
MAHMOUD ELFAR, Duke University
ZHANWEI ZHONG, Duke University
ZIPENG LI, Duke University
KRISHNENDU CHAKRABARTY, Duke University
MIROSLAV PAJIC, Duke University

A digital microfluidic biochip (DMFB) is an attractive technology platform for various biomedical applications. However, a conventional DMFB is limited by: (i) the number of electrical connections that can be practically realized, (ii) constraints on droplet size and volume, and (iii) the need for special fabrication processes and the associated reliability/yield concerns. To overcome the above challenges, DMFBs based on a micro-electrode-dot-array (MEDA) architecture have been proposed and fabricated recently. Error recovery is of key interest for MEDA biochips due to the need for system reliability. Errors are likely to occur during droplet manipulation due to defects, chip degradation, and the uncertainty inherent in biochemical experiments. In this paper, we first formalize error-recovery objectives, and then synthesize optimal error-recovery protocols using a model based on Stochastic Multiplayer Games (SMGs). We also present a global error-recovery technique that can update the schedule of fluidic operations in an adaptive manner. Using three representative real-life bioassays, we show that the proposed approach can effectively reduce the bioassay completion time and increase the probability of success for error recovery.

CCS Concepts: • Computer systems organization → Embedded software; System on a chip; • Mathematics of computing → Markov processes; • Theory of computation → Formalisms;

Additional Key Words and Phrases: Digital microfluidics, controller synthesis, lab-on-chip, micro-electrode-dot-array (MEDA), formal methods, stochastic games

ACM Reference format:
https://doi.org/0000001.0000001
1 INTRODUCTION

Digital microfluidics enables the manipulation of droplets of picoliter volumes under program control based on the principle of electrowetting-on-dielectric (EWOD) [10]. Digital microfluidic biochips (DMFBs) are revolutionizing point-of-care diagnostics [20], high-throughput DNA sequencing [23], drug discovery [2], and environmental toxicity monitoring [6]. However, today’s DMFBs suffer from several limitations, mainly: (i) inability to vary droplet volume in a fine-grained manner, (ii) the lack of integrated sensors for real-time detection, and (iii) the need for special fabrication processes and the associated reliability/yield concerns. To overcome the above limitations, a micro-electrode-dot-array (MEDA) architecture has been proposed recently [7, 12, 13]. MEDA is based on the concept of a sea-of-micro-electrodes with an array of identical basic microfluidic unit components called microelectrode cells (MCs) [12], as illustrated in Fig. 1. Each MC consists of a microelectrode and a control/sensing circuit. MEDA allows microelectrodes to be dynamically grouped under program control to form a micro-component (e.g., mixer or diluter) that can perform different microfluidic operations on the chip. Prototypes of MEDA-based biochips have been fabricated using TSMC 0.35 μm CMOS technology.

A major obstacle that impedes reliable microfluidic operations on MEDA biochips is the lack of adaptive and efficient techniques that can facilitate recovery from unexpected errors. Faults in MEDA biochips may arise during bioassay execution. For example, excessive actuation voltage may lead to electrode breakdown and charge trapping [5], and DNA fouling may lead to the malfunction of multiple electrodes in the biochip [21]. Such faults may eventually result in errors (e.g., a splitting operation with unbalanced droplets), which can adversely impact the correctness of fluidic operations.

In order to ensure robust fluidic operations and high confidence in the outcome of biochemical experiments, error-recovery techniques have recently been proposed [8, 11, 17, 24]. Zhao et al. [24] proposed an efficient control-path design method based on error-propagation estimates for fluidic operations. Luo et al. [17] presented a cyberphysical approach that uses sensor data at intermediate checkpoints to dynamically reconfigure the biochip. Jaress et al. [11] proposed a roll-back-based error-recovery technique that leverages a virtual topology to quickly obtain resynthesis results.

All the above techniques are targeted at conventional DMFBs, therefore they can not fully exploit the advantages specific to MEDA-based biochips (e.g., real-time droplet sensing). For MEDA
biochips, Li et al. [15] recently presented an error-recovery strategy based on local adaptation (i.e., “local recovery”), and analyzed it using probabilistic timed automata (PTA). A control flow was also proposed to connect local recoveries with global recovery. However, despite its benefits, this method has several shortcomings:

- It uses static error-recovery protocols for local errors, i.e., the error-recovery protocols are identical for the same type of errors irrespective of when or where on the chip they occur. This approach is not efficient for minimizing the recovery overhead, e.g., sample cost and chip-area impact.
- The solution in [15] makes an unrealistic assumption that resources are always available for local recovery, e.g., there is always a sufficient number of fluidic modules for re-execution of operations. In practice, however, the available resources must be considered in making decisions about local recovery.
- In [15], the same recovery time is assigned for each operation, which is not efficient in minimizing the time spent on error recovery. The recovery-time assignment must consider various factors, e.g., the type and the extent of detected errors, to determine the recovery time for each detected local error.
- Finally, [15] uses all the assigned recovery time for local errors, and it does not consider the likelihood that the local recovery can be successfully completed using less recovery time. Thus, a more adaptive synthesis technique is needed to derive online synthesis results to guide execution of the remaining microfluidic operations.

