In-Field Testing of Digital Microfluidic Biochips using Multi-Agent Reinforcement Learning

A M. Tech Project Report Submitted
in Partial Fulfillment of the Requirements
for the Degree of

Master of Technology

by

Ritik Kumar Koshta
(234101044)

under the guidance of

Dr. Sukanta Bhattacharjee

to the

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI
GUWAHATI - 781039, ASSAM

CERTIFICATE

*This is to certify that the work contained in this thesis entitled "**In-Field Testing of Digital Microfluidic Biochips using Multi-Agent Reinforcement Learning**" is a bonafide work of **Ritik Kumar Koshta (Roll No. 234101044)**, carried out in the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati under my supervision and that it has not been submitted elsewhere for a degree.*

Supervisor: **Dr. Sukanta Bhattacharjee**

Assistant Professor,

May, 2025 Department of Computer Science & Engineering,

Guwahati. Indian Institute of Technology Guwahati, Assam.

Acknowledgements

I extend my deepest gratitude to Dr. Sukanta Bhattacharjee for his remarkable patience, invaluable wisdom, and unwavering support throughout my project and placement process. His guidance, particularly during pivotal moments, has been nothing short of indispensable, and I am profoundly appreciative of his commitment to my growth and success. I am equally indebted to my family for their extraordinary generosity and unwavering encouragement, which have far exceeded my expectations. Their steadfast belief in me has been a constant source of strength, and I am truly grateful for their immeasurable support.

Abstract

Digital microfluidic biochips (DMFBs) have emerged as a transformative technology in biomedical applications, enabling programmable droplet manipulation for tasks such as drug discovery, clinical diagnostics, and point-of-care testing. Among their many advantages, DMFBs offer programmability, flexibility in fluidic operations, and versatile droplet mobility. However, their reliability can be compromised by factors such as manufacturing defects, electrode degradation, or dielectric breakdown, leading to incorrect fluidic behavior. Ensuring reliable operation is especially critical in safety-critical bioassays. In prior work, SAT solvers have been employed for in-field testing to determine optimal paths for test droplets. These droplets traverse the chip concurrently with ongoing bioassays to verify electrode functionality while maintaining assay integrity. Although SAT- based methods are robust, their scalability is significantly limited by the computational complexity associated with larger grids and complex bioassays, resulting in prohibitive test completion times. To overcome these challenges, we propose a reinforcement learning (RL)-based approach for optimizing test droplet routing in DMFBs. By leveraging RL to learn efficient and reliable paths under predefined constraints, our method ensures rapid and effective in-field testing without disrupting ongoing assays. Experimental results demonstrate that the RL-based solution not only scales efficiently to larger grids but also achieves significant reductions in test completion time, thereby enhancing the reliability and scalability of DMFB testing. This work represents a critical step forward in ensuring the dependability of DMFBs in real-world biomedical applications.

Contents

List of Figures **ix**

List of Tables **xi**

1 Introduction **1**

1.1 Microfluidic Technology 1

1.2 Digital Microfluidic Miochip DMFB 2

1.3 Defects and its Classification 4

1.4 Testing and its Types 5

1.4.1 Offline vs. Online Testing 6

1.4.2 Structural and Functional Testing 7

1.5 Context and Purpose of the Study 7

2 Review of Prior Works **11**

2.1 Prior Work 11

2.2 Motivation 14

3 Background **15**

3.1 Elements of Reinforcement Learning 15

3.2 Agent-Environment Interaction 17

3.3 Markov Decision Process (MDP) 18

3.3.1 Objective 20

3.4 Policies and Value Functions 20
3.5 Optimal Policies and Value Functions 21
3.6 Q-Learning Algorithm 22
3.7 Applications of RL 23

4 Reinforcement Learning based In-Field Testing **25**
4.1 Problem Formulation 25
4.2 Modeling In-field Testing as MDP 26
4.3 Training of a single RL Agent 27
4.4 Multi-Agent Solution for Testing 31
4.5 Optimistic Greedy Policy 33

5 Experimental Setup and Results **37**
5.1 Experimental Setup 37
5.2 Results 38
5.3 Conclusion 40
5.4 Future Scope 41

References **43**

List of Figures

1.1 (a) Schematic view and (b) Side-view of a general DMFB. The droplet is moving to the right by EWOD. 3

1.2 Mapping of a Sequential-mixing assay in 1:2:5 on a 7 × 7 array: : (a) sequencing graph, (b) synthesized output, and (c) final mapping along with droplets position. 4

3.1 Agent–Environment interaction in a Markov decision process. 17

4.1 Single agent Q-Learning architecture. The agent updates the Q-table during training (left) and uses the optimized Q-table to navigate the DMFB during deployment (right). 28

4.2 Training performance of the agent on Sequential-mixing assay. The graph shows the reward agent obtained per epoche. 30

4.3 Footprint edges covered per epoch by the agent during training in the Sequential-mixing assay. 30

4.4 Test paths obtained for the Sequential-mixing assay: (a) SAT-based method, (b) Proposed method. 31

List of Tables

1.1 Classification and examples of common defects in DMFB's 5

1.2 Differences between structural and functional testing approaches 7

2.1 Characteristics of Various Test Approaches for DMFBs 13

5.1 Summary of Assays used for comparing different approaches 38

5.2 Comparison of Experiments on Test Assays Across Methods 40

Chapter 1

Introduction

Advancements in microfluidic technologies enable miniaturization of conventional biochemical laboratory protocols into a compact, low-cost platform, often referred to as a Lab-on-a-Chip (LoC) [1]. By manipulating nano/pico liter of fluids in a programmable fashion, these chips have revolutionized point-of-care medical diagnosis [2], DNA analysis [3], cell biology [4], and biomedical research [5].

1.1 Microfluidic Technology

LoCs are broadly categorized into two main types: Digital Microfluidic Biochips (DMBs) and Flow-based Microfluidic Biochips (FMBs). FMB is a multilayer device (flow and control layers) that manipulates fluids inside a network of microchannels by actuating pressure-driven microvalves [6]. These chips are mostly designed for specific applications and offer less programmability. On the other hand, DMBs emerged as a general-purpose liquid-handling technology that manipulates pico-liter volume of discrete fluid droplets using electrical actuators on a two-dimensional array of electrodes [7]. By varying the electrode actuation patterns, many fluid-handling operations, such as droplet dispensing, transportation, mixing, and splitting, can be performed on DMBs. Several biochemical

applications including nucleic acid [8, 9], protein [10, 11], immunoreaction [12], and cell assays [13], have been successfully demonstrated on the DMBs. The FDA has cleared the DMB-based SEEKER (Baebies, NC) for clinical applications to screen lysosomal storage disorders in newborns [14].

1.2 Digital Microfluidic Miochip DMFB

A DMB consists of a two-dimensional array of electrodes for biochemical reactions and reservoirs at the boundary for reagent loading, as shown in Fig. 1.1(a). DMFs leverage electrowetting-on-dielectric (EWOD) technology to perform several fluidic operations, such as dispensing a droplet from the reservoir, movement of droplets, and mixing and splitting of droplets by controlling electrodes through a sequence of electrical actuation. EWOD-based DMF consists of two parallel glass plates (Fig. 1.1(b)), where the bottom plate is comprised of a substrate, a patterned array of individually controllable electrodes, dielectric and hydrophobic layers, and the top plate is coated with a continuous ground electrode. The control electrodes are coated with a dielectric insulator. A hydrophobic thin film is also added to the top and bottom plates to decrease the wettability of the surface.

A DMB can be represented as a grid graph $G = (V, E)$, where V denotes the set of electrodes and E is the set of edges representing the adjacency between two electrodes. A biochemical assay is represented as a directed acyclic graph (DAG) (aka sequencing graph), where nodes represent fluidic operations and edges signify the dependencies between operations. Fig. 1.2(a) shows a bioassay for mixing three fluids in $1 : 2 : 5$ ratio. The sequencing graph of a bioassay is synthesized into actuation sequence in order to realize the bioassay on the target DMB. Fig. 1.2(b) shows the symbolic actuation sequence for realizing the mixing on a 7×7 DMB. Note that the first line of the Fig. 1.2(b) shows the dimension of target DMB. The next two lines describe the input/output/waste reservoir locations. Subsequent lines of Fig. 1.2(b) represent fluidic operations executed at time step t. For example, '1 d([6,1]) d([6,7])' represents the dispense of two droplets at location $(6, 1)$

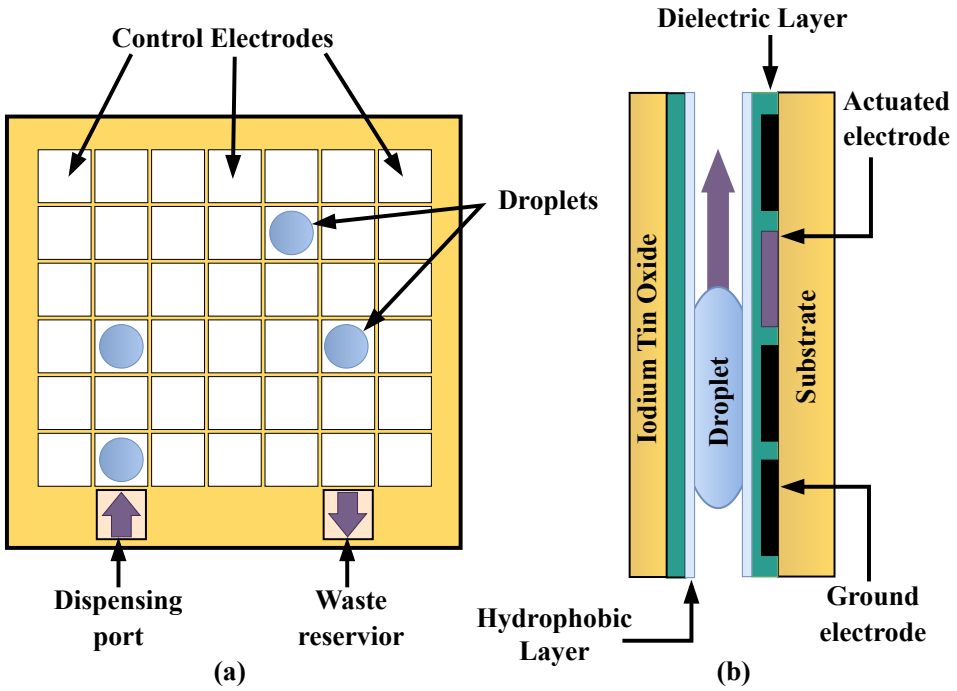

Fig. 1.1 (a) Schematic view and (b) Side-view of a general DMFB. The droplet is moving to the right by EWOD.

and $(6, 7)$, respectively, at $t = 1$. Similarly, '2 m([6,1], [6,2]) m([6,7], [6,6]) denotes the movement of droplets from locations $(6, 1)$ and $(6, 7)$ to $(6, 2)$ and $(6, 6)$, respectively, at $t = 2$. Moreover, '4 mix([6,2] [6,5], 8, 1x4) d([6,7])' denotes the mixing of two droplets located at $(6, 2)$ and $(6, 5)$ using a mixer of size 1×4. Note that the mixing operation requires 8 time steps and after mixing i.e., at $t = 12$ two resultant droplets are also placed at $(6, 2)$ and $(6, 5)$, respectively. Also a new droplet is dispensed at location $(6, 7)$ at $t = 4$. Snapshot of the DMB at any time step t is defined as the subset of vertices on which a droplet was present. An assay footprint is the union of all snapshots during assay time i.e., the set of all vertices on which a droplet was present during assay execution. We also define footprint edges as the set of all edges induced by the footprint. As depicted in Fig. 1.2(c), a droplet of reagent type B occupies vertex $(3, 7)$ in the DMB snapshot at time step $t = 7$, and transitions to vertex $(3, 6)$ at time step $t = 8$. This temporal movement induces a undirected edge $((3, 7), (3, 6))$, which we define as a footprint edge-an edge that connects

consecutive positions of the same droplet across adjacent snapshots.

