Reconfiguration of Linear Surface Chemical Reaction Networks with Bounded State Change

Robert M. Alaniz* Michael Coulombe† Erik D. Demaine† Bin Fu* Timothy Gomez†
Elise Grizzell* Ryan Knobel* Andrew Rodriguez* Robert Schweller* Tim Wylie*

Abstract

We present results on the complexity of reconfiguration of surface Chemical Reaction Networks (sCRNs) in a model where surface vertices can change state a bounded number of times based on a given burnout parameter k. We primarily focus on linear $1 \times n$ surfaces. Without a burnout bound, or even with an exponentially high bound on burnout, reconfiguration on linear surfaces is known to be PSPACE-complete. In contrast, we show that the problem becomes NP-complete when the burnout k is polynomially bounded in n. For smaller $k = O(1)$, we show the problem is polynomial-time solvable, and in the special case of $k = 1$ burnout, reconfiguration can be solved in linear $O(n + |R|)$ time, where $|R|$ denotes the number of system rules. We additionally explore some extensions of this problem to more general graphs, including a fixed-parameter tractable algorithm in the height m of an $m \times n$ rectangle in 1-burnout, a polynomial-time solution for 1-burnout in general graphs if reactions are non-catalytic, and an NP-complete result for 1-burnout in general graphs.

1 Introduction

A prominent area of research in molecular computation, Chemical Reaction Networks (CRNs), study well-mixed solutions of molecules. Limited by the inherent lack of geometry, the model has important restrictions on its computational power, including no proven capability of error-free computation of logarithm [6] or Turing universality [16]. Specifically, CRNs are capable of computing all semilinear functions [5]. The introduction of a surface and, by extension, geometry, with abstract Surface Chemical Reaction Networks (sCRNs) removes these limitations, and thus has increased computational power. Molecular computing on a surface is an increasingly popular direction in both experimental [4, 18] and theoretical [10, 13] research.

In this paper, we explore a restricted version of the powerful surface CRN model, where each molecule in the system can only change in a reaction a set number of times. We refer to this constraint as *burnout*. Bounding the number of state changes leads to polynomial-time and XP algorithms for many reconfiguration problems that are otherwise PSPACE-complete.

Motivations for the study of problems with burnout include examples such as optimizing limited lifetime biomolecules or modeling redox reactions in which the electron transfer from one chemical species to another increases the cost of further reaction beyond what any other current or future neighbors could afford.

1.1 Previous Work

Surface Chemical Reaction Networks (sCRNs) were introduced in [15] with a simulator provided in [7]. These papers show various constructions such as Boolean circuits and a Cellular Automata simulation.

Another restricted version of sCRNs uses only swap reactions, in which the two species only change position, Example: $A + B \rightarrow B + A$. In [2], the authors show swap reactions are capable of feed-forward computation and provide an analysis of thermodynamic properties of the circuit. Recently, [1] showed that reconfiguration is PSPACE-complete for swap surface CRNs with only 4 species and 3 reactions, and in P with any system of fewer species or reactions. This work also introduces k-burnout surface CRNs and show two important results: that 1-reconfiguration (whether a single cell can change) is NP-complete with 1-burnout and that general reconfiguration is NP-complete with 2-burnout. Burnout is similar to the freezing concept from Cellular Automata [11, 12, 17] and Tile Automata [3], but while freezing is defined as having an ordering on states or a tile never revisiting a state, burnout is a constraint where a cell never reacts more than a fixed number of times. Thus, returning to a previous state is possible, unlike the freezing restrictions.

1D Cellular Automata are capable of Turing computation from [8]. P-completeness of prediction, is this cell in state at time step less than t, for Cellular Automata Rule 110 was shown in [14], implying it is also capable of efficient computation. This problem is also P-complete for a number of Freezing CAs in 2D, while it is always in NL for Freezing 1D CAs [12]. This work also gives a 1D freezing CA, which is Turing universal.

