corresponding channel is routed. Port pair Φ_3's probe collides with the outline at two distinct collision points.

While there are channels to be routed, the algorithm pulls the port coordinates pairs (Fig. 15: $\{\Phi_1, \Phi_2, \Phi_3\}$) that will serve as the abstraction that will attempt to connect the two locations on D and c (Lines 9-12). A Bézier path for each connection is instantiated to serve as the "probe" connecting the source and sink locations (Lines 13-14). If the probe intersects with c (Fig. 16) (Line 16), the intersection points are added to a set along with the points that make up the sub-curve of c between them (Lines 17-19). The *precurve* (Line 20) is sub-curve of the start of the *probe* to the point where it intersects c and is used to determine if the probe intersects with itself.

The points in the set *curve* are then each evaluated by creating a new node in the *probe* (Line 22) corresponding to the original node in c and moved away from c by half of the width of the channel component that will be generated (Fig. 17, Line 23). In this way, the *probe* will produce a curve running parallel to the previously intersected sub-curve of c. Based on the current position of c with respect to L, c may be moved to a distance away from L to allow for the width of the channel to be routed (Fig. 18) if c collides with any already-placed components. The sub-curve is also tested for intersection with the *precurve* (Lines 24-25); a positive

Algorithm 6 The Radar Route Algorithm

```
1:  function RadarRoute($D, c$)
2:      $Chs \leftarrow \emptyset$, $I \leftarrow \emptyset$
3:      $buffer \leftarrow channel.width/2$
4:      $odd \leftarrow$ isOdd($c.inputs.size$)
5:      $midpoint \leftarrow \lceil c.inputs.size/2 \rceil$
6:      $i, j \leftarrow midpoint$
7:      $(odd)\ ?\ j \leftarrow i : j \leftarrow i+1$
8:      while $i >= 0 \parallel j < c.inputs.size$
9:          $p_{D_i}.(x,y) \leftarrow D.\Phi_i.(x,y)$
10:         $p_{D_j}.(x,y) \leftarrow D.\Phi_j.(x,y)$
11:         $p_{c_i}.(x,y) \leftarrow c.\Phi_i.(x,y)$
12:         $p_{c_j}.(x,y) \leftarrow c.\Phi_j.(x,y)$
13:         $probe_i \leftarrow$ new Path($p_{D_i}(x,y), p_{c_i}(x,y)$)
14:         $probe_j \leftarrow$ new Path($p_{D_j}(x,y), p_{c_j}(x,y)$)
15:         for each $probe_{i,j}$
16:             if $I \leftarrow$ Intersect($c, probe$)
17:                 $probe.addNode(I.node_\alpha)$
18:                 $probe.addNode(I.node_\omega)$
19:                 $curve \leftarrow$ GetSubCurve($c, I$)
20:                 $precurve \leftarrow$ GetSubCurve($probe, I_\alpha$)
21:                 for each $node \in curve$
22:                     $probe.addNode(node)$
23:                     Move($probe.node, buffer$)
24:                     if Intersect($curve, precurve$)
25:                         return $Chs$
26:             if HowSnug($D, c$) $< channel.width$
27:                 $c.moveby(channel.width)$
28:         if $odd$ && $i == midpoint$
29:             if $probe_i.length < probe_j.length$
30:                 $Chs.add($ new Channel($probe_i$) )
31:             else
32:                 $Chs.add($ new Channel($probe_j$) )
33:             $j \leftarrow i$
34:         else
35:             $Chs.add($ new Channel($probe_i$) )
36:             $Chs.add($ new Channel($probe_j$) )
37:         Merge($c.outline, Chs$)
38:         $i \leftarrow i-1$
39:         $j \leftarrow j+1$
40:     return $Chs$
```