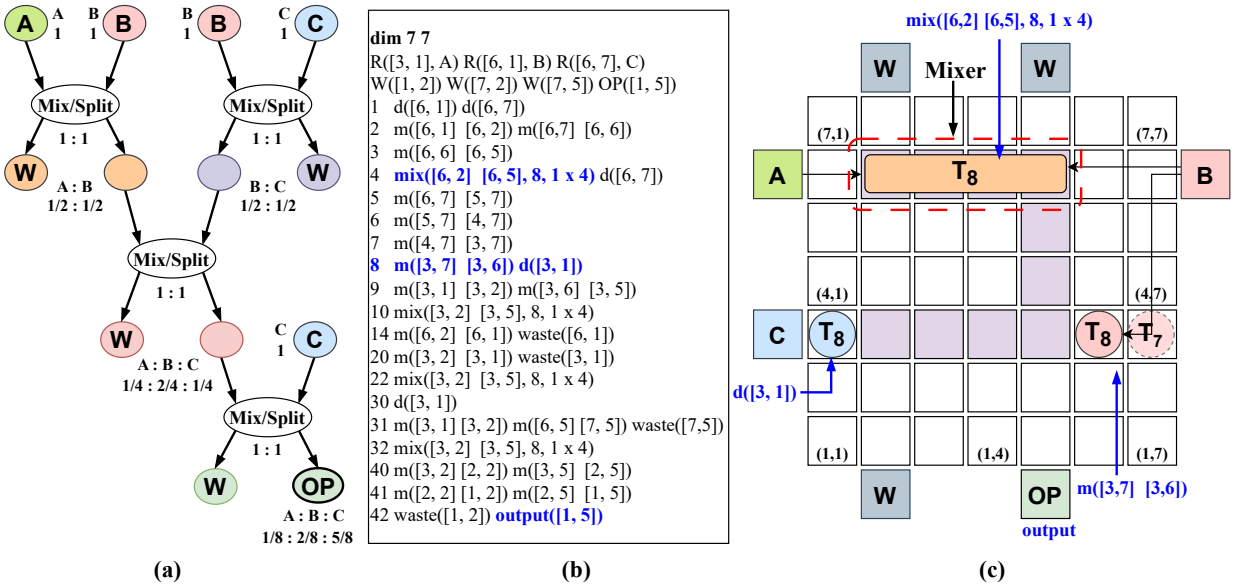

Fig. 1.2 Mapping of a Sequential-mixing assay in 1:2:5 on a 7 × 7 array: : (a) sequencing graph, (b) synthesized output, and (c) final mapping along with droplets position.

1.3 Defects and its Classification

The DMFBs faces many challenges such as material fatigue, surface contamination, electrode failure and other which can possibily interrupt in the bioassays resulting in critical errors in experiment. As the size of biochips increases these issues occures more frequently. The challenge is to ensur that testing operates alongside with standard assays without interference enabling real-time defect detection while preserving bioassay integrity [15].

The droplet based (DMFBs) are always prone to defects that can harm their functionality and reliability. These defects can be classified into three categories:

- **Permanent Defects**: These arise from irreversible physical damage or manufacturing flaws and they require hardware intervention.
- **Transient Defects**: These result from temporary environmental and operational conditions. They resolved once these conditions are addressed.

Table 1.1 Classification and examples of common defects in DMFB's

Defect	Defect Type	Description	Examples/Manifestations
Permanent Defects	Catastrophic Faults	Defects that lead to complete and irreparable system malfunction.	Electrode short (e.g., between adjacent electrodes)
			Dielectric breakdown
			Electrode open fault (failure in electrode activation)
	Electrode-Specific Faults	Issues related to electrode fabrication or configuration that permanently impact functionality.	Misaligned electrodes
			Variations in metal deposition causing improper connections
Transient Defects	Fluidic Faults	Temporary issues caused by environmental or operational factors.	Particle contamination between plates causing high impedance
			Droplet stalling due to increased friction at the medium interface
	Parametric Faults	Environmental or fabrication deviations that degrade performance temporarily.	Increased droplet viscosity due to temperature variation
			Minor deviations in insulator thickness or electrode dimensions
Logical Defects	Edge-Dependent Faults	Issues in droplet routing or electrode interactions arising from logical or operational errors.	Errors due to electrode shorts along a specific flow direction
			Cross-contamination from residue left by previous droplets
	Routing Errors	Faults arising from improper or unintended droplet paths due to misconfigured routing logic.	Droplets taking unintended paths
			Collision of droplets during complex bioassay executions

- **Logical Defects**: They originate from errors in the control logic or routing algorithms which leads to incorrect droplet behavior that can often be corrected via software adjustments.

Table 1.1 provides a detailed classification of these defect and categories along with respective examples [16]. This categorization shows the importance of understanding diverse defect types for enhancing biochip reliability.

1.4 Testing and its Types

Digital microfluidic biochips (DMFBs) undergo with various types of testing to ensure their functionality and reliability, addressing potential defects that could impact performance during bioassay execution [17][7].

1.4.1 Offline vs. Online Testing

Based on when it is conducted the testing may be divided into two categories these are as following:

Offline Testing: Offline testing is conducted post-fabrication and prior to deployment it ensures that the biochip is free from physical and structural defects.

Extensive testing is performed to cover all electrodes and pathways of the biochip it makes sure it works as expected. This involves sending tiny drops of fluid through specific channels to check if the electrodes are functioning correctly. This approach is effective in identifying critical defects, such as shorts, dielectric breakdown, or open faults.

Purpose:

- To ensure that biochips are defect-free and reliable before being used for bioassays.
- Suitable for both disposable and reusable biochips.

Online Testing: Online testing is performed during real-time bioassay execution to detect and address defects dynamically without halting operations.

It actively monitors droplet operations to detect anomalies such as stalling or routing errors during real-time bioassay execution. Unlike offline testing, it does not require additional test droplets, as assay droplets are utilized for detecting anomalies. However, this method is limited to testing only the regions of the chip that are actively in use during the bioassay.

Purpose:

- To ensure the correctness of droplet operations and mitigate errors during assay execution.
- Enables real-time fault detection and correction for improved reliability.

Table 1.2 Differences between structural and functional testing approaches

Aspect	Structural Testing	Functional Testing
Objective	Ensure physical integrity of the biochip by identifying defects in electrodes, connections, and pathways.	Validate the biochip's ability to perform fluidic operations like mixing, splitting, and dispensing.
Key Focus	Detect permanent and transient defects in structure.	Detect logical and transient defects during droplet manipulation.
Techniques	Graph-Based Models (Hamiltonian/Eulerian paths). Predefined Droplet Paths (Routing test droplets). Electrical Testing (voltage-driven tests).	Real-Time Monitoring (tracks droplet movement). Feedback Mechanisms (dynamic adjustments). Capacitive Sensing (detects droplet position and volume).
Defects Detected	Electrode shorts, dielectric breakdown, open faults, particle contamination, droplet stalling.	Routing errors, edge-dependent faults, stalling, irregular movement, volume errors.
Outcome	Ensures connectivity and operability of electrodes.	Ensures operational reliability during real-time bioassays.

1.4.2 Structural and Functional Testing

Testing in digital microfluidic biochips (DMFBs) can also be categorized based on what is being tested:

Structural Testing: Structural testing ensures the physical integrity of the biochip, verifying that all electrodes, connections, and pathways are free from defects.

Functional Testing: Functional testing validates the biochip's ability to perform fluidic operations like mixing, splitting, and dispensing.

The differences between structural and functional testing are summarized in the table 1.2.

1.5 Context and Purpose of the Study

DMBs can fail due to manufacturing defects (e.g., non-uniform coating, electrode short, misalignment of parallel plates, and broken connection to the control pin) or during operation (e.g., excessive actuation voltage applied to an electrode may cause dielectric break-

down resulting a short between a droplet and the electrode, electrode stuck on due to the actuation of the electrode for a long time). Moreover, harsh operational environments (e.g., excessive mechanical force may cause misalignment of parallel plates) and biological samples, e.g., protein adsorption, deposit residue on the surface of the chip, which may lead to contamination and problems in droplet movements. Since DMBs are often deployed in life-critical applications, the reliability of fluidic operations is of utmost importance.

DMBs must be tested adequately after manufacturing for reliable fluidic operation during assay execution. Several testing methods have been proposed to guarantee high coverage of manufacturing defects by routing test droplet(s) on the electrode array before deployment (aka offline testing) while minimizing the test application time. However, some defects may be latent and may cause errors during field operations. One way to detect in-field errors is to route test droplet(s) to cover all edge boundaries between all pairs of adjacent electrodes visited by functional droplets (aka footprint) in parallel with assay execution. However, finding a suitable test droplet routing schedule without interfering with functional droplets is challenging. Bhattacharjee et al. [18] proposed a satisfiability (SAT)-based solution to find an optimal test plan (routing path and schedule) to cover the assay footprint. The SAT-based solution is computationally expensive and, therefore, applicable only for simple bioassays running on tiny DMB grids.

