

Efficient Droplet Routing Algorithm on Microfluidic Chips

Author: Jiwoo Sim

Cranbrook Kingswood High School

jwsim2019@gmail.com

DOI: 10.26821/IJSHRE.11.4.2023.110401

ABSTRACT

Digital microfluidic biochips (DMFBs) is one of the most actively researched technologies in the recent information technology and biotechnology fields. In particular, microfluidic chip technology is implemented to practice research upon various life phenomena, such as biomaterials, cells and tissues. However, to further optimize time-efficiency and performance quality, it is crucial to set effective pathway and initial position for each droplets within the limited area. In this study, a program is designed to calculate efficient droplet pathways for DMFBs by combining a heuristic-based algorithm and a genetic algorithm.

Since there are more moving obstacles on a microchip when droplets are manipulated simultaneously, it is more difficult to find the shortest route than conventional problems. It is vital to maintain a safe distance between the droplets as they will get merged when the distance between them becomes too close.

Therefore, the cost function was defined based on the distance from the start and end points, as well as the relative distance from the other droplets to help determine efficient pathways for droplets through the existing Dijkstra algorithm. In addition, based on the newly defined cost function, a genetic algorithm was used to find the most efficient initial location of each droplet. By randomly generating scenarios in the experiment, it was confirmed that the algorithm presented performs up to two times faster than the general method.

1. INTRODUCTION

There are many problems with the conventional procedures of conducting droplet experiments,

essential experiments to conduct blood tests, drug tests, etc. Due to the reliance on manual labor, droplet experiments are often error-prone; in fact, sixty percent of errors in wet labs come from human mistakes. In addition, due to the usage of pipettes when analyzing droplets, droplet experiments are also very harmful to the environment. Four million pounds of plastic are used to produce pipettes, and seventy percent of it goes to landfill, demonstrating the impact pipettes have on the environment.

With the improvement in nanotechnology, DMFBs (digital microfluidic biochips) have become much more prevalent in conducting these biological/chemical reactions in the form of droplets. Unlike the conventional procedures described above, DMFBs do not require pipettes and manual labor, making this solution much more environment-friendly and less error-prone.

However, currently, coming up with an efficient routing algorithm for droplets on DMFBs is difficult as it is hard to find the shortest path when there are many moving obstacles (other droplets). Also, in certain conditions, deadlocks can occur, which is when droplets are unable to move at all due to obstacles. On top of this, efficiency can be dramatically different with slightly different paths for the droplets. Therefore, developing an efficient routing algorithm for DMFBs requires us to take all these factors into account.

In this paper, we introduced a genetic droplet routing algorithm with a Dijkstra-based heuristic strategy to create an efficient algorithm for these DMFBs. Instead of encoding a complete path from the source to the target for each droplet, we considered an indirect representation in the genetic algorithm,

allowing simple permutation operators to be directly applied in the evolutionary process without any complex population initialization process. In the Dijkstra-based heuristic algorithm, a problem-specific cost function is introduced into the Dijkstra algorithm to find a more time-efficient path for each droplet. The experimental results demonstrate the superiority of the proposed method based on a synthetic benchmark scenario and a real-world bioassay benchmark scenario.

2. BACKGROUND

2.1 DMFB

A DMFB is composed of a two-dimensional electrode array and also consists of devices such as optical detectors, dispensing ports (where droplets are distributed from), waste reservoirs (where droplets are collected from), and control pins, creating an electric field that can be used to manipulate and move droplets. Electric fields can be used to maneuver the droplets as electric fields can influence the shape of droplets.

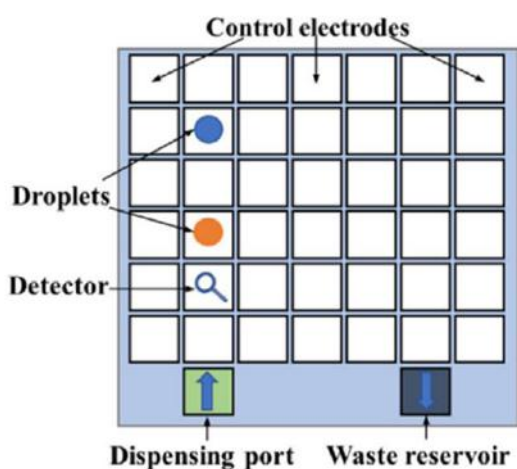


Figure 1. Model structure of DMFB

Therefore, by rapidly turning on and off specific squares in the electrode grid, droplets can be moved. The electrode array also has a hydrophobic layer on it, allowing the droplet and the layer to repel from each other, which maintains the droplet's shape. In the two-dimensional electrode array, basic operations (such as transporting, splitting, merging, mixing, and storing the droplets) can be performed. The area where these operations are performed is called modules [1].