*Department of Computer Science, University of Texas Rio Grande Valley

†Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology

Shape	Burnout	Result	Theorem
$1 \times n$	1	$O(n + \|R\|)$	Thm. 1
$1 \times n$	2	$O(n \cdot \|S\|^2 \cdot \|R\|^4)$	Thm. 2
$1 \times n$	$O(1)$	P for $O(1)$ degree	Thm. 3
$1 \times n$	k (unary)	NP-complete	Thm. 4
$1 \times n$	Unbounded	PSPACE-complete	[15]
Planar	1	$O(\|V\|^{1.5} + \|R\|)$	Thm. 5**
General	1	NP-complete	Thm. 7
$m \times n$	1	NP-complete	[1]‡
$m \times n$	1	FPT in m	Thm. 8‡

Table 1: Comparison of reconfiguration results. For a CRN system, R is the set of rules and S is the set of species. V is the set of vertices for the graph defining the shape. **Non-catalytic rules only. ‡These results are for the problem of 1-Reconfiguration.

1.2 Our Contributions

This work investigates the reconfiguration problem for linear surface CRNs with k-burnout. Our results are outlined in Table 1. We begin in Section 3, where we present a polynomial-time algorithm for 1D 1-burnout. We then increase the burnout number to investigate 1D 2-burnout systems and prove that this is still in P. Following this, we show that for the case of any fixed $k = \mathcal{O}(1)$, there exists an algorithm that has a polynomial runtime. In the terms of parameterized complexity classes, this is the class XP, also known as slice-wise polynomial [9]. We then present an NP-completeness proof for when the burnout is a unary input. This result contrasts PSPACE-completeness known when the burnout is unbounded or exponentially high [15].

After $1 \times n$ lines, we begin investigating 1-burnout in 2D systems in Section 5. We start with the problem of reconfiguration, where we only have non-catalytic rules. We then show that on an arbitrary graph and with all types of rules, the reconfiguration problem is NP-complete. Finally, we study the problem of 1-reconfiguration for bounded-height surfaces, presenting an XP algorithm parameterized by height.

2 Preliminaries

A brief overview of the model and relevant problems.

Surface, Cells and Species. A *surface* for a CRN Γ is an undirected graph G of large size n. The vertices of the surface are also referred to as *cells*. Many of our results deal with $1 \times n$ grid graphs, or *linear* surfaces.

The state of a vertex is representative of a molecular *species* in the system. Chemical *reactions* considered here are bimolecular, as in they occur between two species in neighboring vertices. A rule denoting that neighboring species A and B may react to become C and D is written as $A + B \rightarrow C + D$. This is a **non-catalytic** rule, as both species change. In a **catalytic** reaction, only one of the two species will change, e.g.

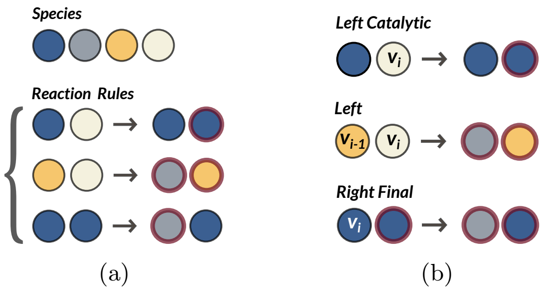

Figure 1: (a) An example sCRN system with 4 species, three rules, and 1 burnout. (b) Rule types used in Figure 2 example. *Note: The red ring outline shows whether the vertex has been "burned out." There is no effect on the reaction rule itself.*

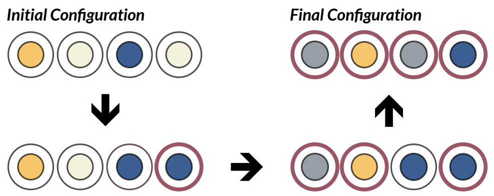

Figure 2: A possible sequence of reactions for the system described in Figure 1

$C + D \rightarrow C + B$, the other used as a catalyst.

A **surface Chemical Reaction Network (sCRN)** consists of a surface, a set of molecular species S, and a set of reaction rules R. A *configuration* is a mapping from each vertex to a species from the set S.

Reachable Configurations. For two configurations I, T, we write $I \rightarrow^1_\Gamma T$ if there exists a $r \in R$ such that performing reaction r on a pair of species in I yields the configuration T. Let $I \rightarrow_\Gamma T$ be the transitive closure of $I \rightarrow^1_\Gamma T$, including loops from each configuration to itself. Let $\Pi(\Gamma, I)$ be the set of all configurations T where $I \rightarrow_\Gamma T$ is true.