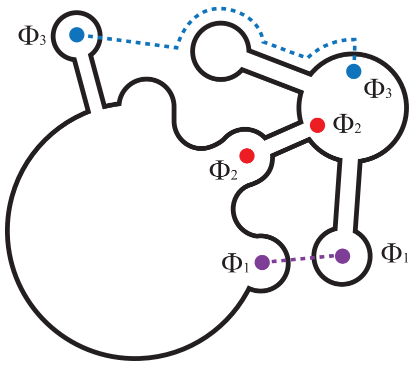

Fig. 17. A sub-path is routed around the collision ports arising from port pair Φ_3's probe. The sub-path follows the contour of the new component that is added to the layout.

answer here would indicate that the probe under construction intersects with itself (Fig. 19); if this occurs, the probe cannot complete the route and the algorithm terminates. On Line 26, a quick check of the placement of c's snug factor will determine if c needs to be moved away from D by the width of the channel before channel construction begins.

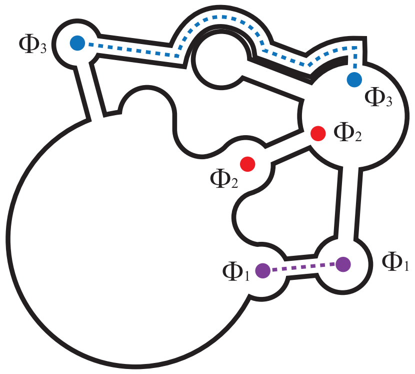

Fig. 18. The routed probe is adjusted to an offset equal to half of the width of the channel being routed, plus any desired buffer distance between barriers of distinct channels. This routes the channel to connect port pair Φ_3. Port pair Φ_1 is also routed trivially.

The algorithm then handles the initial case of routing: where the starting channel is either the middle value of an odd number of channels or if there is an even number. The concern only occurs on the first route as all subsequent routings are performed as left /right pairs. Therefore in the initial case where the number of channels is odd, 2 probes are run with the same source and sink but are deflected each to the left and right (Lines 28-36). At completion of probe generation, the probe length is evaluated and the shorter probe is used to generate the Channel.

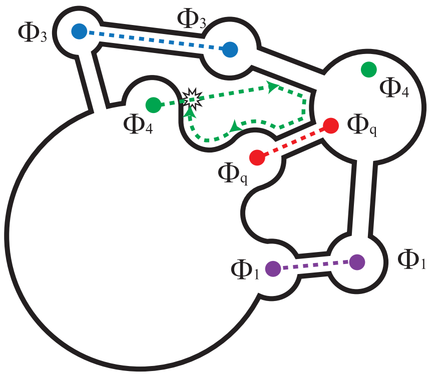

Fig. 19. Example of a probe failure. In this case, in attempting to connect port pair Φ_4, the probe loops back and collides with itself indicating the probe is trapped by already placed components.

Once the probe is generated, a Channel component is instantiated for the probe and merged (Lines 37) into c using standard Bézier intersection and merging algorithms [41] where it will become part of the outline for the next probes to avoid. If the probe cannot be deflected around a routed channel, the routing fails and what was successfully routed is returned along with an error (Fig. 20). The iterating values of i and j are incremented (Lines 38-39) and the next pairs are evaluate If the ports are all successfully routed, then Chs is returned without error (Fig. 21, Lines 40).

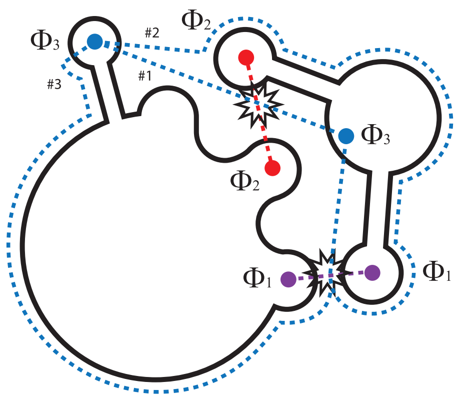

Fig. 20. Example of a routing failure. In this case, it is not possible to find a route that connects port pair Φ_3 that doesn't cross the routes for at least one of port pairs Φ_1 and Φ_2.