There are two approaches to modeling DMFBs. The first is the modular design approach. While the

modular design approach clearly demonstrates how droplets will be matched in a simple manner, it does not model the empty spaces when the droplets move. The second approach is a droplet path design, which models the movement of droplets, meaning that the empty spaces will also be modeled. There are many coding algorithms that people use to model the droplet path design, but among them, the greedy algorithm is the most common.

2.2 Dijkstra Algorithm

2.2.1 Overview

Dijkstra's Algorithm helps us find the shortest distance from a starting vertex to another vertex by constantly updating the distance from that vertex to another. By default, the initial distance from the starting vertex to other vertices will be set as infinity because the distance will be updated when the newly calculated distance is smaller than the current value of the distance. Subsequently, the current node will be reallocated to the node that has the smallest assigned distance. Again, the neighboring nodes are analyzed, and the distance is updated if the newly calculated distance is smaller than the current distance, which will also reallocate the current node. This process is repeated until the destination becomes the current node [2].

2.2.2 Application to Pathfinding in DMFBs

As mentioned earlier, our algorithm to efficiently route droplets on DMFBs is based on a Dijkstra algorithm, an algorithm that finds the shortest path between a node and other nodes in a graph, and in our case, these nodes represent each droplet. The Dijkstra algorithm uses the weight of the edges in a graph, which is the distance between two droplets in our case, to minimize the total distance between that node and any other node in the graph. As our algorithm is to efficiently route these droplets, a Dijkstra algorithm is a perfect match for our goal as it can be used to calculate the shortest distance between two droplets.

However, since the original Dijkstra algorithm doesn't consider moving obstacles (simultaneous movement of other droplets) we couldn't directly apply this algorithm to pathfinding problems in DMFBs. So in section 3, we introduce a modified Dijkstra algorithm to deal with this situation.

2.3 Genetic Algorithm

As mentioned earlier, our algorithm also uses a genetic algorithm, which is inspired by the process of Charles Darwin’s natural selection as it picks the fittest responses and produces a new population based on those responses. When implementing the genetic algorithm, you must have a gene (which represents the variables) and a chromosome (a response, which is a set of these genes). After creating an initial population of random chromosomes, mutations and crossovers need to occur in order to have a variety of responses [3]. From there, a method of measuring fitness is needed to select the fittest chromosomes that will be used to create the next generation. This process, similar to natural selection, repeats until the best possible response is produced.

The genetic algorithm was implemented in two parts. When determining the best setup for the start and destination of the droplets, the genetic algorithm was used by mutating the coordinates to another empty location until the best pair of the start and destination coordinates were found. In addition, when finding the most efficient path for the droplets by using our modified Dijkstra algorithm, a genetic algorithm was also implemented to determine which droplet pathways should be prioritized (which will be explained in more detail in section 3).

3. EXPERIMENT

3.1 Modeling

For our algorithm, we have source electrodes and target electrodes to determine the start and goal locations for each droplet. In addition, we satisfied the static fluidic constraint to determine the safe distance between two droplets according to their fluidic properties, preventing any unexpected mixing between two droplets. When testing our algorithm, we used an $N \times N$ grid coordinate system and ignored blockages and timing constraints to simplify the scenario.

To summarize, our algorithm inputs N (the size of the grid), the set of droplets $D = \{d_1, d_2, \dots, d_n\}$ with the source electrodes $S = \{s_1, s_2, \dots, s_n\}$ and target electrodes $G = \{g_1, g_2, \dots, g_n\}$. In addition, we take into account the static fluidic constraint, but blockages and timing constraints are both ignored.

3.2 Genetic Algorithm Setup

3.2.1 Initial Location Planning Algorithm

The path-finding algorithm (the modified Dijkstra algorithm) finds the shortest path between droplets given the start and end locations, meaning that the overall efficiency of our algorithm also depends on what the start and end locations are set as. However, since there are many pairs of start and end locations, we implemented a genetic algorithm to determine the best start and end coordinates for the droplets.

For this purpose of implementing a genetic algorithm, the chromosomes will represent the pair of coordinate values for the start and destination for every droplet: chromosome = { (StartingCoordinates, EndingCoordinates) pairs for every droplet i }. Meanwhile, the mutations will represent the random change of the starting and/or ending locations for each droplet. Finally, each crossover will represent each swap between the chromosomes defined earlier.