To address these challenges, we propose a reinforcement learning (RL) based in-field testing framework that learns to find test droplet(s) routing plan to cover the assay footprint without disrupting the ongoing assay running on the DMB. The RL agent (i.e., test droplet) successfully finds the in-field test plan for larger DMBs quickly. The specific contributions of the paper are summarized below.

- We proposed for the first time a tabular RL-based Q-learning agent to find a near-optimal in-field test plan.
- We have performed an extensive simulation by running several assays on different size DMB grids and found that the tabular Q-learning agent could find a near-optimal

test plan for the DMB gid size up to 50×50 quickly on which SAT-based technique fails to find optimal test plan in a reasonable amount of time.

Structural testing is applicable both online for real-time validation and offline for post-manufacturing defect detection. Functional testing is primarily used online to ensure operational correctness but can also support offline tests for specific assay validations.

This chapter provided a foundational background for the topics covered in this report. It begins with an overview of microfluidic technology, highlighting its significance and applications. We then dived into the principles of digital microfluidic biochips (DMFB), exploring their structure and functionality. A classification of defects in DMFBs is presented, followed by a discussion on various testing methodologies, including offline versus online testing and the distinction between structural and functional testing. The rest of the report is organized as follows: Chapter 2 presents a review of prior research, offering insights into existing approaches and the motivation behind this study. Chapter 3 covers essential reinforcement learning concepts, including fundamental elements, agent-environment interactions, Markov Decision Processes (MDPs), policies, and learning algorithms. Chapter 4 applies reinforcement learning to in-field testing, discussing problem formulation, modeling testing as an MDP, single-agent training, and multi-agent solutions. Chapter 5 presents the experimental setup, details results obtained through our proposed methods, discusses their implications and finally, summarizes the key findings and outlines potential future research directions.

Chapter 2

Review of Prior Works

The automation of biochemical processes in drug development, genome research, and clinical diagnostics has drawn interest in digital microfluidic biochips, or DMFBs. However, flaws like electrode deterioration and dielectric breakdown raise serious concerns about their dependability. To solve these problems, a number of testing approaches have been put out over time.

2.1 Prior Work

The automation of biochemical processes in drug development, genome research, and clinical diagnostics has drawn interest in digital microfluidic biochips, or DMFBs. However, flaws like electrode deterioration and dielectric breakdown raise serious concerns about their dependability. To solve these problems, a number of testing approaches have been put out over time.

Offline testing techniques were designed to detect physical defects in DMFBs immediately after manufacturing. Su et al. [19, 20] and Mitra et al. [21] proposed methods for structural testing to ensure that all cells and boundaries in the microfluidic array function correctly. These techniques involved using Eulerization to generate efficient paths for test droplets to traverse all electrodes. Offline testing is effective in identifying manufacturing

defects but does not address faults arising during runtime operations.

Mitra et al. [22] developed an online fault detection technique to guarantee dependability while the chip is operating. Without the need for extra test droplets, this technique dynamically repurposes assay droplets to identify problems during real-time execution. This method's coverage is constrained, nevertheless, because assay droplets can only follow the routes specified by the running assay and are unable to identify every physical flaw in the chip.

Concurrent testing was proposed to address the limitations of offline and online strategies by interleaving the testing process with bioassay operations. Su et al. [23] developed a Hamiltonian-path-based structural testing approach, where the objective was to traverse all cells in the microfluidic array while minimizing conflicts with functional droplets. However, this method focused on nodes (cells) and did not ensure complete edge traversal. Su et al. [20] reformulated the concurrent testing problem to focus on edge-based testing (Eulerian path), ensuring all cell boundaries between adjacent electrodes were traversed. While this approach improved defect coverage, it failed to address potential deadlock scenarios where test droplets could neither proceed nor stall.

Bhattacharjee et al. [18] suggested a SAT-based modeling technique to determine the best test schedules in order to overcome the difficulties associated with concurrent testing in digital microfluidic biochips (DMFBs). This approach traverses all cell borders within the assay footprint, reformulating the in-field testing problem as a Boolean satisfiability problem (SAT) and guaranteeing robust, edge-based testing. In order to prevent conflicts between test and assay droplets, the SAT solver repeatedly calculates the minimal test application time. Due to the iterative nature of SAT solvers, this method involves a large computing burden even if it ensures scheduling free of deadlocks and robust fault detection. For real-time applications, this restriction creates a bottleneck, especially for complicated tests or big biochips.

Dinh et al. [24] have proposed a comprehensive testing methodology for DMFBs that

integrates practical constraints and optimizes test-application time but their approach has some limitations. The Integer Linear Programming (ILP) model, although capable of producing optimal schedules it is not scalable to larger biochip designs which restricts its applicability to modest layouts. The heuristic algorithm, though computationally efficient but it may not always achieve optimal results and serves primarily as an approximation to the ILP model's performance. Table 2.1 presents a comprehensive summary of previous research conducted in the field of testing DMBFs, including our proposed method.

Table 2.1 Characteristics of Various Test Approaches for DMFBs

Test Approach	**Testing Type**	**Def. Detect.**	**No. of Test Droplet**	**Scalability**	**Approach Type**
Offline Testing [20][21]	Structural	Full	1–2 (heuristic)	till 15×15	PMF/CPP
Online error detection [22]	Functional	Partial	Nil	any size	Table-driven
In-field (concurrent) [23]	Structural (Hamiltonian)	Partial	1–2 (heuristic)	till 15×15	ILP
In-field (concurrent) [20]	Structural (Eulerian)	Full	1–2 (heuristic)	till 15×15	PMF (Heuristic)
In-field (concurrent) [18]	Structural (Eulerian)	Full	up to 4	till 15×15	SAT
Offline Testing [24]	Structural (Eulerian)	Full	any number	till 30×30	ILP/Heuristic
In-field (concurrent) **Proposed**	Structural (Eulerian)	Full	up to 4	till 50×50	Reinf. Learn.

Recent advancements have explored reinforcement learning techniques for droplet routing under fault conditions, especially in reconfigurable biochip platforms like MEDA. Elfar et al. [25][26] proposed deep reinforcement learning (DRL)-based approaches using PPO and CNN to dynamically route droplets while avoiding degraded micro-cells, enabling real-time performance and high scalability (up to 180×180). The earlier work focuses on adaptive rerouting based on local electrode health, while the later framework incorporates transfer learning for efficient deployment across varying chip layouts. Although these methods target MEDA biochips rather than conventional DMFBs, they highlight the growing relevance of AI-based adaptive testing and routing strategies in microfluidic systems.

To enable high-throughput and fault-resilient droplet routing on MEDA biochips, Liang et al. [27] proposed a multi-agent reinforcement learning (MARL) framework where each assay droplet is autonomously controlled by a dedicated agent. The agents are trained using

PPO, ACER, and Double DQN algorithms to route droplets efficiently across degraded microelectrodes, relying solely on on-chip sensors without external cameras. The method supports parallel routing and is demonstrated on a real 60×80 fabricated MEDA chip, executing 49 concurrent droplet operations. Although targeted at MEDA platforms, this work showcases the power of MARL for dynamic, scalable droplet control in fault-prone environments

Kawakami et al. [28] proposed a deep reinforcement learning (DRL)-based droplet routing algorithm for digital microfluidic biochips (DMFBs) that can dynamically adapt to both known and unknown faults. Unlike earlier approaches that rely on static sensor data, this method updates the routing policy during execution by detecting anomalies in droplet movement, allowing it to reroute around unexpected faults. Using a PPO-trained CNN agent, the algorithm achieves high success rates and near-optimal paths on DMFBs up to 16×16 in size, without needing test droplets or camera-based feedback.

2.2 Motivation

The SAT-based method for in-field testing of digital microfluidic biochips (DMFBs) provides reliable, edge-based defect detection and prevents deadlocks. However, its iterative SAT-solving process introduces considerable computational delays, making it less suitable for large biochips or time-sensitive applications. This paper introduces a reinforcement learning (RL) approach to significantly reduce test scheduling time, providing a faster, scalable, and efficient solution while preserving the robustness and precision of the SAT-based technique.

Chapter 3

Background

In this chapter, we explored the fundamentals of Reinforcement Learning (RL), a powerful approach in machine learning where agents learn to make decisions by interacting with an environment. We covered key concepts such as rewards, policy optimization, and value functions, highlighting how RL enables adaptive learning in dynamic scenarios. Additionally, we discussed practical applications, including robotics, game playing, and autonomous systems, illustrating RL's impact across various domains. This chapter serves as a foundation for understanding how intelligent systems refine their strategies through trial and error, ultimately achieving optimal decision-making.

3.1 Elements of Reinforcement Learning

Reinforcement Learning is a type of machine learning where an agent learns how to act in an environment to achieve a goal. It does so by interacting with the environment, taking actions, and receiving feedback in the form of rewards. The goal of the agent is to maximize the total reward over time by discovering the best actions to take in different situations.

- **Agent:** The decision-maker that learns by interacting with the environment to achieve a goal.

- **Environment:** Everything the agent interacts with, which responds to the agent's actions by providing new states and rewards.

Apart from the agent and the environment, a reinforcement learning system can be described through four key components: a policy, a reward signal, a value function, and a model of the environment.

- **Policy:** A policy in reinforcement learning defines the agent's approach to making decisions in a given state. It is essentially a mapping from states of the environment to actions, determining the behavior of the agent. The policy can be a simple function, such as a lookup table, or a more complex computation like a probabilistic model. For instance:
  - **Deterministic policies** always choose the same action for a given state.
  - **Stochastic policies** assign probabilities to actions, allowing for exploration.
- **Reward signal:** A reward signal is the feedback an agent receives from the environment after each action, guiding it to maximize total rewards over time. Positive rewards reinforce good actions, while negative rewards discourage bad ones, helping the agent adjust its decisions for better outcomes.
- **Value function:** A value function specifies the long-term desirability of a state by estimating the total reward an agent can expect from that state onward. Unlike rewards, which indicate immediate outcomes, value functions focus on future rewards by accounting for subsequent states and their potential.
- **Enviroment Model:** A model of the environment predicts the next state and reward for a given state-action pair, enabling planning by simulating future scenarios. Model-based methods use this for decision-making, while model-free methods rely on trial-and-error. Modern reinforcement learning often integrates both approaches.

3.2 Agent-Environment Interaction

The interaction between the agent and the environment unfolds in discrete time steps $t = 0, 1, 2, \ldots$:

- The agent observes the current state S_t.
- Based on a policy $\pi(a \mid s)$, the agent selects an action A_t.
- The environment transitions to a new state S_{t+1} and provides a reward R_{t+1}, determined by P_{sa} and R.