Burnout. A limit on the number of changes that can occur in any vertex v_i. In systems that allow catalytic reactions, after this limit has been reached, while v_i will not change again, neighboring species may still use the species in that cell as a catalyst.

Reconfiguration Problem. Given an sCRN Γ and two configurations I and T, is $T \in \Pi(\Gamma, I)$?

1-Reconfiguration Problem. Given an sCRN Γ, configuration I, vertex v, and species s, does there exist a $T \in \Pi(\Gamma, I)$ such that T has species s at vertex v?

3 Algorithms for Constant Burnout

We show that reconfiguration of a linear surface is solvable in polynomial-time when the burnout is one or two.

3.1 1-Burnout Linear Surfaces

In the case of 1-burnout with a $1 \times n$ line, the problem of reconfiguration is solvable in linear time with respect to n and the size of the rule set. As an observation, there are at most six reactions for any vertex, v_i, on a linear surface since a vertex has at most two neighbors. These reactions include the following:

- A left reaction, where vertex v_i reacts with vertex v_{i-1} and both vertices reach their final states.
- A left catalytic reaction, where vertex v_i reacts with vertex v_{i-1} in its initial state to transition vertex v_i to its final state without changing v_{i-1}.
- A left final-catalytic reaction (or left final), where vertex v_i reacts with vertex v_{i-1} in its final state to transition vertex v_i to its final state without changing v_{i-1}.
- A right reaction, where vertex v_i reacts with vertex v_{i+1} and both vertices reach their final states.
- A right catalytic reaction, where vertex v_i reacts with vertex v_{i+1} in its initial state to transition vertex v_i to its final state without changing v_{i+1}.
- A right final-catalytic reaction (or right final), where vertex v_i reacts with vertex v_{i+1} in its final state to transition vertex v_i to its final state without changing v_{i+1}.

Additionally, we also consider when a vertex is in its final state. An example system and sequence of reactions can be found in Figures 1 and 2.

We construct a $7 \times n$ table (Example in Table 2), where each row represents one of the possible reactions, including no reaction, and each column represents the starting configuration's vertices from left to right. For each entry in the table, we see if the reaction exists for that vertex and if the vertex reaches its final state. If both cases are satisfied, place a 1 in the corresponding row, otherwise, place a 0. After all cells are evaluated, we construct a directed graph with edges being directed from column i to column $i+1$ with the following properties for each row entry in column i:

- In final state: edge to every row in column $i+1$ with a 1 except left reaction.
- Left final: edge to every row in column $i+1$ with a 1 except left reaction.
- Left catalytic: edge to every row in column $i+1$ with a 1 except left reaction.
- Left reaction: edge to every row in column $i+1$ with a 1 except left reaction.
- Right final: edge to every row in column $i+1$ with a 1 except left reaction and left final.
- Right catalytic: edge to every row in column $i+1$ with a 1 except left reaction and left catalytic.
- Right reaction: edge only to the row corresponding to left reaction in column $i+1$ if there is a 1.

These edges ensure that no matter which reaction is chosen for a vertex represented by column i, the reaction chosen for the column $i+1$ vertex will be able to perform its reaction either before or after the previous reaction.

Once these edges are defined for every column, the problem is then finding a path from s to t, where s is a vertex that has directed edges to each entry in column 1 and t is a vertex that can be reached from each entry in column n (see Table 2 and Figure 3 for reference). Any path represents a set of rules that can be assigned an ordering to reconfigure all vertices to their final states.

Reaction Type	🟠	⚪	🔵	⚪
In Final State	-	-	-	-
Left	-	1	-	-
Left Catalytic	-	-	-	1
Left Final	-	-	-	-
Right	1	-	-	-
Right Catalytic	-	-	-	-
Right Final	-	-	1	-

Table 2: Turning the example system from Figure 1 into a table of reactions.