3.2.2 Choosing Priority Between Droplets

When finding the most efficient path for the droplets by using our path-finding algorithm, a genetic algorithm was also implemented to determine which droplet pathways should be prioritized [4]. The ranking of priority between 2 droplet pathways can influence the routing result greatly.

For example, in the diagram below, when dpB is prioritized, the droplets from group A and group B would collide in the narrow pathway, meaning that it would not be possible for the droplets to be routed in this case.

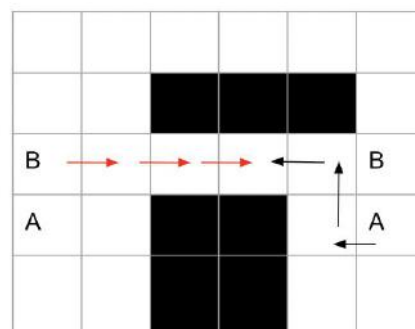


Figure 2. When Droplet Pathway B is prioritized, pathways are blocked

However, as shown in the next diagram, if dpA is prioritized, the droplet from group A will move

3.3 Modified Dijkstra Algorithm

The original Dijkstra algorithm calculates the shortest distance with fixed obstacles (non-moving obstacles). However, in our problem, several droplets move, so the distance would need to be updated according to each individual droplet movement that occurs simultaneously. In addition, if the droplet paths have the same distance, it would be advantageous to have them take a different path than other droplets (to prevent a collision). Therefore, with all these factors in mind, we modified the Dijkstra algorithm to implement it for our problem.

Our modified Dijkstra algorithm finds the shortest path of droplets by also using the priority rankings of the droplet paths. The algorithm would determine the shortest path for the droplets with the highest priority, and with this path in mind, it would calculate the shortest possible route for the next prioritized droplets while preventing collisions with droplets having a higher priority. This process would repeat until all the droplet paths are iterated through. At this time, the cost function, the function to calculate the shortest path of droplets, is defined as follows:

$$\text{cost}(p) = \alpha * \text{distance from } p \text{ to destination} + \beta * \text{minimum distance to other droplet path}$$

The value p indicates the starting location, and the values for α and β were defined experimentally, explained in section 3.3.

Moreover, in order to prevent a deadlock situation, the droplets can pass through the place where it has already been or stay in the current place.

If our modified Dijkstra algorithm is expressed in pseudo-code, it would be as followed:

for each droplet i from the highest priority:

for every position p :

let x = the distance from p to the destination

let y = min distance(other droplet path)

$\text{cost}(p) = \alpha * x + \beta * y$;

move droplet i to the position where distance is minimum

3.3 Experiment Procedures

To demonstrate the performance of the proposed algorithm, experiments were conducted with various sizes and numbers of droplets. The size of the DMFB used in the experiment was set to 12x12, 16x16, and 24x24, and the number of droplets were also set to 12, 16, and 24. For each experiment, the experiment was conducted 10 times with the same random seed number, and the time required to complete the experiment was averaged in order to rule out any possible outliers in the data set.

To find out the effectiveness of using a genetic algorithm to determine the start and end positions and to find out the effectiveness of our modified Dijkstra algorithm to route the droplets, we experimented with four algorithms. The first algorithm would be to randomly generate the start and end locations while using a greedy algorithm to determine the shortest path. By doing so, we could control certain variables to compare the individual effectiveness of the two parts to our algorithm, shown in the table in section 4. Algorithm 1 would be the combination of randomly creating the start and end locations while using the normal Dijkstra algorithm. Algorithm 2 would be the combination of randomly creating the start and end locations while using the modified Dijkstra algorithm. Algorithm 3 would be the combination of using a genetic algorithm to determine the start and end locations while using the normal Dijkstra algorithm. Finally, algorithm 4 would be the combination of using a genetic algorithm to determine the start and end locations while also using the modified Dijkstra algorithm.

For the parameters for the genetic algorithm, the crossover rate and the mutation rate are set as 1 and 0.5 respectively. The population size and the number of generations should increase as the number of droplets increases. In the cost function of the path, the parameters are set as $\alpha = 2$, $\beta = 1$ experimentally. The proposed algorithms are implemented using python language, and all the experiments are performed on a 4.2 GHz Intel(R) Core(TM) i7- 8700 windows workstation with 16 GB memory.