To overcome the above drawbacks, this paper advances error-recovery by exploring (i) a flexible error-recovery solution derived using formal methods, and (ii) adaptive online synthesis based on real-time sensing results. The proposed approach provides error tolerance with significantly reduced recovery cost and higher probability of success. The main contributions of this paper are:

- We model the error-recovery procedure using Markov decision processes (MDPs). We also describe the platform as another MDP, composing a unified model using stochastic multi-player games.
- We formalize the error-recovery objectives and use them to synthesize optimal error-recovery protocols for local recovery. The system behaviors under the optimal protocols are then analyzed.
- We describe an efficient method to determine available recovery resources for each detected error. Based on the available resources, we dynamically assign the recovery time for different local errors.
- We present an adaptive online synthesis flow to recompute new schedules, module placements, and droplet routes on-the-fly in response to errors.

The rest of the paper is organized as follows. Section 2 presents an overview of MEDA biochips and formal methods that are relevant to this work. In Section 3, we present the problem formulation before describing the synthesis of error-recovery protocols for local errors in Section 4. Section 5 presents an adaptive online synthesis technique for global error recovery. Experimental results on real-life benchmarks, as well as comparison with [15], are presented in Section 6. Finally, Section 7 concludes the paper.

2 PRELIMINARIES

In this section, we provide an overview of digital microfluidics and MEDA, as well as experimental results that illustrate the need for probabilistic modeling of MEDA operations. We then provide an introduction to the formal methods used for modeling, design and analysis of stochastic systems, which are utilized for the synthesis of optimal error-recovery protocols in Section 4.
2.1 Digital Microfluidics and MEDA

A DMFB is able to manipulate and move picoliter droplets containing biological samples on a two-dimensional electrode array. MEDA extends this basic architecture by adding more flexibility [12]. Furthermore, the size of the microelectrodes can be 10 times smaller (e.g., 100 µm in length) than conventional electrodes. The MEDA biochip consists of microelectrode cells (MCs). Each MC includes a microelectrode, an activation circuit, and a sensing circuit [13].

2.1.1 Sensing Scheme in MEDA. In contrast to conventional DMFBs, real-time sensing can be achieved on MEDA biochips. MEDA biochips can detect the property (droplet-property sensing) and the location (droplet-location sensing) of on-chip droplets. Sensing results are presented in the form of a sensing map. Fig. 2 shows the diagram of the sensing circuit in a MEDA biochip. The parasitic capacitance with and without droplets between the top reference electrode and the bottom microelectrode is defined as $C_{\text{Drop}}$ and $C_{\text{Empty}}$, respectively. During droplet-location sensing, droplet precharge circuit first charges the parasitic capacitor. Transistor $M_{N1}$ is then turned on to discharge the parasitic capacitor. Due to the difference between capacitances $C_{\text{Drop}}$ and $C_{\text{Empty}}$, $INV$ outputs different voltage levels. Similar to droplet-location sensing, droplet-property sensing utilizes the capacitance difference between different types of droplets to generate different charging and discharging time.

Sensing on MEDA biochips can provide users with detailed information about the outcomes of on-chip operations. Therefore, errors can be detected in real-time, and error-recovery techniques can then be invoked for error correction.

2.1.2 Outcome Classification. As described in [15], the outcomes of fluidic operations can be experimentally classified into three categories: minor error, major error, and no error. The classification is based on the level of completeness (LOC) of an operation. The LOC measures the extent to which an operation is complete within a pre-specified window, compared with the ideal case (e.g., a uniform mixing and a balanced splitting), and it ranges between 0 and 1 — a larger LOC represents a higher degree of completeness. Here, we use formulas from [15] to calculate the LOC for every operation.

To obtain outcome probabilities for mixing and splitting operations, we performed 100 mixing and 100 splitting operations on a MEDA biochip. The corresponding LOC for each operation was
obtained and the distribution of LOCs for mixing and splitting operations are shown in Fig. 3. Based on Fig. 3, the probabilities that the outcome of a mixing operation is a major error, minor error, and no error are 0.05, 0.13, and 0.82, respectively. Also, the probabilities that the outcome of a splitting operation is a major error, minor error, and no error are 0.09, 0.23, and 0.68, respectively.

2.2 Formal Modeling of Stochastic Processes

2.2.1 Markov Decision Processes (MDPs). MDPs are a formalism widely used to model systems that exhibit both stochastic and nondeterministic behaviors. MDPs are similar to Finite-State Machines, with standard guards, specifying the logical conditions that enable transitions [1], and nondeterministic actions/events that can be used in addition to probabilistic transitions. Stochastic behaviors capture scenarios when a system randomly, with predefined probability distributions, evolves from one state to another [19]. In contrast, nondeterministic behavior means that the system can evolve to the next state via any enabled transition, in a way that is unknown. Intuitively, nondeterminism captures choices that a system (e.g., controller or environment) could make; in every state, once a nondeterministic choice is made, the next state is selected in a probabilistic manner, as in Discrete-Time Markov Chains (DTMC).

Formally, an MDP is specified as a tuple $M = (S, A, \delta, s_0)$, where: (a) $S$ is a finite set of states and $s_0 \in S$ is the initial state; (b) $A$ is a finite set of actions; and (c) $\delta : S \times A \times S \rightarrow [0, 1]$ is a transition probability function such that for all $s \in S$ and $a \in A$, it holds $\sum_{s' \in S} \delta(s, a, s') \in \{0, 1\}$. Note that if action $a$ is enabled in $s$, no matter if it is active or inactive, the sum should be equal to 1; otherwise, it is equal to 0. We use $A(s) \subseteq A$ to denote the set of enabled actions in $s$. For any state $s \in S$, it is required that $A(s) \neq \emptyset$. When there is more than one action available in state $s$ (i.e., $|A(s)| > 1$), a nondeterministic choice should be made; on the other hand, if for all $s$ it holds that $|A(s)| \leq 1$, then the MDP is effectively a DTMC. The system evolution starts in $s_0$. At each step, the system moves from $s$ to $s'$ with probability $\delta(s, a, s')$ if there exists a transition $s \xrightarrow{g,a} s'$ such that the guard $g$ holds and action $a$ is active. Note that this transition is probabilistic if $\delta(s, a, s') \in (0, 1)$ or deterministic if $\delta(s, a, s') = 1$.