This process generates a trajectory:

$$S_0, A_0, R_1, S_1, A_1, R_2, \ldots$$

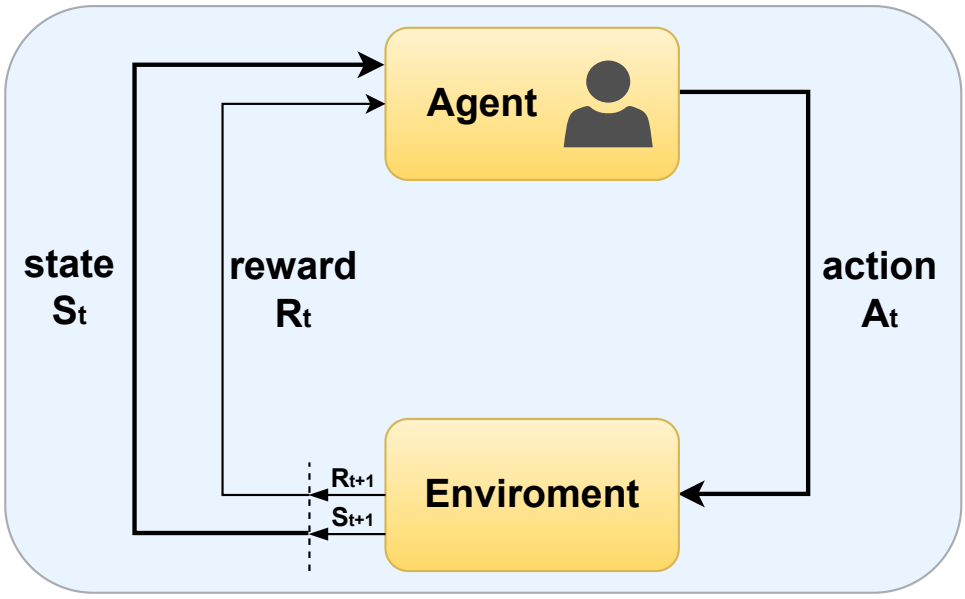

Fig. 3.1 Agent–Environment interaction in a Markov decision process.

Fig. 3 shows a flow diagram illustrating the interaction between states, actions, rewards, and transitions.

3.3 Markov Decision Process (MDP)

A Markov Decision Process (MDP) is a mathematical framework that models sequential decision-making problems where outcomes are influenced by both current actions and random factors. It provides optimal action mappings for each state under stochastic circumstances. An MDP is formally represented as a tuple: $(S, A, \{P_{sa}\}, \gamma, R)$, where:

- **State Space (S)**: The state space S is the set of all possible states the environment can occupy. A state $s \in S$ contains all the information necessary to fully describe the environment at a given time step t. The state is used to determine both the available actions and the transitions to future states.

  - S is often finite or countably infinite.
  - The state captures only the relevant information needed for decision-making, following the Markov property.

  The **Markov property** states that the next state S_{t+1} and reward R_{t+1} depend only on the current state S_t and action A_t, and not on the sequence of prior states or actions.

- **Action Space (A)**: The action space A is the set of all possible actions an agent can take. For each state $s \in S$, the agent can choose an action $a \in A(s)$, where $A(s) \subseteq A$ is the set of valid actions in state s.

  - Actions can be deterministic or stochastic.
  - The choice of an action affects both the immediate reward and the future state of the environment.

- **Transition Probabilities (P_{sa})**: The transition probabilities describe how the en-

vironment evolves based on the agent's actions. Specifically:

$$P(s', r \mid s, a) = \Pr(S_{t+1} = s', R_{t+1} = r \mid S_t = s, A_t = a)$$

This function defines the probability of transitioning to a new state s' and receiving reward r, given the current state s and action a.

- Transition probabilities satisfy:

$$\sum_{s',r} P(s', r \mid s, a) = 1 \quad \text{for all } s \in S \text{ and } a \in A(s).$$

  This property means that the total probability of transitioning to all possible next states (s') and receiving all possible rewards (r) from a given state (s) and action (a) must equal 1. This is because probabilities must account for all possible outcomes.

- Transition probabilities $P(s', r \mid s, a)$ encode the dynamics of the environment. This means they describe how the environment evolves based on the agent's actions and the likelihood of specific outcomes.

- **Discount factor (γ)**:The discount factor $\gamma \in [0, 1]$ determines the importance of future rewards relative to immediate rewards. It controls how far into the future the agent considers rewards when making decisions.

  - For $\gamma = 0$, the agent focuses only on immediate rewards.
  - For $\gamma = 1$, the agent considers all future rewards equally (for finite-horizon problems or episodic tasks).
  - Intermediate values of γ balance the trade-off between short-term and long-term rewards.

- **Reward Function (R)**: The reward function $R(s, a, s')$ assigns the expected scalar reward for transitioning from state s to state s' after taking action a. Formally, it is defined as:

$$R(s, a, s') = E[R_{t+1} \mid S_t = s, A_t = a, S_{t+1} = s'].$$

The equation expresses the expected reward (R_{t+1}) given that the agent is in state s, takes action a, and transitions to state s' at the next time step.

  - Rewards can be positive, negative, or zero, depending on the transition's outcome.
  - This function quantifies the immediate feedback the agent receives and is critical for learning and decision-making.

3.3.1 Objective

The agent's objective is to maximize the expected cumulative discounted reward, also known as the return:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots$$

Or equivalently:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

This ensures the agent considers both immediate and future rewards, balancing short-term gains and long-term benefits.

3.4 Policies and Value Functions

- **Policy (π)**: A policy $\pi(a \mid s)$ defines the probability of taking action a in state s. The goal of reinforcement learning is to find an optimal policy π^* that maximizes the expected return.

- **State-Value Function ($v_\pi(s)$)**: The expected return starting from state s and following policy π is defined as:

$$v_\pi(s) = E_\pi[G_t \mid S_t = s]$$

Expanded:

$$v_\pi(s) = E_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s \right]$$

- **Action-Value Function ($q_\pi(s, a)$)**: The expected return starting from state s, taking action a, and then following policy π is defined as:

$$q_\pi(s, a) = E_\pi[G_t \mid S_t = s, A_t = a]$$

Reinforcement learning methods are categorized based on how the agent learns from interactions with the environment. On-policy methods, like SARSA, learn and improve the policy that the agent is currently following, balancing exploration and exploitation within the same policy. In contrast, off-policy methods, like Q-learning, allow the agent to learn about a separate target policy while exploring the environment using a different behavior policy. This distinction is critical in applications requiring flexible exploration strategies or learning optimal policies efficiently.

3.5 Optimal Policies and Value Functions

- **Optimal State-Value Function ($v^*(s)$):** The maximum value achievable from state s under any policy:

$$v^*(s) = \max_\pi v_\pi(s)$$

- **Optimal Action-Value Function ($q^*(s, a)$):** The maximum value achievable from

state s, taking action a, and then following the optimal policy:

$$q^*(s,a) = \max_{\pi} q_\pi(s,a)$$

3.6 Q-Learning Algorithm

Q-Learning is a *model-free, off-policy reinforcement learning algorithm* used to learn the optimal action-value function, $Q^*(s,a)$, without requiring prior knowledge of the environment's dynamics. As an off-policy method, Q-Learning learns the optimal policy while following a potentially different behavior policy, such as ϵ-greedy exploration.

The goal of Q-Learning is to estimate $Q^*(s,a)$, which represents the maximum expected cumulative reward achievable from taking action a in state s and following the optimal policy thereafter. Using the Bellman optimality equation, $Q^*(s,a)$ is expressed as:

$$Q^*(s,a) = \max_{\pi} E\left[G_t \mid S_t = s, A_t = a\right],$$

where G_t is the cumulative reward.

The Q-Learning update rule is defined as:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[R_t + \gamma \max_{a'} Q(S_{t+1}, a') - Q(s,a)\right],$$

where:

- $Q(s,a)$: Current estimate of the action-value function.
- α: Learning rate, controlling the step size of updates.
- R_t: Reward received after taking action a in state s.
- γ: Discount factor, weighing future rewards.
- $\max_{a'} Q(S_{t+1}, a')$: Estimate of the best future value from the next state S_{t+1}.

Q-Learning is guaranteed to converge to the optimal action-value function $Q^*(s, a)$ under the following conditions:

- All state-action pairs are visited infinitely often.
- The learning rate (α) decreases over time but remains positive.

As an off-policy algorithm, Q-Learning separates the behavior and target policies, allowing flexibility in exploration. It uses the Bellman optimality equation as a foundation and iteratively updates $Q(s, a)$ to converge to $Q^*(s, a)$, enabling the agent to learn optimal behavior in an unknown environment.

3.7 Applications of RL

Reinforcement learning is used in many domains, such as healthcare [29], robotics [30], autonomous control, and natural language processing (NLP) [31]. Its applicability also extends to scheduling, resource setup, autonomous systems, gaming, networking and communication, the Internet of Things, and computer vision.

Chapter 4

Reinforcement Learning based In-Field Testing

In this section, we first formulate the in-field testing problem on DMBs. We then model the in-field testing problem as a Markov Decision Process (MDP), enabling the use of RL algorithms to find a test plan.

4.1 Problem Formulation

Input: An actuation sequence for implementing a bioassay on the target DMB.

Output: Test droplet routing plan (path and schedule) between source and sink reservoirs.

Objective: To find test droplet routing plan covering assay footprint edges (i.e., an edge that connects consecutive positions of the same droplet across adjacent snapshots) without colliding with assay droplets (i.e., test droplet must not enter the 3×3 grid region centered on any assay droplet at the same timestep). Also, minimize the test droplet routing time.

4.2 Modeling In-field Testing as MDP

We model the in-field testing of DMB as MDP $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, p, \gamma, \mathcal{R}\}$, where an agent (test droplet) select an action (either moves to one of its four neighborhood or remain stationary) based on its state to maximize the future reward (calculated based on the footprint edges covered by the agent). The elements of $\mathcal{M}$ are defined as follows.