4. RESULTS AND ANALYSIS

Table 1. Results for the use of Dijkstra and genetic algorithm

Grid	# of Droplet	Algorithm 1		Algorithm 2		Algorithm 3		Algorithm 4	
		T _{best}	T _{avg}	T _{best}	T _{avg}	T _{best}	T _{avg}	T _{best}	T _{avg}
12x12	12	56	62.4	51	57.6	43	46.1	42	45.5
12x12	16	62	73.3	56	68.9	47	52.3	46	51.0
12x12	24	64	91.0	62	86.7	63	66.6	56	61.0
16x16	12	40	45.9	38	46.8	34	38.4	28	29.1
16x16	16	44	51.5	44	47.1	41	45.4	32	40.3
16x16	24	47	52.4	45	49.6	39	47.2	37	43.0
24x24	12	39	44.8	36	42.7	37	42.8	29	31.9
24x24	16	45	50.1	42	46.8	41	47.6	33	36.8
24x24	24	46	53.6	43	46.2	43	48.4	34	35.2

As shown in the table, algorithm 4 showed the best performance. Overall, the smaller the size of the grid and the larger the number of droplets, the higher the time required to complete the experiment. First, by comparing algorithm 1 and algorithm 2, we can see how effective the modified Dijkstra algorithm was. In all three experimental scenarios, algorithm 2 finished about 10% faster on average, indicating that the pathfinding using our modified Dijkstra algorithm was efficient even if it started at the same random position (both algorithms 1 and 2 created the start and end locations randomly).

By comparing algorithm 1 and algorithm 3, you can find out how effective the starting position using GA was (both algorithms 1 and 3 used the normal Dijkstra algorithm for pathfinding, making the variable of pathfinding controlled). Specifically, when the size of the grid is small, it was found that the set-up/placement of the positions has a much greater effect on the performance. This is believed to be due to the fact that when the size of the grid is small and the initial position is set incorrectly, a bottleneck may occur in which many droplets want to pass through, resulting in inefficiency.

Finally, looking at algorithm 4, it can be seen that the performance is much better when both the genetic algorithm for setting the location and the modified Dijkstra algorithm are implemented, unlike algorithm 2 and algorithm 3 where these two algorithms were not used together. With this in mind, it is thought that the initial position setting and the path-finding algorithm have a synergistic effect.

5. CONCLUSION

In this study, an efficient path design algorithm was proposed by combining a heuristic-based modified shortest-distance algorithm and a genetic algorithm. In DMFBs, it is important to route the droplets so that they move simultaneously and do not collide with each other. To this end, chromosomes representing the initial position and droplet priority were designed, and mutations and crossovers suitable for each were also defined. And in the modified Dijkstra algorithm, a new cost function was created by considering how close the path is to other paths in addition to the distance to the destination.

As a result of comparing the performance of the algorithm presented in this paper through randomly generated scenarios in the experiment, it was shown that both the initial position calculation using the Genetic Algorithm and the modified Dijkstra algorithm were effective [5]. In particular, when both methods were applied, it was confirmed that the experiment could be completed up to twice as fast as the general method.

As DMFBs are applied in various fields today (such as for blood testing, drug testing, antibody screening, etc), it is hoped that the efficient path-finding algorithm studied in this paper can be further developed and applied to the actual industry.

6. REFERENCES

- [1]. S. Kasasbeh, R. Gharghan, and S. S. Alrehaili, "Design and Simulation of Digital Microfluidics for Droplet-Based Microfluidic Biochips," ResearchGate, 2018. [Online]. Available: https://www.researchgate.net/figure/Main-principle-of-the-DRM-process-for-DMFBs_fig2_322655196. [Accessed: Apr. 14, 2023].
- [2]. "Dijkstra's Shortest Path Algorithm | Greedy Algo-7," GeeksforGeeks, Jul. 13, 2021. [Online]. Available: <https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/>. [Accessed: Apr. 14, 2023].
- [3]. J. Chapman, "Introduction to Genetic Algorithms, including Example Code," Towards Data Science, May 14, 2019. [Online]. Available:

<https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3>. [Accessed: Apr. 14, 2023].

- [4]. A. Postula and P. Niedziałkowski, "Optimal design of digital microfluidic biochips based on the genetic algorithm," ResearchGate, 2019. [Online]. Available: https://www.researchgate.net/figure/Structure-of-a-DMFB-a-unit-cell-side-view-b-a-general-purpose-DMFB-concits-of-a-an_fig1_336440220. [Accessed: Apr. 14, 2023].
- [5]. F. Ullah, M. Ahmad, A. Imran, S. A. Madni, and S. A. Hussain, "Digital Microfluidic Biochips: Recent Advancements towards Automated Programmable Platforms for Biological Applications," *Micromachines*, vol. 11, no. 12, p. 1052, Dec. 2020. [Online]. Available: <https://www.mdpi.com/2072-666X/11/12/1052/pdf>. [Accessed: Apr. 14, 2023].