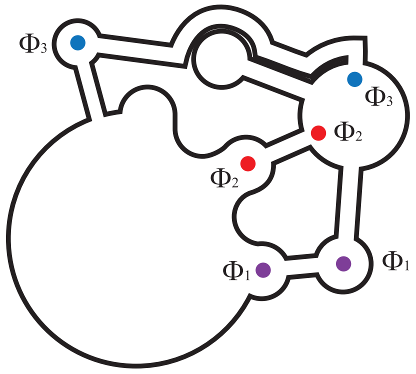

Fig. 21. A successfully routed layout. The newly introduced component and routed channels are integrated into the device. The algorithm is ready to place-and-route the next component.

V. Experimental Results

A. Methodology

As the microfluidic space is non-discrete, the DICE[18] algorithm was chosen as an appropriate benchmark to compare the RADAR approach. Although DICE is built to a grid, the units are based on the physical dimensions of the components with a user-specified buffer between them. Components are placed down and to the right of placed components yielding a diagonal placement that allows for routing to be performed in a mostly-linear fashion. Once routed, the layout is then rotated 45° to reduce it's footprint for fabrication

Testing harnesses were built to generate tests for area utilization. Three test cases were developed for comparing DiCE (Fig. 22a,c,e) and RADAR (Fig. 22b,d,f): a "chain" of components (Fig. 22a,b) that are directly connected one time from source to sink for each component, an "orbital" approach (Fig. 25c,d) where a single source is connected to any number of sinks surrounding it, and a "tree" (Fig. 26e,f) where each component is connected to 2 sinks. All tests utilize a directed acyclic graph for maintaining connections thus protecting against loops, however both DICE and RADAR

TABLE II
RUNTIMES FOR EACH DiCE AND RADAR FOR EACH LAYOUT TEST, ALONG WITH THE NUMBER OF COMPONENTS PLACED AND ROUTED. Δ INDICATES THE DIFFERENCE IN RUNTIME. RADAR WAS SIGNIFICANTLY SLOWER IN ALL CASES; THIS IS TO BE EXPECTED AS RADAR IS EXHAUSTIVE WHILE DiCE IS A HEURISTIC.

Test	Algorithm	#	Time	Δ
Chain	DiCE	4	0:00:04	
	RADAR	4	0:05:48	**0:05:44**
	DiCE	8	0:00:10	
	RADAR	8	1:25:54	**1:25:44**
Orbit	DiCE	5	0:00:05	
	RADAR	5	0:14:55	**0:14:50**
	DiCE	9	0:00:11	
	RADAR	9	1:16:11	**1:16:00**
Tree	DiCE	7	0:00:09	
	RADAR	7	0:39:39	**0:39:30**
	DiCE	15	0:00:27	
	RADAR	15	1:46:49	**1:46:22**

can handle cycles without indefinite loops. Several sets of component counts were chosen as initial tests to illustrate potential trends due to layout growth.

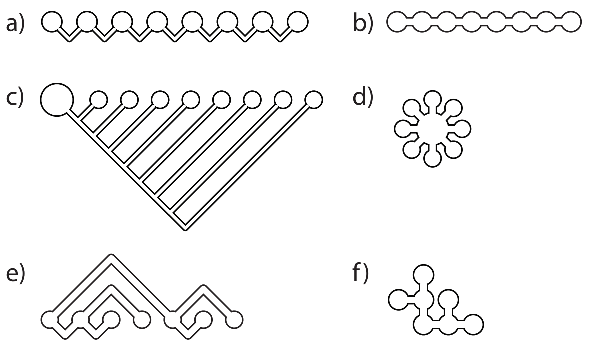

Fig. 22. Three test cases were developed to evaluate the quality of layouts produced by DICE [18] (left column) vs. RADAR (right column): a,b) depict the "Chain" test that connects each source to the next sink sequentially; c,d) shows the "Orbit" layout where several sinks are connected to a central source; and e,f) the "Tree" structure where each component is connected to two sinks.