State ($\mathcal{S}$): Each state $s \in \mathcal{S}$ consists of agent's position on the DMB grid and the current time step t i.e., $s = (x, y, t)$ where x and y denote the test droplet's position on the $M \times N$ DMB grid at t. Therefore, $\mathcal{S} = \{(x, y, t) : x \in 1, 2, \cdots, M, y \in \{1, 2, \cdots, N\}, t \in \{1, 2, \cdots, T\}\}$, where T denotes the assay time. Since we know the assay droplet locations on the DMB grid at any time step $t \in \{1, 2, \cdots, T\}$, inclusion of the time as part of the state representation should suffice to identify obstacles for the agent. Note that time t acts as a correlated feature that indirectly captures the effects of obstacles without including their exact locations in the state space. Such state representation drastically reduce the size i.e., for a typical DMB of size $M \times N$, $|\mathcal{S}| = M \cdot N \cdot T$.

Action ($\mathcal{A}$): At any time step, the test droplet (agent) can remain stationary (action: $\odot$) or move one cell to its right (action: $\rightarrow$), left (action: $\leftarrow$), up (action: $\uparrow$) or down (action: $\downarrow$). Formally, $\mathcal{A}(S) = \{\odot, \leftarrow, \rightarrow, \uparrow, \downarrow\}$ for all $s \in \mathcal{S}$.

Transition probabilities (p): An agent moves deterministically as a consequence of an action. For example, at time t, the agent is on location (x, y) and selects action $\rightarrow$ to find itself on location $(x + 1, y)$ at time $t + 1$. Such transition probability is defined as

$$p(s' \mid s, a) = \begin{cases} 1, & \text{if } s' = (x+1, y, t+1), \\ & s = (x, y, t), a = \rightarrow \\ 0, & \text{otherwise.} \end{cases}$$

Reward ($\mathcal{R}$): The agent is rewarded when it covers new footprint edges and penalized for colliding with assay droplets or performing non-productive steps, i.e., without covering

any new footprint edge at any time step. After the completion of the assay, i.e., at time T, a final reward is given based on the proportion of covered footprint edges. The reward function guides an agent to find a test plan (policy) by maximizing the coverage of footprint edges within T.

Discount Factor (γ): A high discount factor 0.99 is chosen to encourage the agent to prioritize long-term rewards.

4.3 Training of a single RL Agent

For training, we use the tabular RL algorithm Q-learning [32] to learn the action-value function $q(s, a)$ for all state-action pairs (s, a). The Q-learning agent maintains a two-dimensional table $Q_{|\mathcal{S}|\times|\mathcal{A}|}$ for storing $q(s, a)$. Algorithm 1 shows the pseudo-code of the Q-learning-based RL agent for in-field testing. The algorithm takes a symbolic representation of the actuation sequence as input for getting the target DMB grid size ($M \times N$), assay time (T) and extract the DMB snapshot for each time step $t = 1, 2, \cdots, T$ along with the footprint edges to be covered by the test droplet. Before starting the learning process, the agent initializes the Q-table and hyperparameters. In each episode of the learning, the agent initializes the covered footprint edges to null, resets the test droplet location, and assigns time step $t = 0$. After that, the agent selects an action for each time step to route the test droplet using an ϵ-greedy exploratory policy [33]. Upon taking an action, the agent transitions deterministically to the next state (s') and receives a reward (r) as follows: 15 for successfully covering a footprint, -20 for colliding with an assay droplet, -1 for taking an action without covering a new footprint edge. A final reward incentivizing the agent for both survival and exploration is granted if the agent does not collide with assay droplets until the end of the episode. Value of the final reward is determined based on the ratio of covered footprint edges to the total footprint edges, with a maximum possible reward of 50. After that, for the transition (s, a, s'), the agent updates the Q-table. Note that an episode can terminate before time step T if all footprint edges are covered or the

agent collides with an assay droplet at any time step $t \in \{1, 2, \cdots, T\}$. Starting from a high value, ϵ is typically decayed after each episode, allowing the agent to gradually shift from exploration to exploitation as it gathers more experience. To prevent the agent from becoming completely greedy, ϵ can reach up to a minimum threshold $\epsilon_{\min}$, i.e., $\epsilon \geq \epsilon_{\min}$ is set, ensuring that a small degree of exploration is always maintained. This gradual transition helps the agent to focus on exploiting optimal actions while still exploring new possibilities during the early stages of learning. We used learning rate α, which increases gradually by a factor of α_{gain} until $\alpha \leq \alpha_{\max}$, for controlling how much the agent updates its Q-values in response to new experiences. Smaller values of α favor stability and slower learning, while larger values allow quicker adaptation but may introduce noise. By gradually increasing α, the agent starts with conservative updates and gains confidence as training progresses.

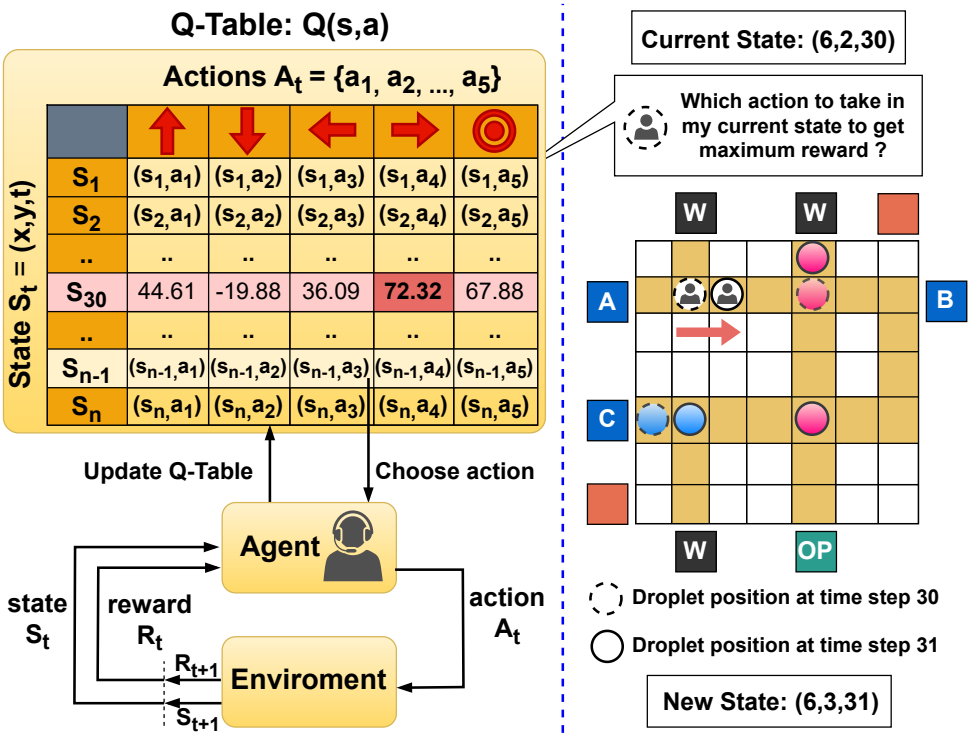

Fig. 4.1 Single agent Q-Learning architecture. The agent updates the Q-table during training (left) and uses the optimized Q-table to navigate the DMFB during deployment (right).

To demonstrate the RL-based in-field testing, an RL agent is trained for the mixing

Algorithm 1: Q-Learning for In-Field Testing

Input: Actuation sequence
Output: In-field test plan
// Training Setup
Initialize $Q(s,a) = 0, \forall s \in \mathcal{S}, a \in \mathcal{A}$
Set hyper-parameters for learning $(\alpha, \alpha_{\text{gain}}, \alpha_{\max})$, exploration $(\epsilon, \epsilon_{\min}, \epsilon_{\text{decay}})$ and discount factor γ
// Training continues for n_{epochs}
for *(*$i = 1$*;* $i \leq n_{epochs}$*;* $i = i + 1$ *)* **do**
  for *(each episode in* i_{th} *epoch)* **do**
    resetEnv() // reset test droplet position, covered footprint edges and time step
    $s =$ initState() // initialize state
    for *(each time step* $t = 1, 2, \cdots, T$*)* **do**
      $a =$ selectAction() // using ϵ-greedy policy
      $s' =$ takeAction(a) // take agent to new state
      flag $=$ collisionCheck(s', t) // agent-obstacle
      $r =$ getReward()
      $\text{TD}_{\text{error}} = \gamma \max_{a'} Q(s', a') - Q(s, a)$
      $Q(s,a) = Q(s,a) + \alpha \times \text{TD}_{\text{error}}$
      $s = s'$
      if *(all footprint edges are covered) or (flag)* **then**
        break // terminate the episode
      $\epsilon = \max\{\epsilon_{\min}, \epsilon \times \epsilon_{\text{decay}}\}, \alpha = \min\{\alpha_{\max}, \alpha \times \alpha_{\text{gain}}\}$
return $\pi(s) = \arg\max_a Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}$

assay (Fig. 1.2(a)) running on the 7×7 DMB (Fig. 1.2(c)). The agent was trained with the following hyper-parameters: starting with learning rate $\alpha = 0.001$ we increase it by factor of $\alpha_{\text{gain}} = 1.00005$ till $\alpha_{\max} = 0.8$; discount factor $\gamma = 0.99$; and an initial exploration rate $\epsilon = 1.0$, which decays exponentially with a factor of $\epsilon_{\text{decay}} = 0.99995$ until reaching $\epsilon_{\min} = 0.01$. The training was carried out over 150 epochs, where each epoch runs for 1000 episodes.

Fig. 4.2 shows the average reward obtained by the agent increases and stabilizes as training progresses, indicating that the agent successfully learns a high-reward policy. Fig. 4.3 shows the average number of footprint edges covered by the agent also increases, showing high coverage with learning. To better visualize underlying trends, we plotted the maximum, minimum, and mean number of footprints covered at each epoch. These curves provide a clear representation of the performance range and average behavior throughout the training process.