2.2.2 Model Composition. When modeling complex systems, it is more favorable to use a number of MDPs to capture the behavior of different system components instead of adopting a monolithic approach. Model composition can then be used to merge those models after defining the synchronization rules among them. Let $M_1 = (S_1, A_1, \delta_1, s_1)$ and $M_2 = (S_2, A_2, \delta_2, s_2)$ be two
To map a bioassay to a MEDA biochip, synthesis techniques are required to bind the assay operation to on-chip resources and generate an optimized schedule of fluidic operations. The input bioassay is modeled by a sequencing graph, which denotes the dependencies between fluidic operations. As discussed in Section 1, errors may occur during the execution of bioassays on a MEDA biochip. A MEDA biochip is said to have an error if its operation does not match its specified behavior. In this work, we assume that chips have been carefully tested before they can be used for bioassay execution, i.e., we assume that droplets will never be stuck during their transportation and droplet dispensing can always be successfully achieved. However, some manufacturing defects may be latent, and they may produce errors during field operation. Thus, we focus on developing
error-recovery approaches for online errors, i.e., mixing and splitting errors during bioassay execution.

For a given bioassay, we use on-chip sensing to evaluate the quality of output droplets of each mixing and splitting operation. Recall that MEDA can provide real-time sensing results for on-chip operations, and thus the time required for outcome evaluation is negligible. Once an error is detected, a local recovery approach is desired to recover from the error. Local recovery for a specific error refers to adaptation actions that are invoked to maximize probability of success for the operation that was affected by that error. However, local error recovery requires extra chip area and consumes time, which may result in a conflict with the original schedule [11]. Consequently, we also focus on global error recovery to efficiently coordinate local-recovery procedures for the complete bioassay. The focus of global recovery is on (i) dynamic assignment of resources used for local recoveries, and (ii) generating new synthesis results (i.e., operation scheduling and module placement) to reduce the interference between error-recovery procedures and bioassay execution. The relationship between local and global error recovery is shown in Fig. 4. The formal models reviewed in Section 2, and described in more detail in Section 4, allow us to make informed decisions about the local recovery procedures, and hence they also play a role in efficient global recovery.

The objective of this work is to develop an error-recovery strategy that can minimize the impact on bioassay completion time and maximize the probability of successful error recovery. The strategy includes approaches for both local recovery (for erroneous operations) and global recovery (for the complete bioassay). For local recovery, we introduce an optimal recovery approach that maximizes the probability of successful recovery based on available resources. For global recovery, we introduce an online synthesizer to seamlessly connect local recovery with global recovery.

3.1 Strategy Overview

The proposed error-recovery strategy (Fig. 5) contains two major components: (i) online synthesizer, which is used to dynamically adjust operation scheduling and module placement, and (ii) local error recovery model that, based on available resources, provides an optimal local-recovery protocol.
These components seamlessly coordinates with each other for the implementation of the proposed error-recovery strategy on the hardware platform.

Details of the error-recovery model and online synthesizer are presented in Section 4 and Section 5, respectively. The error-recovery model is used offline to generate optimal error-recovery protocols for any level of available resources. Then, all these optimal error-recovery protocols are stored in a lookup table. At runtime, if an error is detected during the execution of a bioassay, the online synthesizer first performs an analysis on available resources and then obtains the optimal recovery protocol from the constructed lookup table. Finally, the online synthesizer dynamically adjusts synthesis results to avoid interference between error-recovery protocols and the execution of following operations.

4 MODELING AND SYNTHESIS OF LOCAL ERROR-RECOVERY PROTOCOLS

In this section, we introduce a method to design optimal error recovery protocols for any level of available system resources (e.g., time for recovery, mixers). We start by constructing MDP models of the error recovery process for various fluidic operations. These models, along with an abstracted model of the platform, are obtained from the experimental data presented in Section 2.1, and then integrated using model composition to obtain an SMG model of the system. Finally, we formalize the error-recovery process objectives and use them to automatically synthesize optimal error-recovery protocols.

4.1 Error-Recovery Model

Consider a bioassay that includes a combination of mixing, splitting and dilution operations. Fig. 6 shows the data flow structure of the error-recovery process, which forms the basis of our formal model. The model captures the control execution as well as the probabilistic behavior of the platform. When a bioassay operation is performed, the resource manager updates the amount of resources that can be allocated for error recovery (i.e., available time $r_{time}$, mixers $r_{mix}$, and backup droplets $r_{cbu}$), while the platform provides the information about the operation outcome. Based on this information, the error-recovery model identifies both optimal actions and resources required to recover from the error.

4.1.1 Mixing Error-Recovery Model ($M_{mix}$). We start by considering the error-recovery model $M_{mix}$ for mixing operations (Fig. 7). Initially, the model moves to Check state, where the process outcome is checked. This outcome falls into one of three categories as shown in Fig. 3, namely
Fig. 6. Data flow structure of the error-recovery model.

Fig. 7. Error-recovery model for mixing ($M_{mix}$).

successful, unsuccessful with minor error or with major error, upon which the model moves to the corresponding state. For successful outcome, the model returns to Start, signaling the action doNothing. Otherwise, in Minor Error or Major Error states, four actions are available to choose from.