Each layout was then output onto LabNerd® paper stock, reheated to sublimate the wax ink into the paper, and a PCR backing tape applied to isolate the layout from the work surface. Filtered water with several drops of green food coloring was pipetted into the source region of each layout. Initially 40 μL of fluid was delivered with additional fluid delivered in 20 μL steps if the device showed signs of drying out until the fluid movement reached all sinks in the layout, or the fluid ceased to travel any further due to the leading edge drying out forming a barrier to additional flow.

B. Algorithm Results

Results are plotted such that the vertical axis plots area usage measured in mm^2 on a logarithmic scale (Fig. 23). The horizontal axis shows the individual test results with the first letter being the algorithm (D)ICE or (R)adar; the second letter the test performed: (C)hain, (O)rbit, or (T)ree; and lastly, the number of components placed and routed.

In each test and for nearly every metric, RADAR outperformed DICE in area utilization. In terms of fluid area, RADAR would outperform DICE due to being able to more compactly place components thereby shortening the channel and therefore the fluid travel. The chain test allowed for a more level playing field due to reducing the amount of channel deflection occurring.

Notable however, both DICE and RADAR initially failed routing of all 15 components during the tree test which indicated a weakness in their approaches when the "greedy" criteria of lowest $snugnessfactor$ produces a layout to compact to route past the second level of the tree. It was determined that presenting the tree netlist in a breadth-first manner resulted in the algorithms' inability to place and route and when the netlists were traversed in a depth-first manner, both algorithms were able to successfully place and route the netlists.

RADAR is intended to explore the paper microfluidic device layout space as completely as possible. High execution times are expected, and this will inevitably limit scalability. As shown in Table II, RADAR runs several orders of magnitude slower than DiCE (a heuristic), due to the ever-expanding set of critical points that it enumerates. RADAR examines $\Theta(m \times n)$ critical points, per pass, as candidate locations for routing. While the routing phase can terminate in $\Omega(1)$ time if the first candidate is routed successfully, it is also possible that routing may fail for all candidates. Consequently, runtimes can become extremely large, for example, as RADAR took nearly 2 hours to complete route the 15-component "Tree" benchmark. Even the smaller case, a 4-component "Chain," required nearly six minutes to complete. Future work may examines strategies to reduce the runtime of RADAR, including techniques that limit the portion of the search space explored, as well as a parallel implementations of key bottlenecks.

C. Physical Device Performance Results

The physical devices output from generated layouts were tested for real-world performance by delivering fluid to each device's source region and monitored for either failure to complete or time to completion. Although DICE did successfully generate placed and routed layouts, not all devices were able to run to completion. Devices featuring long channels and distant sinks would fail due to the leading edge of the fluid drying out and forming a barrier to any further fluid travel – even when additional fluid delivered to the device. As noted in Table III, only two versions of the DICE layouts ran to completion even though additional fluid (amounts listed) was dispensed. By comparison each RADAR device was not only able to run to completion with only 20 μL of additional fluid required for the eight component chain and tree versions (Figs. 24 and 26 respectively). As indicated by the Δ column, devices laid out by RADAR devices have factors of improvement in the range of $3.5-113\times$ over those laid out by DiCE, while successfully running to completion in all cases.

VI. Conclusion and Future Work

The RADAR approach to placement and routing demonstrates that layouts that can be generated and optimized for metrics such as fluid travel and usage, device area, and total