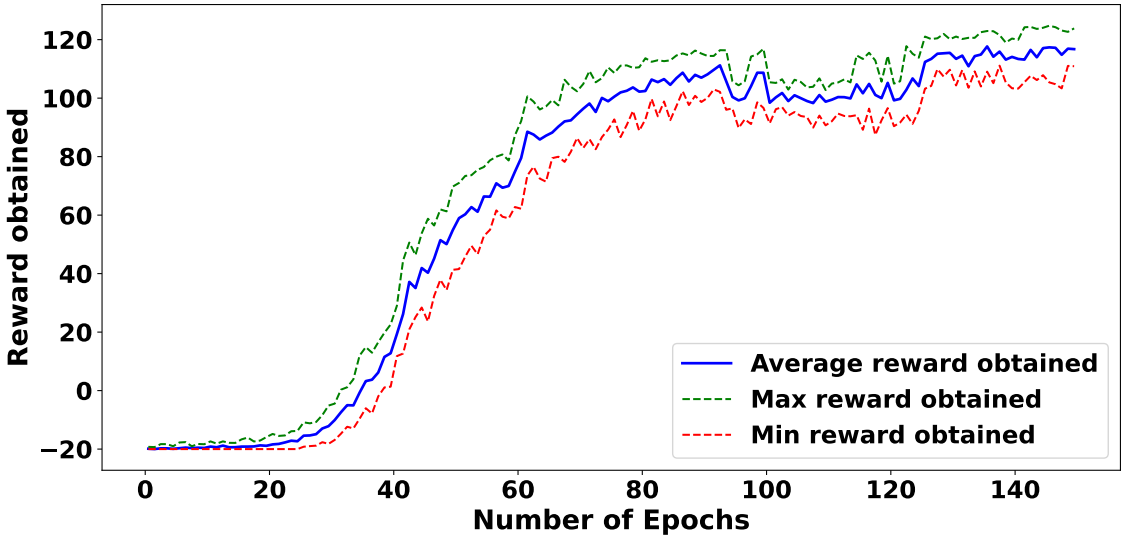

Fig. 4.2 Training performance of the agent on Sequential-mixing assay. The graph shows the reward agent obtained per epoche.

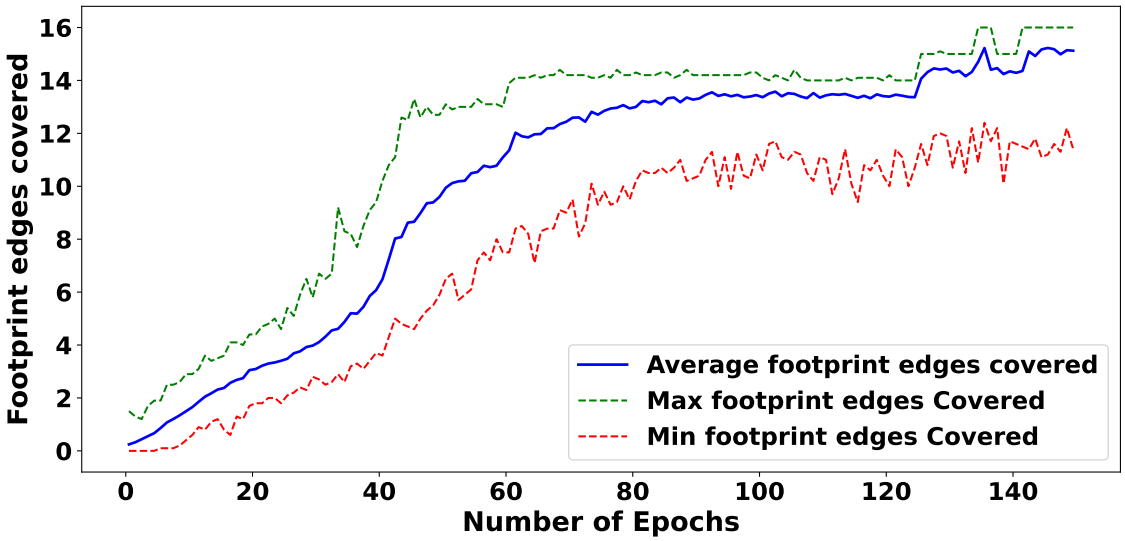

Fig. 4.3 Footprint edges covered per epoch by the agent during training in the Sequential-mixing assay.

Once training is complete, the learned Q-table is frozen, and the test droplet routing plan is represented as the final policy. At each time t, the agent selects actions according to the learned policy $\pi(s) = \arg\max_a Q(s, a)$ to route the test droplet without collision and cover footprint edges. Figs. 4.4(a)-(b) show the in-field test plan generated with the existing SAT-based approach and proposed Q-learning agent, respectively. The proposed

method can cover all footprint edges by taking only two extra time steps compared to the optimal SAT-based method.

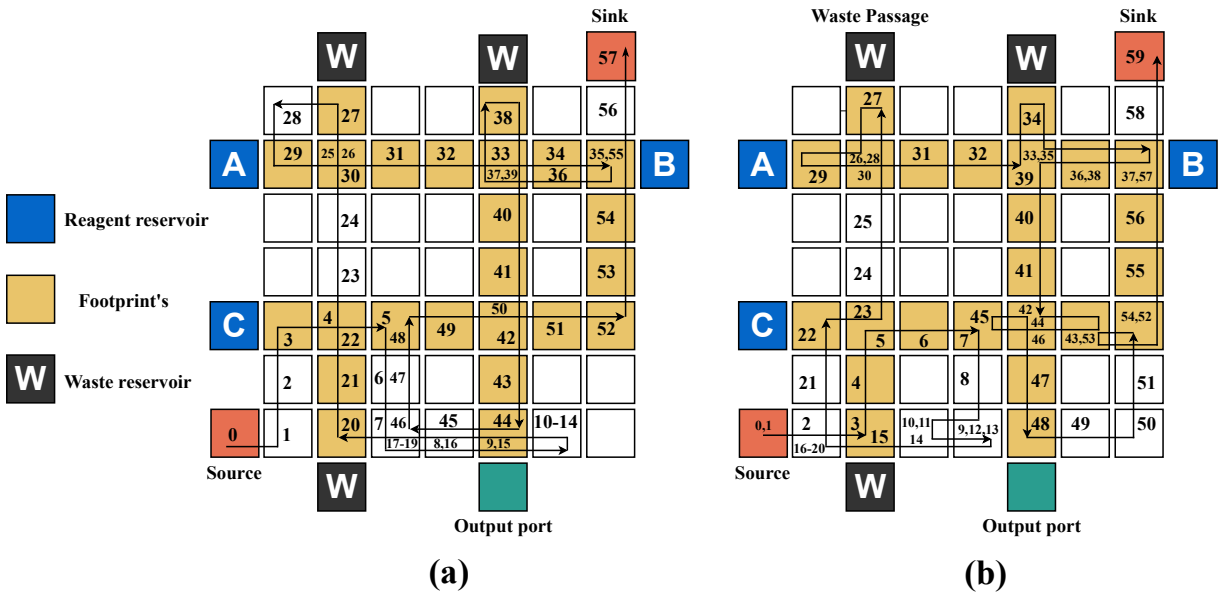

Fig. 4.4 Test paths obtained for the Sequential-mixing assay: (a) SAT-based method, (b) Proposed method.

4.4 Multi-Agent Solution for Testing

Instead of adopting conventional Multi-Agent Reinforcement Learning (MARL) training schemes such as centralized training, concurrent learning, or parameter sharing [27], our framework employs a custom decentralized variant of Independent Q-Learning (IQL) [34] tailored for efficient multi-agent coordination on a Digital Microfluidic Biochip (DMFB). Standard MARL schemes, particularly centralized approaches, suffer from scalability issues due to the exponential growth in joint state-action spaces with the number of agents. To overcome this, our approach models each test droplet as an independent Q-learning agent that maintains its own Q-table, learns its own policy, and operates within its spatial quadrant on the shared grid. This custom MAQL setup avoids the computational overhead of centralized or shared policies while enabling safe, distributed exploration and coordination through a reward structure and a priority-based action execution scheme. The result is

a lightweight, scalable, and effective solution specifically optimized for footprint coverage and collision avoidance in DMFB assay testing.

Algorithm 2 is the extension of single-agent Q-learning framework to allow concurrent operation of multiple test droplets on a DMFB. Each test droplet is modeled as an autonomous reinforcement learning agent that operates in a shared grid-based environment. The objective is to find test paths that collectively cover all assay footprint edges using multiple test droplets, while strictly avoiding any collisions with active assay droplets and with each other. Each agent A_i learns independently using Independent Q-Learning (IQL), maintaining its own Q-table $Q_i(s, a)$ where s is the agent's state (x_i, y_i, t) and $a \in \mathcal{A}$. To ensure balanced coverage of the assay footprint across the grid and promote effective coordination, each agent A_i is assigned a specific quadrant $q_i \subseteq G$, where G denotes the entire grid. An agent A_i is rewarded only for covering footprint edges that lie in his assigned quadrant q_i. While A_i is allowed to move outside q_i if necessary, no reward is granted for covering footprint edges outside of his own quadrant. This strategy serves two key purposes: it significantly reduces the likelihood of agent-agent collisions by spatially distributing their areas of operation, and it prevents over-concentration of activity in a single region, thereby promoting uniform coverage of the entire assay footprint. The reward function $\mathcal{R}$ is designed to promote cooperative behavior among agents while ensuring safe and efficient coverage of the assay footprint. Each agent receives a reward of 15 for covering a previously unvisited footprint edge, a penalty of -20 for colliding with an assay droplet, and -15 for colliding with another test droplet. Additionally, a small penalty of -1 is applied to discouraging unnecessary or unproductive movement. To enforce strict safety constraints, the episode is immediately terminated if any agent collides with either an assay droplet or another agent. However, if all agents survive until the end of the episode without any collisions, a final reward is granted to each agent. This final reward is equal to the total footprint edges collectively covered by all the agents during the episode divided by the total number of footprint edges, up to a maximum possible reward of 100. This design

encourages agents to act both cautiously and cooperatively, as maximizing the final reward requires full team survival and effective coverage of the assay footprint.

To further avoid collisions among agents during execution, a priority-based coordination system is implemented.At each time step t, the action sequence of agents is determined based on their priority, where an agent with a greater number of footprint edges in its quadrant is assigned higher priority ($A_1 > A_2 > A_3 > A_4$). The highest-priority agent selects and executes its action first, without considering the actions of others. Subsequently, each lower-priority agent selects its action while checking for potential collisions with all agents that have already moved in the current time step t. If a collision is detected i.e., if the next position of a lower-priority agent overlaps with that of a higher-priority agent, it will remain stationary for that step. Now if the event of an actual agent-agent collision occur, the penalty is assigned only to the higher-priority (initiating) agent, which is held responsible for the conflict. This asymmetric penalty encourages leading agents to make safer and more conservative decisions. The priority-based system introduces a lightweight form of coordination without the need for explicit inter-agent communication or centralized control, enabling scalable and safe execution in multi-agent droplet routing and footprint testing scenarios.

After the assay time T ends, both single-agent and multi-agent systems apply a greedy nearest-first strategy to cover any remaining footprint edges. Agent selects the closest uncovered footprint based on Manhattan distance from its current position. This process continues until all edges in the assigned region are covered. Finally, the agent navigates to its nearest sink location to complete the task.