The first three actions are to reuse both the same droplet and mixer ($a_{mix}$), to move the droplet to a new mixer ($a_{mov}$), and to use a backup droplet to complete the operation ($a_{cbu}$). Each of these actions is guarded by the availability of the required resources, and further ensures that such resources are consumed by synchronizing with the resources model (described in Section 4.1.4). Depending on the selected action, the model progresses to Mix, Move or Call BU state, before returning to Check by accordingly updating the variable action. The fourth action $a_{rsc}$ captures rescheduling of the mixing operation, when the process aborts by moving from Fail to Start.

4.1.2 Splitting Error-Recovery Model ($M_{split}$). Similar to the mixing operation, an error-recovery protocol for a splitting operation starts by reading its outcome. The process aborts by moving from Check to Start states if the operation is successful. Otherwise, it moves to either Minor Error or Major Error states, from which it can proceed with $a_{merge}$, $a_{reroute}$ or $a_{rsc}$ actions to Merge, Reroute or Schedule states, respectively. While actions $a_{merge}$ and $a_{reroute}$ request merging the
defective droplets or rerouting to a new splitter before any further splitting attempts, the action $a_{rsc}$ requests rescheduling. The variable $action$ is used to store the decision made upon synchronizing over $return$ action.

4.1.3 Dilution Error-Recovery Model. In a dilution operation, a droplet first undergoes a mixing operation, after which it is split into two droplets. Hence, we construct the dilution error-recovery model $M_{dilute}$ as $M_{mix}$ followed by $M_{split}$.

4.1.4 Resources Model. Each error-recovery action is associated with specific resource demands, as shown in Table 1. We model these resources as DTMCs from Fig. 9 with: $M_{time}$ tracking time through $r_{time}$ variable, $M_{mixers}$ tracking the number of mixers $r_{mix}$, $M_{drop}$ tracking the number of backup droplets $r_{cbu}$, $M_{actMix}$ tracking current mixer condition through $mixer$ variable, and $M_{actDrp}$ tracking current droplet condition $droplet$. All five variables are initialized by receiving init action, triggered by $M_{plat}$ (Fig. 10). These variables are also updated by one or more of the actions $a_{mix}$, $a_{mov}$, $a_{cbu}$, $a_{mr}$, and $a_{rr}$, triggered by $M_{mix}$ (as in Fig. 7) or $M_{split}$ (Fig. 8). The action use, triggered by $M_{plat}$ (Fig. 10, described in Sec. 4.1.5), marks the current droplet and mixer as used by assigning the value isOld to both droplet and mixer variables, while actions $a_{mov}$ and $a_{cbu}$ set the variables back to isNew.

4.1.5 Platform Model ($M_{plat}$). Since our focus is on the local error-recovery process, it is imperative that the system model captures two fundamental aspects — operation outcome and resources availability. In a typical MEDA biochip, on-chip sensors provide real-time measurements of a process outcome. Being unknown at design stage, we assume that the outcome follows a uniform probability distribution, an assumption supported by the experimental results presented in Section 2.1. Nevertheless, one limitation of the aforementioned results is that they only characterize the outcome probability of an operation at its first attempt. Hence, in Table 2 we augment these results with a model of how the error type of last trial and droplet and mixer conditions may influence the outcome probability of a recovery operation, mostly due to mixer degradation as discussed in [9, 18]. We assume that the probability of the outcome to be successful is the highest when the droplet and mixer used are both new. Similar model is used for splitting operations.
Fig. 9. Model of resources.

Table 1. List of Corrective Actions and Associated Costs

<table>
<thead>
<tr>
<th>Operation</th>
<th>Action</th>
<th>Time</th>
<th>Locations</th>
<th>Droplets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mixing</td>
<td>Retry Mixing</td>
<td>2 s</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Move Droplet</td>
<td>2 s</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Call Backup</td>
<td>2 s</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Reschedule</td>
<td>0 s</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

| Splitting | Merge             | 1 s  | 0         | 0        |
|           | Move Droplet      | 1 s  | 1         | 0        |
|           | Reschedule        | 0 s  | 0         | 0        |

Table 2. Probability distributions over the possible outcomes of mixing and splitting process.

<table>
<thead>
<tr>
<th>Droplet</th>
<th>Mixer</th>
<th>Previous Outcome</th>
<th>Mixing Outcome</th>
<th>Splitting Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>$p_s$</td>
<td>$p_{f1}$</td>
</tr>
<tr>
<td>New</td>
<td>New</td>
<td>N/A</td>
<td>0.820</td>
<td>0.130</td>
</tr>
<tr>
<td>New</td>
<td>Used</td>
<td>minor error</td>
<td>0.656</td>
<td>0.248</td>
</tr>
<tr>
<td>New</td>
<td>Used</td>
<td>major error</td>
<td>0.164</td>
<td>0.232</td>
</tr>
<tr>
<td>Used</td>
<td>New</td>
<td>minor error</td>
<td>0.492</td>
<td>0.367</td>
</tr>
<tr>
<td>Used</td>
<td>New</td>
<td>major error</td>
<td>0.246</td>
<td>0.209</td>
</tr>
<tr>
<td>Used</td>
<td>Used</td>
<td>minor error</td>
<td>0.328</td>
<td>0.485</td>
</tr>
<tr>
<td>Used</td>
<td>Used</td>
<td>major error</td>
<td>0.082</td>
<td>0.255</td>
</tr>
</tbody>
</table>

Fig. 10 shows the MDP $M_{\text{plat}}$ that captures the probabilistic behavior of the platform by abstracting the scheduler, controller, sensors and biochip. The model moves from Start to Do Operation state when an operation is being processed. The transition from Do Operation to Check is associated with the init action, which synchronizes with the resources model (from Fig. 9) to initialize the available resources. The transition exiting Check state is sampled from a uniform distribution characterized by $p_s$, $p_{f1}$ and $p_{f2}$. These probabilities, rather than being constant, depend on the variables droplet, mixer, and err (from Table 2), where err denotes the last type of error. Any of
these three probabilistic transitions triggers the action use, marking both the droplet and mixer as used, and assigns a new value to err.