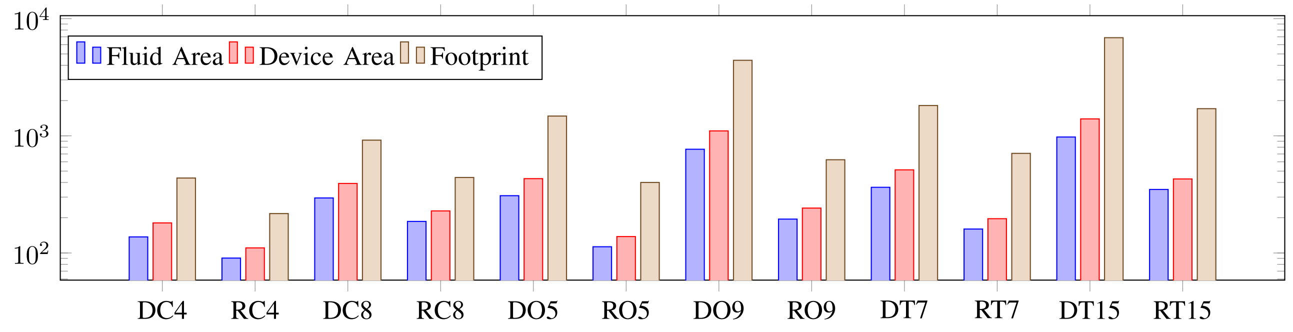

Fig. 23. Area utilization in terms of fluid space, device occupancy, and total area required for fabrication (in mm^2).

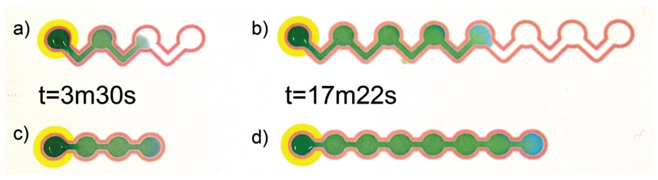

Fig. 24. The "Chain" layout consisting of a number of components connected in a series. DiCE layouts of a) 4 components and b) 8 components, RADAR versions are c) 4 and d) 8 components. Times listed refer to completion of devices for the RADAR versions and the state of the DiCE versions. Table III lists the completion and/or failure times for the DiCE versions.

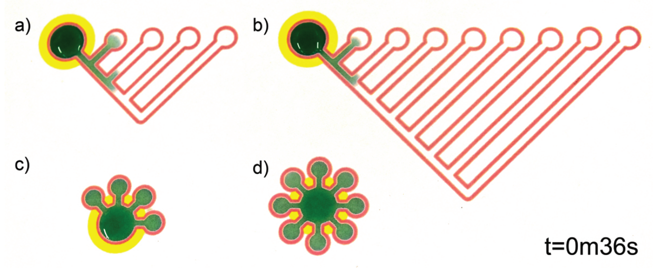

Fig. 25. The "Orbit" layout consisting of a single source feeding multiple sinks shows the importance of compact layouts. Layouts a-b) use DiCE, c-d) use RADAR.

materials required – all desirable goals to help meet the World Health Organization's ASSURED criteria for medical diagnostics. In the future, other placement selection criteria will be looked at, as well as comparing several successful routings that may help avoid fluid routing failures and further optimize paper microfluidic devices.

Acknowledgment

This work was supported by the National Science Foundation Division of Computing and Communication Foundations under grants 1423414 and 1536026, and the National Science Foundation Division of Industrial Innovation and Partnerships under grant 1640757.

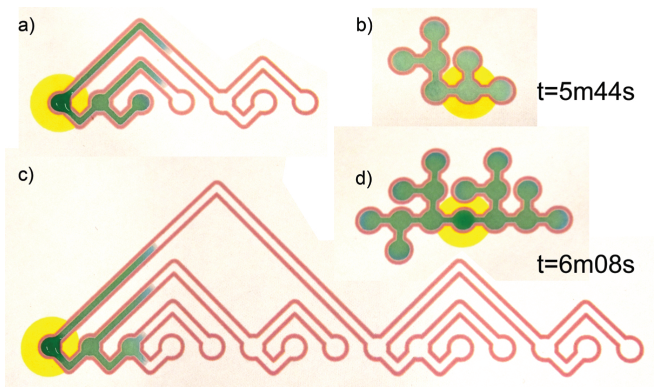

Fig. 26. The "Tree" layout consisting of each node feeding 2 nodes. a,c) DiCE, b,d) RADAR

TABLE III
Experimental results listing the which test, algorithm used, the number of components placed and routed, the volume of fluid delivered, and the time for the device to complete or the time of failure (in bold) due to the leading edge of the fluid drying out and blocking any further flow. Δ lists in red the additional time required for the devices laid out by DiCE to run to completion, compared to the devices laid out by RADAR.