4.5 Optimistic Greedy Policy

In addition of ϵ-greedy behavior we have also used **Optimistic Greedy Policy** [35]. This policy encourages exploration by initializing all Q-values to high values, leading the agent to initially favor less-visited actions. As the agent interacts with the environment, the

Q-values are refined, gradually reflecting the true expected rewards of actions and the agent shifts toward exploiting actions that yield the highest rewards. The initial optimism promotes broader exploration, which is particularly useful when the agent has minimal prior knowledge of the environment. This strategy ensures diverse behavior and thorough exploration before converging on the optimal policy. Additionally, once the learning process reaches a saturation point, a soft epsilon is maintained to allow occasional exploration, helping the agent avoid local optima and adapt to any dynamic changes in the environment.

Algorithm 2: MAQL for In-Field Testing

Input: Actuation sequence, number of test droplets k
Output: In-field test plan

// Training Setup
for *($j = 1$; $j \leq k$; $j = j + 1$)* **do**
    $Q_j(s, a) = 0, \forall s \in \mathcal{S}, a \in \mathcal{A}$
    Assign quadrant q_i
    Assign priority based on the number of footprint edges in q_i
Set hyper-parameters for learning $(\alpha, \alpha_{\text{gain}}, \alpha_{\max})$, exploration $(\epsilon, \epsilon_{\min}, \epsilon_{\text{decay}})$ and discount factor γ
// Training continues for n_{epochs}
for *($i = 1$; $i \leq n_{epochs}$; $i = i + 1$)* **do**
    for *(each episode in i_{th} epoch)* **do**
        resetEnv()
        // initialize state of each agent
        s_j = initState(), for $j = 1, 2, \cdots, k$
        flag = false
        for *(each time step $t = 1, 2, \cdots, T$)* **do**
            // for each agent
            for *($j = 1$; $j \leq k$; $j = j + 1$)* **do**
                a_j = selectAction()
                s'_j = takeAction(a_j)
                // check if agent j collides with higher priority agents $1, 2, \cdots, j-1$
                if *(isCollisionBetweenAgents($j, 1, j-1$)* **then**
                    $a_j = stay$; s' = tickState(s)
                    // 'tickState' keep agent position same while increment t
            for $j = 1$; $j \leq k$; $j = j + 1$ **do**
                // check if agent j collides with lower priority agents $j+1, j+2, \cdots, k$
                if *(isCollisionBetweenAgents($j, j+1, k$))* **then**
                    flag = true; $r_j -=$ penalty; // penalize the higher priority agent
                // 'isCollision' checks for collision between agents and functional droplets at t
                flag = flag *or isCollision*(s'_j, t);
                $r_j +=$ getReward(s_j);
                $\text{TD}_{\text{error}} = \gamma \max_{a'} Q(s'_j, a') - Q(s_j, a_j)$
                $Q_j(s_j, a_j) \leftarrow Q(s_j, a_j) + \alpha \times \text{TD}_{\text{error}}$
                $s_j = s'_j$;
            if *((all footprint edges are covered) or (flag))* **then**
                break; // terminate the episode
        $\epsilon = \max\{\epsilon_{\min}, \epsilon \times \epsilon_{\text{decay}}\}, \alpha = \min\{\alpha_{\max}, \alpha \times \alpha_{\text{gain}}\}$
return $\pi_i(s) = \arg\max_a Q_i(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}, \forall i \in [1, k]$

Chapter 5

Experimental Setup and Results

This section outlines the system specifications, experimental setup, and results obtained from our reinforcement learning (RL)-based testing framework for digital microfluidic biochips. We also present a comparative analysis against the traditional SAT solver method to evaluate the efficiency and scalability of the RL approach.

5.1 Experimental Setup

The experiments were conducted on a system running Microsoft Windows 11 Pro, equipped with an AMD Ryzen 5 3550U CPU, Radeon Vega Mobile Graphics, 8 GB RAM, and 476.9 GB dual SSD storage. This setup ensured smooth execution of both simulation and RL model training.

Table 5.1 summarizes the full set of experimental assays considered in this study. It includes both standard benchmark assays (originally analyzed using a SAT-based method) and newly constructed, more complex assays introduced in this work. The latter were designed by merging multiple baseline assays and executing them concurrently to simulate larger, more challenging testing conditions. These composite assays were specifically created to stress-test the scalability and robustness of our RL-based framework.

For all experiments reported in Table 5.1, the discount factor $\gamma = 0.99$ is kept constant.

The learning rate is set to $\alpha = 0.001$, with a gain factor $\alpha_{\text{gain}} \in [1.00003,\ 1.00005]$, where lower values are used for more complex assays. The maximum learning rate is bounded by $\alpha_{\max} \in [0.8,\ 0.9]$ with higher values applied to complex assays; The exploration rate is initialized at $\epsilon = 1.0$, with a decay factor $\epsilon_{\text{decay}} \in [0.99995,\ 0.999995]$; higher decay values are used for more complex assays. The minimum exploration rate is fixed at $\epsilon_{\min} = 0.1$. The number of training epochs ranges from 150 to 1000 depending on the assay complexity, with more epochs allocated to more complex assays.

Table 5.1 Summary of Assays used for comparing different approaches

Test ID	**Assay Name**	**Dimension**	T_{assay}
Assay 1	InVitro [18]	7 x 7	36
Assay 2	Sequential mixing of three reagents (Fig.1.2)	7 x 7	42
Assay 3	Multiple target mixture preparation: $A : B : C = 1 : 2 : 1$ and $A : C : D = 1 : 2 : 1$.	9 x 9	122
Assay 4	MinMix mixing [36] $R_1 : R_2 : R_3 : R_4 = 95 : 87 : 59 : 15$	11 x 11	192
Assay 5	Multiple target mixture preparation: $A : B : C : D : E : F = .1 : 1 : 1 : 1 : 2 : 10$, and $1 : 1 : 1 : 1 : 10 : 2$	13 x 13	155
Assay 6	Multiplexed InVitro [37]	16 x 16	160
Assay 7	Three instances of PCR mixture [38]	16 x 16	220
Assay 8	Plasmid DNA [36] mixing ratios = $(57 : 28 : 6 : 6 : 6 : 3 : 150)$	16 x 16	195
Assay 9	Multiplexed InVitro & Glucose [37, 20]	18 x 33	203
Assay 10	Mixing of six reagents: $A : B : C : E : G : H = 1 : 1 : 2 : 2 : 1 : 1$ and $A : B : D : F : G : H = 1 : 1 : 2 : 2 : 1 : 1$	30 x 30	184
Assay 11	Four instances of Multiplexed Glucose assays [20]	32 x 32	128
Assay 12	Combined execution of five complex assays—Assay 2, Assay 4, Assay 6, Assay 8, and Assay 10	50 x 50	533

5.2 Results

Table 5.2 presents a comprehensive comparison of all experimental assays evaluated using different testing approaches and agent configurations. For each test case, the number of agents, the total number of footprints, the test application time (TAT), and the running time are reported for three methods: SAT solver, ε-greedy reinforcement learning, and optimistic Q-learning. One point to note is that In the multi-agent setup with four agents, the SAT-based method utilizes a divide-and-conquer approach to simplify computation. On a larger grid, it reduces the number of boolean variables by dividing the space into four distinct regions, each assigned to a specific test droplet. The solution is then determined independently for each part using test droplets, making the problem more manageable. In contrast, reinforcement learning (RL) approaches—whether epsilon or optimistic greedy

treat the problem as a unified whole rather than decomposing it. While agents are encouraged to remain within their designated quadrants for structured movement, they can also step outside if necessary to avoid collisions.

The TAT columns represent the total time taken to complete all test footprints using the specified strategy, while the running time columns indicate the computational effort (e.g., SAT solving or RL training time) required to generate the test path. Across various assays and grid sizes, RL-based approaches demonstrate competitive TAT—often near or slightly higher than that of SAT—while significantly reducing total computation time. Edge Coverage % refers to the percentage edges from the footprint that are successfully traversed by the algorithm within the allotted assay T_{assay} time. It serves as a key metric for evaluating the completeness and effectiveness of a solution in covering the designated spatial objectives. A higher Edge Coverage % indicates that the algorithm was able to explore a greater portion of the footprint within the time limit, reflecting better spatial coverage performance under time-constrained conditions. To ensure practicality, a cutoff time of approximately 3 hours was used for SAT-based experiments. If a solution was not obtained within this limit, the SAT solver was considered to have timed out, and we skipped further SAT evaluation for that assay and the TAT and Edge Coverage% are mentioned as NA for that particular assay on SAT solver.

In many scenarios, reinforcement learning incurs a modest increase in TAT due to its exploration-driven nature. However, it compensates by rapidly finding valid solutions, especially in larger or more complex grid environments. For example, SAT solvers may require 19,825.72 seconds to solve a 16×16 grid, whereas RL-based approaches can complete training and testing for a 16×16 grid in just 1695.78 seconds.

Among the two RL strategies, optimistic Q-learning consistently converges faster and achieves TAT values that are near similar to or slightly better than ε-greedy, while offering improved runtime performance. This makes it particularly suitable for real-time and scalable test generation in digital microfluidic biochips.

Table 5.2 Comparison of Experiments on Test Assays Across Methods

ID	T_{assay}	# Footprints	# Agents	Test Application Time			Running Time (seconds)			Edge Coverage %		
				SAT	ε-greedy	Opt. greedy	SAT	ε-greedy	Opt. greedy	SAT	ε-greedy	Opt. greedy
Assay 1	36	20	1	45	46	46	0.74	88.77	68.24	75.00	70.00	70.00
Assay 2	42	24	1	57	65	59	7.25	78.00	69.19	66.67	62.50	66.67
Assay 3	122	104	1	159	186	179	4831.05	1094.54	983.49	73.91	69.23	68.26
Assay 4	192	49	1	192	207	202	852.01	762.16	715.41	100	95.91	97.95
Assay 5	155	108	1	191	220	210	12698.28	1275.11	1201.97	88.04	83.33	85.18
Assay 6	160	319	4	160	207	182	19825.27	1842.81	1695.78	100.00	87.77	95.61
Assay 7	220	126	1	NA	256	252	Time Out	1396.39	1249.65	NA	92.06	92.85
Assay 8	195	115	1	NA	243	220	Time Out	1366.26	1197.09	NA	83.47	87.82
Assay 9	203	382	4	NA	277	239	Time Out	2058.34	1975.68	NA	84.03	97.12
Assay 10	184	261	4	NA	218	211	Time Out	2178.30	2094.44	NA	95.40	98.08
Assay 11	128	252	4	NA	171	151	Time Out	1673.85	1587.95	NA	84.92	90.47
Assay 12	533	768	4	NA	636	598	Time Out	8312.18	7893.27	NA	91.53	95.18

5.3 Conclusion

This study introduces a novel reinforcement learning (RL)-based framework for in-field testing of digital microfluidic biochips (DMFBs), addressing the critical need for scalable and efficient fault detection in real-world biomedical applications. By dynamically computing optimal test paths that align with fluidic constraints, the proposed approach enables seamless concurrent testing alongside ongoing bioassays without interference.