If the sampled outcome leads to either Minor Error or Major Error, the model proceeds to Error Recovery. The next state is defined according to the action defined by \( M_{mix} \) or \( M_{split} \) through the variable action. Any action other than rescheduling leads to Do Action state and subsequently back to Do Operation state, where the operation is repeated after the corrective action is done. In contrast, if action = doResch, the model moves to Reschedule state, which marks unsuccessful attempt to recover from the error. On the other hand, if the sampled outcome leads to Success, the operation ends. While \( M_{plat} \) in Fig. 10 models a single bioassay operation, for a sequence of operations the model can be augmented with the grayed extension (Fig. 10) to execute in a loop parsing a list of operations.

### 4.2 Model Composition

The aforementioned models present the components and functionalities of a MEDA-based system. The overall system model is obtained by composing those models along with a synchronization model \( M_{sync} \) from Fig. 11, to ensure a turn-based synchronization — i.e., that only one action is triggered at a time. This is achieved using the variable player that indicates which model is allowed to execute actions. In addition, all transitions of models capturing error-recovery process (i.e., \( M_{mix} \) and \( M_{split} \)) are guarded by player = erp, while all transitions of the platform model \( M_{plat} \) are guarded by player = plat. All other models, capturing available resources, are reactive as they do not trigger any actions, and thus none of the player-based global guards are needed.

As the error-recovery protocol is triggered after a platform error occurs, \( M_{sync} \) is initialized with player = plat, i.e., \( M_{plat} \) progresses first. Whenever the platform model decides on the operation (i.e., \( init \) is triggered — see Fig. 10), \( M_{plat} \) is halted (i.e., player = erp). Then, depending on the selected operation, either \( M_{mix} \) or \( M_{split} \) becomes active. Similarly, when an operation either succeeds or fails (i.e., \( return \) is triggered — see Fig. 7 and Fig. 8), the control goes back to the platform model \( M_{plat} \).

The overall system model can be established through parallel composition. If initial values for the resources are assumed to be fixed then only one source of nondeterminism exists, and hence the system model is an MDP. However, synthesized protocols based on this assumption are only
optimal for that specific number of available resources. We relax this assumption by allowing \( M_{plat} \) to nondeterministically select the initial (i.e., available) resources. Thus, the overall system model is an SMG, denoted by \( G_{sys} \), and defined as

\[
G_{sys} = M_{plat} \parallel M_{mix} \parallel M_{split} \parallel M_{time} \parallel M_{mixers} \parallel M_{drop} \parallel M_{actMix} \parallel M_{actDrp} \parallel M_{sync}.
\]

Finally, the game is played between the error-recovery process, denoted by \( erp \), and the platform, denoted by \( plat \).

### 4.3 Strategy Synthesis

In order to synthesize optimal strategies for error recovery, we use the composed SMG model \( G_{sys} \) to find an error-recovery policy that satisfies a specification capturing the desired optimality objective. We define the objective (i.e., specification) of the optimal protocol as follows: *Given a set of resources, find the recovery action that maximizes the probability that the process is eventually successful.* This objective can be formally expressed with the rPATL query [3]

\[
ϕ := \langle\langle erp \rangle\rangle P_{max}=? \ [ \varphi := ◇ Succeed ],
\]

where ◇ is the temporal operator *eventually*. Hence, the synthesis problem aims to find the optimal strategy \( π^* \) such that

\[
G_{sys}^{π^*,σ} = \phi \quad ∀σ ∈ Σ,
\]

where Σ is the set of all possible opponent (i.e., platform) strategies. Although the query \( ϕ \) results in a single value for \( P_{max} \), the synthesis problem seeks the strategy that maximizes the probability \( P \) for every possible opponent strategy \( σ ∈ Σ \), i.e., for every possible assigned resources, resulting in an optimal error-recovery protocol.

We use PRISM-games [4] to implement \( G_{sys} \) along with the query \( ϕ \). The maximum probability of satisfying \( ϕ \) at state \( s \) can be found using value iteration algorithm [3], which is implemented in PRISM-games. For the obtained optimal recovery policies, in Section 6.1 we investigate the relationship between several attributes, such as the effect of resource availability on the probability of success for these operations.

## 5 GLOBAL ERROR RECOVERY

The local error-recovery protocols, derived offline using the techniques from Section 4, depend on the upper limit on the recovery time and the set of available resources that can be used for recovery. However, to adaptively invoke error recovery for the bioassay, we need a global error-recovery technique that can tackle two key problems. First, to determine an optimal error-recovery policy, the local error recovery model must have knowledge of the available resources, e.g., the number of fluidic modules and backup droplets, as well as the time available for recovery. Second, multiple errors can occur at different locations on the chip at nearly the same time, and the error-recovery procedures for these errors can be intertwined through resource sharing or droplet-path overlap.
Therefore, there is a need to dynamically generate new schedules and module placements for error recovery and other bio-protocol-related operations, such that the adverse impact of error-recovery procedures on protocol execution is minimized. In this section, we first describe how we determine resource availability. Next, an online synthesis approach is presented to dynamically generate new schedules and module placements in response to local-recovery decisions.