Test	Algorithm	#	μL	Time (h:m:s)	Δ
Chain	DiCE	4	40.0	00:12:16	8:46
	RADAR	4	40.0	00:03:30	**3.5x**
	DiCE	8	120.0	**01:11:40**	54:18
	RADAR	8	60.0	00:17:22	**4x**
Orbit	DiCE	5	100.0	00:43:48	43:12
	RADAR	5	40.0	00:00:36	**73x**
	DiCE	9	200.0	**00:52:38**	52:10
	RADAR	9	40.0	00:00:28	**113x**
Tree	DiCE	7	120.0	**00:45:48**	40:04
	RADAR	7	40.0	00:0544	**8x**
	DiCE	15	140.0	**00:45:30**	39:22
	RADAR	15	60.0	00:06:08	**7x**

References

[1] C. E. Mills, J. M. Robins, and M. Lipsitch, "Transmissibility of 1918 pandemic influenza," *Nature*, vol. 432, no. 7019, pp. 904–906, 2004.

[2] C. Fraser, C. A. Donnelly, S. Cauchemez, W. P. Hanage, M. D. Van Kerkhove, T. D. Hollingsworth, J. Griffin, R. F. Baggaley, H. E. Jenkins, E. J. Lyons *et al.*, "Pandemic potential of a strain of influenza a (h1n1): early findings," *science*, vol. 324, no. 5934, pp. 1557–1561, 2009.

[3] D. Mabey, R. W. Peeling, A. Ustianowski, and M. D. Perkins, "Tropical infectious diseases - diagnostics for the developing world," *Nature Reviews Microbiology*, vol. 2, no. 3, pp. 231–240, 2004.

[4] M. T. Osterholm, "Preparing for the next pandemic," *New England Journal of Medicine*, vol. 352, no. 18, pp. 1839–1842, 2005.

[5] N. M. Ferguson, D. A. Cummings, C. Fraser, J. C. Cajka, P. C. Cooley, and D. S. Burke, "Strategies for mitigating an influenza pandemic," *Nature*, vol. 442, no. 7101, pp. 448–452, 2006.

[6] E. Bendavid, B. Mulaney, N. Sood, S. Shah, E. Ling, R. Bromley-Dulfano, C. Lai, Z. Weissberg, R. Saavedra-Walker, J. Tedrow, D. Tversky, A. Bogan, T. Kupiec, D. Eichner, R. Gupta, J. Ioannidis, and J. Bhattacharya, "Covid-19 antibody seroprevalence in santa clara county, california," *medRxiv*, 2020. [Online]. Available: https://www.medrxiv.org/content/early/2020/04/30/2020.04.14.20062463

[7] S. Hong and W. Kim, "Dynamics of water imbibition through paper channels with wax boundaries," *Microfluidics and Nanofluidics*, vol. 19, no. 4, pp. 845–853, 2015.

[8] E. Elizalde, R. Urteaga, and C. L. Berli, "Precise capillary flow for paper-based viscometry," *Microfluidics and Nanofluidics*, vol. 20, no. 10, p. 135, 2016.

[9] B. M. Cummins, R. Chinthapatla, F. S. Ligler, and G. M. Walker, "Time-dependent model for fluid flow in porous materials with multiple pore sizes," *Analytical Chemistry*, vol. 89, no. 8, pp. 4377–4381, 2017.

[10] M. G. Pollack, A. D. Shenderov, and R. B. Fair, "Electrowetting-based actuation of droplets for integrated microfluidics," *Lab Chip*, vol. 2, pp. 96–101, 2002. [Online]. Available: http://dx.doi.org/10.1039/B110474H

[11] M. A. Unger, H.-P. Chou, T. Thorsen, A. Scherer, and S. R. Quake, "Monolithic microfabricated valves and pumps by multilayer soft lithography," *Science*, vol. 288, no. 5463, pp. 113–116, 2000. [Online]. Available: http://science.sciencemag.org/content/288/5463/113

[12] E. Carrilho, A. W. Martinez, and G. M. Whitesides, "Understanding wax printing: a simple micropatterning process for paper-based microfluidics," *Analytical chemistry*, vol. 81, no. 16, pp. 7091–7095, 2009.