The RL framework effectively leverages the fault detection model, identifying defects such as electrode malfunctions or droplet transfer failures with high precision. Experimental results underscore its superiority over traditional SAT-based methods, demonstrating significant reductions in test completion times and scalability to larger grid sizes and more complex bioassay operations.

This work establishes a robust, scalable, and efficient foundation for ensuring the reliability of DMFBs, facilitating their adoption in advanced biomedical applications such as drug discovery, clinical diagnostics, and point-of-care testing. By overcoming the computational bottlenecks of existing methods, this approach represents a critical advancement in

the dependability of DMFBs, ensuring their long-term viability in safety-critical scenarios.

5.4 Future Scope

To enhance the scalability and adaptability of our model, our future research will focus on integrating advanced neural network-based methods. Specifically, we aim to explore Deep Q-Networks (DQN) and other reinforcement learning techniques to refine our approach. By leveraging these methodologies, we seek to improve the model's ability to dynamically adjust and optimize performance in complex environments, ensuring greater efficiency and flexibility in its applications.

References

[1] X. Xu, L. Cai, S. Liang, Q. Zhang, S. Lin, M. Li, Q. Yang, C. Li, Z. Han, and C. Yang, "Digital microfluidics for biological analysis and applications," *Lab on a Chip*, vol. 23, no. 5, pp. 1169–1191, 2023.

[2] R. Sista, Z. Hua, P. Thwar, A. Sudarsan, V. Srinivasan, A. Eckhardt, M. Pollack, and V. Pamula, "Development of a digital microfluidic platform for point of care testing," *Lab on a Chip*, vol. 8, no. 12, pp. 2091–2104, 2008.

[3] D. J. Boles, J. L. Benton, G. J. Siew, M. H. Levy, P. K. Thwar, M. A. Sandahl, J. L. Rouse, L. C. Perkins, A. P. Sudarsan, R. Jalili, *et al.*, "Droplet-based pyrosequencing using digital microfluidics," *Analytical chemistry*, vol. 83, no. 22, pp. 8439–8447, 2011.

[4] M. J. Jebrail and A. R. Wheeler, "Let's get digital: digitizing chemical biology with microfluidics," *Current opinion in chemical biology*, vol. 14, no. 5, pp. 574–581, 2010.

[5] E. K. Sackmann, A. L. Fulton, and D. J. Beebe, "The present and future role of microfluidics in biomedical research," *Nature*, vol. 507, no. 7491, pp. 181–189, 2014.

[6] W. H. Minhass, P. Pop, and J. Madsen, "System-level modeling and synthesis of flow-based microfluidic biochips," in *Proceedings of the 14th international conference on Compilers, architectures and synthesis for embedded systems*, pp. 225–234, 2011.

[7] T.-Y. Ho, K. Chakrabarty, and P. Pop, "Digital microfluidic biochips: recent research and emerging challenges," in *Proceedings of the seventh IEEE/ACM/IFIP interna-*

tional conference on Hardware/software codesign and system synthesis, pp. 335–344, 2011.

[8] M. Li, L. Wan, M.-K. Law, L. Meng, Y. Jia, P.-I. Mak, and R. P. Martins, "One-shot high-resolution melting curve analysis for kras point-mutation discrimination on a digital microfluidics platform," *Lab on a Chip*, vol. 22, no. 3, pp. 537–549, 2022.

[9] S. Newman, A. P. Stephenson, M. Willsey, B. H. Nguyen, C. N. Takahashi, K. Strauss, and L. Ceze, "High density dna data storage library via dehydration with digital microfluidic retrieval," *Nature communications*, vol. 10, no. 1, p. 1706, 2019.

[10] D. Liu, Z. Yang, L. Zhang, M. Wei, and Y. Lu, "Cell-free biology using remote-controlled digital microfluidics for individual droplet control," *RSC advances*, vol. 10, no. 45, pp. 26972–26981, 2020.

[11] J. Leipert, M. K. Steinbach, and A. Tholey, "Isobaric peptide labeling on digital microfluidics for quantitative low cell number proteomics," *Analytical Chemistry*, vol. 93, no. 15, pp. 6278–6286, 2021.

[12] B. Seale, C. Lam, D. G. Rackus, M. D. Chamberlain, C. Liu, and A. R. Wheeler, "Digital microfluidics for immunoprecipitation," *Analytical chemistry*, vol. 88, no. 20, pp. 10223–10230, 2016.

[13] J.-L. He, A.-T. Chen, J.-H. Lee, and S.-K. Fan, "Digital microfluidics for manipulation and analysis of a single cell," *International journal of molecular sciences*, vol. 16, no. 9, pp. 22319–22332, 2015.

[14] D. S. Millington and D. S. Bali, "Current state of the art of newborn screening for lysosomal storage disorders," *International Journal of Neonatal Screening*, vol. 4, no. 3, p. 24, 2018.

[15] K. Chakrabarty and T. Xu, *Digital microfluidic biochips: design automation and optimization*. CRC press, 2010.

[16] F. Su, W. Hwang, A. Mukherjee, and K. Chakrabarty, "Testing and diagnosis of realistic defects in digital microfluidic biochips," 2007.

[17] V. Shukla, F. A. Hussin, N. H. Hamid, and N. B. Zain Ali, "Advances in testing techniques for digital microfluidic biochips," *Sensors*, vol. 17, no. 8, p. 1719, 2017.

[18] S. Bhattacharjee, D. Mitra, and B. B. Bhattacharya, "Robust in-field testing of digital microfluidic biochips," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 14, no. 1, pp. 1–17, 2017.

[19] F. Su, S. Ozev, and K. Chakrabarty, "Testing of droplet-based microelectrofluidic systems.," in *ITC*, vol. 46, pp. 1192–1200, Citeseer, 2003.

[20] F. Su, W. Hwang, A. Mukherjee, and K. Chakrabarty, "Testing and diagnosis of realistic defects in digital microfluidic biochips," *Journal of Electronic Testing*, vol. 23, no. 2, pp. 219–233, 2007.

[21] D. Mitra, S. Ghoshal, H. Rahaman, K. Chakrabarty, and B. B. Bhattacharya, "Test planning in digital microfluidic biochips using efficient eulerization techniques," *Journal of Electronic Testing*, vol. 27, pp. 657–671, 2011.

[22] D. Mitra, S. Ghoshal, H. Rahaman, K. Chakrabarty, and B. B. Bhattacharya, "Online error detection in digital microfluidic biochips," in *2012 IEEE 21st Asian Test Symposium*, pp. 332–337, IEEE, 2012.

[23] F. Su, S. Ozev, and K. Chakrabarty, "Concurrent testing of digital microfluidics-based biochips," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 11, no. 2, pp. 442–464, 2006.

[24] T. A. Dinh, S. Yamashita, T.-Y. Ho, and K. Chakrabarty, "A general testing method for digital microfluidic biochips under physical constraints," in *2015 IEEE International Test Conference (ITC)*, pp. 1–8, IEEE, 2015.

[25] M. Elfar, T.-C. Liang, K. Chakrabarty, and M. Pajic, "Adaptive droplet routing for meda biochips via deep reinforcement learning," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 640–645, IEEE, 2022.

[26] M. Elfar, Y.-C. Chang, H. H.-Y. Ku, T.-C. Liang, K. Chakrabarty, and M. Pajic, "Deep reinforcement learning-based approach for efficient and reliable droplet routing on meda biochips," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 4, pp. 1212–1222, 2022.

[27] T.-C. Liang, "Parallel droplet control in meda biochips using multi-agent reinforcement learning," in *International Conference on Machine Learning*, 2021.

[28] T. Kawakami, C. Shiro, H. Nishikawa, X. Kong, H. Tomiyama, and S. Yamashita, "A deep reinforcement learning-based routing algorithm for unknown erroneous cells in dmfbs," in *2023 21st IEEE Interregional NEWCAS Conference (NEWCAS)*, pp. 1–5, IEEE, 2023.

[29] A. Coronato, M. Naeem, G. De Pietro, and G. Paragliola, "Reinforcement learning for intelligent healthcare applications: A survey," *Artificial intelligence in medicine*, vol. 109, p. 101964, 2020.

[30] T. Hester, M. Quinlan, and P. Stone, "A real-time model-based reinforcement learning architecture for robot control," *arXiv preprint arXiv:1105.1749*, 2011.

[31] J. He, J. Chen, X. He, J. Gao, L. Li, L. Deng, and M. Ostendorf, "Deep reinforcement learning with a natural language action space," *arXiv preprint arXiv:1511.04636*, 2015.

[32] B. Jang, M. Kim, G. Harerimana, and J. W. Kim, "Q-learning algorithms: A comprehensive classification and applications," *IEEE access*, vol. 7, pp. 133653–133667, 2019.

[33] C. Dann, Y. Mansour, M. Mohri, A. Sekhari, and K. Sridharan, "Guarantees for epsilon-greedy reinforcement learning with function approximation," in *International conference on machine learning*, pp. 4666–4689, PMLR, 2022.

[34] Z. Zhang and D. Wang, "Adaptive individual q-learning—a multiagent reinforcement learning method for coordination optimization," *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

[35] E. Even-Dar and Y. Mansour, "Convergence of optimistic and incremental q-learning," *Advances in neural information processing systems*, vol. 14, 2001.

[36] S. Bhattacharjee, Y.-L. Chen, J.-D. Huang, and B. B. Bhattacharya, "Concentration-resilient mixture preparation with digital microfluidic lab-on-chip," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 17, no. 2, pp. 1–12, 2018.

[37] F. Su, W. Hwang, and K. Chakrabarty, "Droplet routing in the synthesis of digital microfluidic biochips," in *Proceedings of the design automation & test in Europe conference*, vol. 1, pp. 1–6, IEEE, 2006.

[38] S. Roy, S. Kumar, P. P. Chakrabarti, B. B. Bhattacharya, and K. Chakrabarty, "Demand-driven mixture preparation and droplet streaming using digital microfluidic biochips," in *Proceedings of the 51st Annual Design Automation Conference*, pp. 1–6, 2014.