5.1 Inputs to Local Error Recovery
Recall that the local error recovery requires information about the number of backup droplets, the number of fluidic modules available, and the maximum allowable recovery time. We next describe how these parameters are determined.

5.1.1 Available Backup Droplets. For a splitting or a dilution operation, if only one of its output droplets is used as an input for an immediate successor, the other (redundant) droplet is referred to as a backup droplet for possible error recovery. In addition, dispensing operations can be scheduled for execution as early as possible and extra droplets can be stored on the biochip as backup. Unused backup droplets are sent to the waste reservoir upon the completion of the bioassay.

The number of backup droplets available for an operation can be determined from the sequencing graph. In a sequencing graph, each node represents an operation. Let \( BU(O_i) \) denote the number of backup droplets generated by operation \( O_i \). Suppose operation \( O_j \) is an immediate successor of the \( N \) operations denoted by \( O_{j_1}, O_{j_2}, \ldots, O_{j_N} \). Then, the number of available backup droplets for the operation \( O_j \) is \( \min(BU(O_{j_1}), BU(O_{j_2}), \ldots, BU(O_{j_N})) \), since operation \( O_j \) requires one droplet from each of its immediate predecessors.

5.1.2 Available Fluidic Modules. A fluidic module is a group of microelectrodes on a MEDA biochip that can be configured to perform a type of operation (e.g., mixing). In order to determine the number of fluidic modules available for local error recovery, we use a search algorithm based on the notion of a forbidden set introduced in [16]. A forbidden set refers to a set of locations where new fluidic modules cannot be placed.

If an error occurs for an operation, the algorithm attempts to place appropriate fluidic modules on the biochip so that they can be used for local error recovery. The forbidden set is used to avoid placement conflicts with other operations. This algorithm terminates when either (1) a total of \( N \) fluidic modules are placed on the biochip, where \( N \) refers to the maximum number of fluidic modules designated for use in local error recovery (determined by the local error-recovery model), or (2) not enough space is available to place a fluidic module. When this algorithm terminates, it provides the number and the locations of fluidic modules for local error recovery.

The pseudo-code for determining the number of fluidic modules is shown in Fig. 12. The parameter \texttt{fluidic\_modules} is a container, which stores the placements of available fluidic modules (line 1). According to the current module placement \( MP \), we use the function \texttt{getPlacementSet()} to obtain the set of available locations \( PS \) for a fluidic module (line 2). If \( PS \) is not empty, i.e., there is sufficient space to place a new fluidic module, the function \texttt{findBestPlacement()} is used to find the module placement \( MP_{\text{new}} \) with minimum cost (lines 3–4). The cost is defined as the Manhattan distance [16] between the target fluidic module and the erroneous fluidic module. The newly identified fluidic-module placement is stored in \texttt{fluidic\_modules} (line 5). Next, we examine the number of fluidic-module placements in the container: (1) if we already have \( N \) fluidic-module placements, the search is stopped; (2) otherwise, the newly found fluidic-module placement \( MP_{\text{new}} \) is added to the on-chip module placement set \( MP \), and we continue the search for another fluidic-module placement based on the updated \( MP \) (lines 6–9). Finally, when the search stops, we delete the newly added fluidic module placements from \( MP \) (lines 12–14) and get the number and the
Input: Module placement $MP$;
Output: Number of fluidic modules $N$ and locations of fluidic modules $LO$;
1: $\text{fluidic\_modules} := \{\}$;
2: $PS := \text{getPlacementSet}(MP, \text{module\_type})$;
3: while not $PS$.empty do
4:     $MP_{\text{new}} := \text{findBestPlacement}(PS)$;
5:     $\text{fluidic\_modules}.\text{append}(MP_{\text{new}})$;
6:     if $\text{fluidic\_modules}.\text{size}( ) = N$ then break;
7:     else
8:         $MP.\text{add}(MP_{\text{new}})$;
9:     end if
10: end while
11: for $MP_{\text{new}}$ in $\text{fluidic\_modules}$ do
12:     $MP.\text{delete}(MP_{\text{new}})$;
13: end for
14: $N := \text{fluidic\_modules}.\text{size}( )$;
15: $LO := \text{fluidic\_modules}$;
16: return $N, LO$;

Fig. 12. Pseudo-code for determining the number of available fluidic modules.

locations of fluidic modules from $\text{fluidic\_modules}$ (line 15). For a MEDA biochip with an $M \times N$ microelectrode array, the computational complexity of this search algorithm is $O(MN)$.

5.1.3 Maximum Recovery Time. When more recovery time is available, additional error-recovery operations can be executed and the probability of success will be correspondingly higher. However, if excessive time is allowed for error recovery, there is increased risk that the bioassay will miss the completion-time deadline. Therefore, it is important to assign a maximum recovery time for the recovery procedure when an error occurs in an operation. Here, we utilize the scheduler from [16] to generate 20 feasible schedules for the remaining operations by varying the recovery time $t_r$ from 1 s to 20 s. As a result, we get 20 completion times corresponding to these 20 schedules. Note that we set the upper limit of $t_r$ to 20 seconds, since there is no significant increase in the probability of success for $t_r > 20$ s for the range of laboratory bio-protocol that we studied (as we illustrate in Section 6). Finally, the maximum recovery time $\hat{t}_r$ for local error recovery is equal to the maximum value of $t_r$ that satisfies the inequality $T(t_r) \leq T_0 + \Delta T_{\text{max}}$, where $T_0$ is the completion time of the bioassay when no error occurs, $\Delta T_{\text{max}}$ is the maximum allowable increase in the completion time due to errors, and $T(t_r)$ denotes the completion time when $t_r$ time is allocated for the error recovery.