[13] C. Renault, J. Koehne, A. J. Ricco, and R. M. Crooks, "Three-dimensional wax patterning of paper fluidic devices," *Langmuir*, vol. 30, no. 23, pp. 7030–7036, 2014.

[14] W. Dungchai, O. Chailapakul, and C. S. Henry, "A low-cost, simple, and rapid fabrication method for paper-based microfluidics using wax screen-printing," *Analyst*, vol. 136, no. 1, pp. 77–82, 2011.

[15] J. Potter, P. Brisk, and W. H. Grover, "Using printer ink color to control the behavior of paper microfluidics," *Lab on a Chip*, vol. 19, no. 11, pp. 2000–2008, 2019.

[16] S. C. Terry, J. H. Jerman, and J. B. Angell, "A gas chromatographic air analyzer fabricated on a silicon wafer," *IEEE Transactions on Electron Devices*, vol. 26, no. 12, pp. 1880–1886, 1979.

[17] A. Manz, N. Graber, and H. Widmer, "Miniaturized total chemical analysis systems: A novel concept for chemical sensing," *Sensors and Actuators B: Chemical*, vol. 1, no. 1, pp. 244 – 248, 1990. [Online]. Available: http://www.sciencedirect.com/science/article/pii/092540059080209I

[18] B. Crites, K. Kong, and P. Brisk, "Diagonal component expansion for flow-layer placement of flow-based microfluidic biochips," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 16, no. 5s, p. 126, 2017.

[19] T. Tseng, M. Li, D. N. Freitas, A. Mongersun, I. E. Araci, T. Ho, and U. Schlichtmann, "Columba S: a scalable co-layout design automation tool for microfluidic large-scale integration," in *Proceedings of the 55th Annual Design Automation Conference, DAC 2018, San Francisco, CA, USA, June 24-29, 2018*. ACM, 2018, pp. 163:1–163:6. [Online]. Available: https://doi.org/10.1145/3195970.3196011

[20] B. Crites, K. Kong, and P. Brisk, "Directed placement for mvlsi devices," *J. Emerg. Technol. Comput. Syst.*, vol. 16, no. 2, Dec. 2019. [Online]. Available: https://doi.org/10.1145/3369585

[21] A. T. Jafry, H. Lim, S. I. Kang, J. W. Suk, and J. Lee, "A comparative study of paper-based microfluidic devices with respect to channel geometry," *Colloids and Surfaces A: Physicochemical and Engineering Aspects*, vol. 492, pp. 190–198, 2016.

[22] Z. Liu, J. Hu, Y. Zhao, Z. Qu, and F. Xu, "Experimental and numerical studies on liquid wicking into filter papers for paper-based diagnostics," *Applied Thermal Engineering*, vol. 88, pp. 280–287, 2015.

[23] T. Tseng, M. Li, B. Li, T. Ho, and U. Schlichtmann, "Columba: co-layout synthesis for continuous-flow microfluidic biochips," in *Proceedings of the 53rd Annual Design Automation Conference, DAC 2016, Austin, TX, USA, June 5-9, 2016*. ACM, 2016, pp. 147:1–147:6. [Online]. Available: https://doi.org/10.1145/2897937.2897997

[24] A. Grimmer, Q. Wang, H. Yao, T. Ho, and R. Wille, "Close-to-optimal placement and routing for continuous-flow microfluidic biochips," in *22nd Asia and South Pacific Design Automation Conference, ASP-DAC 2017, Chiba, Japan, January 16-19, 2017*. IEEE, 2017, pp. 530–535. [Online]. Available: https://doi.org/10.1109/ASPDAC.2017.7858377