We evaluate the CPU time needed for global recovery for three bioassays described in Section 6. The maximum time to generate a schedule for these bioassays is 0.05 s on an Intel Core i3 with a 3.7 GHz CPU and 8 GB memory; thus, generating 20 schedules requires 1 s. To speed up the search for the maximum allowable recovery time, we use binary search to generate fewer schedules and quickly find the largest value of $t_r$ that leads to an admissible schedule.

5.2 Online Synthesis

We now describe how online synthesis can be used to integrate local error recovery with global error recovery, as shown in Fig. 13. When a bioassay is mapped to the biochip, the completion-time deadline is also specified by the user. If errors occur, the completion time of the bioassay will be larger than $T_0$, so we use $T_0 + \Delta T_{\text{max}}$ as an upper bound on the acceptable completion time. If the
completion time $T$ satisfies the condition $T \leq T_0 + \Delta T_{\text{max}}$, the bioassay is deemed to be successful; otherwise, it fails.

We first generate an initial schedule and module placement. All operations are stored in the operation queue in ascending order of start times and retrieved sequentially. Before executing new operations, the controller of the MEDA biochip examines the status of all ongoing local error recoveries and: (1) if any local error-recovery procedure fails (e.g., the recovery time exceeds the maximum allowable recovery time), the bioassay fails; (2) if a local error recovery finishes within the maximum allowed recovery time, the schedule and module placement will be updated to carry out the remaining operations as soon as possible.

Following this step, the controller on the MEDA biochip checks if there is any new error. If a new error occurs, the controller determines the resources and time available for recovery from that error, and obtain the optimal error-recovery policy generated by the techniques described in Section 4. Note that these error-recovery policies are obtained offline (i.e., at design time) and stored in a lookup table, and therefore no time is required for their computation at runtime. To accommodate the obtained optimal error-recovery policy, the schedule is updated appropriately.

6 EXPERIMENTAL RESULTS
In this section, we first evaluate our local error-recovery approach by focusing on the composed model from Section 4. Next, we demonstrate the effectiveness of our global error-recovery technique by presenting results of simulating our technique on three representative real-life bioassays. Finally, using the same benchmarks, we compare our approach with the static protocol proposed in [15], since the latter significantly outperforms other existing methods.

6.1 Model Analysis
Since the optimal synthesized strategy $\pi^*$ resolves the nondeterminism of $erp$, the resulting game $G^{\pi^*}$ is reduced to an MDP where platform controls the available resources. For a specific strategy $\sigma$ of platform, the game $G^{\pi^*,\sigma}$ is reduced to a DTMC. Thus, we use the induced model to study the system under the optimal policy.
To understand the impact of available resources on the error-recovery process, we run experiments on $G_{\pi^{*}, \sigma}$ where $\sigma \in \Sigma$ and $\Sigma$ is a set of finite plat strategies constructed by choosing a finite set of values for each resource, namely

$$\scriptsize r_{\text{time}}^{\text{init}} \in \{0 : 2 : 20\}, \ r_{\text{mix}}^{\text{init}} \in \{0 : 1 : 4\}, \ r_{\text{cbu}}^{\text{init}} \in \{0, 1\}. \ (1)$$

We can then obtain the expected success probability $p_{\max}$ under the optimal policy $\pi^{*}$ and all possible initial sets of available resources $(r_{\text{time}}^{\text{init}}, r_{\text{mix}}^{\text{init}}, r_{\text{cbu}}^{\text{init}})$. We formally describe the query as

$$\scriptsize \pi^{*}, \sigma \max Succeed \forall \sigma \in \Sigma. \ (2)$$

The computational complexity of solving (2) is $O(N_{t}^{3}N_{m}^{3}N_{c}^{3})$, where $N_{t}$, $N_{m}$ and $N_{c}$ are the number of values for time units, mixers, and backup droplets, respectively (e.g., 11, 5 and 2 in (1)). For the set of initial resources from (1), PRISM-games solved the query and computed the optimal local error-recovery policy in 2.68 s on an Intel Core i7 4.0 GHz CPU. Recall that this computation is performed only once offline. Fig. 14 shows the obtained results, illustrating the effect of resource availability on the probability that a mixing operation is successfully performed under the optimal error-recovery protocol. The general trend shows that the more the resources available, the higher the success probability is. Since the mixing process takes 2 s to complete, time increments of 1 s do not result in change in probabilities. With the same available amount of time, the improvement in probabilities between having (0 mixers, 0 droplets) and (3 mixers, 1 droplet) peaks when the time is at least 4 s (5.38%) then takes a downward trend as time is increased. The combination (4 s, 2 mixers, 1 droplet) has the lowest time required to guarantee a success probability of at least 95%. If more time is afforded, (6 s, 2 mixers, 0 droplets) can guarantee approximately the same probability.

6.2 Benefits of the Dynamic Protocol

One significant advantage of our proposed dynamic recovery protocol over the static one from [15] is the higher probability to recover from erroneous operations, especially those residing on the side
Fig. 15. The sequencing graph of an example bioassay.