[25] T. Tseng, M. Li, D. N. Freitas, T. McAuley, B. Li, T. Ho, I. E. Araci, and U. Schlichtmann, "Columba 2.0: A co-layout synthesis tool for continuous-flow microfluidic biochips," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 37, no. 8, pp. 1588–1601, 2018. [Online]. Available: https://doi.org/10.1109/TCAD.2017.2760628

[26] K. Yang, H. Yao, T. Ho, K. Xin, and Y. Cai, "AARF: any-angle routing for flow-based microfluidic biochips," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 37, no. 12, pp. 3042–3055, 2018. [Online]. Available: https://doi.org/10.1109/TCAD.2018.2789356

[27] P. Eades, "Drawing Free Trees," International Institute for Advanced Study of Social Information Science, Fujitsu Limited, Tech. Rep., 1991.

[28] G. Book and N. Keshary, "Radial Tree Graph Drawing Algorithm for Representing Large Hierarchies," University of Connecticut, Tech. Rep., 12 2001.

[29] K.-P. Yee, D. Fisher, R. Dhamija, and M. Hearst, "Animated exploration of dynamic graphs with radial layout," in *Proceedings of the IEEE Symposium on Information Visualization 2001 (INFOVIS'01)*, ser. INFOVIS '01. USA: IEEE Computer Society, 2001, p. 43.

[30] C. E. Leiserson, "Area-efficient graph layouts (for VLSI)," in *21st Annual Symposium on Foundations of Computer Science, Syracuse, New York, USA, 13-15 October 1980*. IEEE Computer Society, 1980, pp. 270–281. [Online]. Available: https://doi.org/10.1109/SFCS.1980.13

[31] S. A. Browning, "The Tree Machine: A Highly Concurrent Computing Environment, Computer Science Technical Reports 1980.3760," California Institute of Technology, Tech. Rep., 1980.

[32] J. D. Ullman, *Computational Aspects of VLSI*. USA: W. H. Freeman & Co., 1984.

[33] J. Burkis, "Clock tree synthesis for high performance asics," in *[1991] Proceedings Fourth Annual IEEE International ASIC Conference and Exhibit*, 1991, pp. P9–8/1.

[34] E. Di Giacomo, W. Didimo, and G. Liotta, "Radial drawings of graphs: Geometric constraints and trade-offs," *Journal of Discrete Algorithms*, vol. 6, no. 1, pp. 109 – 124, 2008, selected papers from AWOCA 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1570866707000159

[35] K. Mikami, "A computer program for optimal routing of printed circuit connectors," *IFIPS Proc., 1968*, 1968.

[36] D. W. Hightower, "A solution to line-routing problems on the continuous plane," in *Proceedings of the 6th annual Design Automation Conference*. ACM, 1969, pp. 1–24.

[37] J. Potter, W. H. Grover, and P. Brisk, "Design automation for paper microfluidics with passive flow substrates," in *Proceedings of the on Great Lakes Symposium on VLSI 2017, Banff,AB, Canada, May 10-12, 2017*, 2017, pp. 215–220. [Online]. Available: http://doi.acm.org/10.1145/3060403.3060476

[38] Z. Liu, X. He, J. Han, X. Zhang, F. Li, A. Li, Z. Qu, and F. Xu, "Liquid wicking behavior in paper-like materials: mathematical models and their emerging biomedical applications," *Microfluidics and Nanofluidics*, vol. 22, no. 11, p. 132, 2018.

[39] S. Gruener and P. Huber, "Imbibition in mesoporous silica: rheological concepts and experiments on water and a liquid crystal," *Journal of Physics: Condensed Matter*, vol. 23, no. 18, p. 184109, 2011.

[40] J. M. Keil, "Polygon decomposition," *Handbook of computational geometry*, vol. 2, pp. 491–518, 2000.

[41] T. W. Sederberg and T. Nishita, "Curve intersection using bézier clipping," *Computer-Aided Design*, vol. 22, no. 9, pp. 538–549, 1990.