Fig. 16. The schedule when error (a) never occurs, (b) occurs in $M_4$, (c) occurs in $M_7$ using static protocol, (d) occurs in $M_7$ using dynamic protocol.

branches of a sequencing graph. The reason is that a static protocol follows a fixed policy to recover from erroneous operations, failing to exploit the variance of time and other on-chip resources. On the other hand, a dynamic protocol generates the optimal recovery policy based on the currently available resources. To illustrate this, consider the bioassay protocol shown in Fig. 15, which has a long critical path (from $D_1$ to $S_4$) and a long side branch path (from $D_8$ to $M_5$). The schedule of this particular bioassay when no error happens is shown in Fig. 16(a). Here, the completion-time deadline is 24 s, which means the maximum allowed completion time increment $\Delta t_{\text{max}}$ is 2 s. We discuss the following two cases:

1. If an error occurs in the critical path (e.g., in $M_4$). As shown in Fig. 16(b), the maximum recovery time $\hat{t}_r$ for both the static and dynamic protocols is 2 s. Within this period of time, the
probability of success for local recovery is 0.3 and 0.57 for the static and dynamic protocols, respectively.

(2) If an error occurs in the side branch path (e.g., in $M_7$). Since the static protocol relies on a fixed policy, as shown in Fig. 16(c), the maximum recovery time for $M_7$ to recover from an error is 2 s, and the probability of success for error recovery remains 0.3. However, the dynamic protocol examines the available time and other on-chip resources for error recovery and generate the optimal error-recovery policy. Fig. 16(d) shows that 8 s are used to recover $M_7$, and the probability of success for error recovery significantly increases to 0.9.

6.3 Results for Bioassays

We evaluated the proposed global error-recovery method on three real-life benchmarks shown in Fig. 17, namely CEP, master-mix, and serial-dilution [9]. CEP is a combination of three small bioassays: cell lysis, mRNA extraction and mRNA purification. The experimentally characterized module library for MEDA is presented in [15]. The size of one electrode in a conventional DMFB is equal to a $4 \times 4$ microelectrode array in MEDA biochips. Therefore, a $2 \times 2$ array in the MEDA platform actually represents an $8 \times 8$ microelectrode array. We then set the chip size to be an $8 \times 8$ array.

The error-recovery capability of a MEDA-based biochip can be evaluated based on both the bioassay completion time and probability of success (POS) when errors are detected. In our simulation, we randomly inject up to four errors into each benchmark and simulate 1000 execution scenarios under each configuration. Then, we calculate the average completion time and probability of success under different $\Delta t_{\text{max}}$, where $\Delta t_{\text{max}}$ denotes the maximum allowed bioassay completion time increment. For example, if $\Delta t_{\text{max}}$ is set to 10 s and the completion time for a bioassay without errors is 20 s, then the bioassay completion-time deadline is 30 s.

As shown in Fig. 18, a larger $\Delta t_{\text{max}}$ and a smaller number of inserted errors result in higher probability of success for all three bioassays. Nevertheless, this comes at the cost of a slight increase in the completion time. Therefore, a trade-off between the completion time and the probability of success exists.
when two or more errors occur, it significantly outperforms the static protocol. Finally, from Fig. 19. When only one error occurs, the dynamic protocol shows modest improvement. However, as the probability of success under different $\Delta T_{\text{max}}$ — the results for both protocols are shown in Fig. 19. When only one error occurs, the dynamic protocol shows modest improvement. However, when two or more errors occur, it significantly outperforms the static protocol. Finally, from

To compare the proposed dynamic protocol with the static one from [15], we simulate 1000 times per each number of errors from one to four, where the location at which the errors are injected is randomized. We compare the completion time under the same probability of success, as well as the probability of success under different $\Delta T_{\text{max}}$ — the results for both protocols are shown in Fig. 19. When only one error occurs, the dynamic protocol shows modest improvement. However, when two or more errors occur, it significantly outperforms the static protocol. Finally, from
the experimental data presented in Section 2.1, the probabilities of two or more errors occurring in the CEP, master-mix, and serial-dilution benchmarks are 0.625, 0.947, and 0.986, respectively, highlighting the need to employ the proposed dynamic error-recovery protocol.

7 DISCUSSION AND CONCLUSION

We have presented the first work that can automatically synthesize optimal error-recovery protocols for MEDA biochips. We first model the error-recovery procedure using Markov Decision Processes (MDPs). Along with the abstract model for the platform, we obtain a stochastic multi-player game (SMG)-based system model. We then formalize the error-recovery procedure objectives and use them to generate optimal error-recovery protocols for local recovery. We also propose a global-recovery technique to dynamically (i) assign resources for local recovery and (ii) generate new synthesis results, i.e., operation-scheduling and module-placement results. Although the local recovery plans are optimal, the global scheduling policy is based on heuristics. As part of future work we will focus on deriving more efficient scheduling algorithms. Finally, simulation results from three real-life bioassays demonstrate the effectiveness of the proposed error-recovery technique.

As a biochip ages, the probability that an error occurs in a fluidic operation is more likely to increase over time. Hence, future work can extend the model such that the probability distribution over possible outcomes is dynamically updated, potentially by gathering statistical data from the biochip. In that case, online protocol synthesis can be useful to realize such implementation.

Furthermore, the life of a biochip can be prolonged if less error-recovery operations are executed. The objective of the error-recovery protocol can then be relaxed such that, in the early stages of a bioassay, a number of minor errors are tolerated before performing any local error-recovery attempt. Thus, a balance between minimizing the time required to complete a bioassay and extending the life of the biochip can be achieved.

REFERENCES


Received April 2017; revised May 2017; accepted June 2017