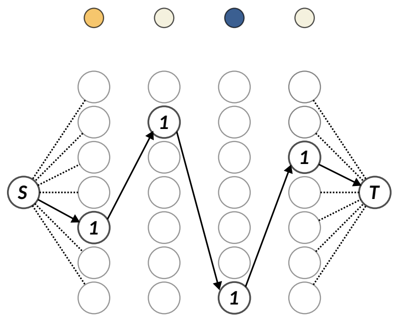

Figure 3: Table 2 as a graph.

Theorem 1 *Reconfiguration in 1-burnout for $1 \times n$ lines is solvable in $O(n + |R|)$ time.*

Proof. We provide proof by induction for the previously described algorithm that solves reconfiguration in $1 \times n$ surfaces. This proof guarantees that any solution from this algorithm constitutes a set of reactions that can be reordered to successfully reconfigure a given initial configuration to its final configuration.

Base case: $n = 2$. Let v_i be the leftmost vertex. Since this vertex does not have a neighbor to its left, there are only 4 reactions we need to consider for this vertex:

1. In final state: vertex v_{i+1} must be in its final state or a left catalytic or left final reaction.
2. Right catalytic: vertex v_{i+1} must be in its final state or a left final reaction.
3. Right final: vertex v_{i+1} must be in its final state or a left catalytic reaction.
4. Right reaction: vertex v_{i+1} must be a left reaction.

If two such reactions exist for each vertex, then a path exists from s to t visiting the vertices in the table that correspond to each reaction. Otherwise, no such path would exist.

Inductive step: let $n = k$. Assume that there is a set of k reactions for vertices $v_1, \ldots, v_k$ that can be reordered to transition all k vertices to their final states.

In order for the reaction chosen for vertex v_{k+1} to be valid, it must not interfere with the k^{th} reaction corresponding to vertex v_k. Consider two cases:

1. Vertex v_k is currently in its final state or reacts with its left neighbor v_{k-1}. Vertex v_{k+1} is never used, so as long as v_{k+1} does not perform a left reaction with v_k, it will not interfere with the k^{th} reaction.
2. Vertex v_k reacts with vertex v_{k+1}. Consider 3 possible reactions for v_k: (1) Right reaction: the only valid reaction for v_{k+1} is a left reaction, (2) Right catalytic: except left or left catalytic, all reactions are valid for v_{k+1}, and (3) Right final: except left or left final, all reactions are valid for v_{k+1}.

If we think of v_k as being column i and v_{k+1} as being column $i+1$, edges are defined from i to $i+1$ in a way that avoid these conflicting reactions. Any other reaction that is chosen for v_{i+1} can always perform its reaction before or after v_i performs its reaction. As a result, any path up to column $i+1$ would represent a set of reactions that can be reordered to transition these $k+1$ vertices to their final states.

Given the initial and final configurations, it takes $O(n)$ time to compare the states. Constructing the table takes $O(|R|)$ time. The path finding algorithm runs in $O(V+E) = O(n+E)$ time. However, the number of edges is a constant factor of the number of vertices, whereas $|R|$ might be exponential in n. Thus, the final runtime for the algorithm is $O(n+|R|)$. $\square$

3.2 2-Burnout Linear Surfaces

Theorem 2 *Reconfiguration of a $1 \times n$ line for surface CRNs with 2-burnout is solvable in $O(n \cdot |S|^2 \cdot |R|^4)$ time.*

Proof. Since we are considering 2-burnout, every cell can only change species twice. This is a cell starting with the initial species, possibly changing to an intermediate species, then finally changing to the target species. It is then possible to track all the possible transitions of a cell in a polynomial sized table. We define the table D with each entry $D(x, s, r_1, r_2)$ being a Boolean indicating if the cells at indices $0, 1, \ldots, x$ can reach their target species using reactions r_1 and r_2 on x, and using s as intermediate species for cell x. (Note, r_1 and s may be null if the cell only reacts once to reach the target species.) The reactions are specific with which neighbor the cell reacts with, left or right. This results in $\mathcal{O}(n \cdot |S| \cdot |R|^2)$ cells of the table.

To compute each entry $D(x, s, r_1, r_2)$, we check if r_1 and r_2 are consistent with cell $x-1$. Meaning, if r_1 reacts with the left neighbor, some entry $D(x-1, s', r_1, r_3)$ or $D(x-1, s', r_3, r_1)$ for any s, r_3 must be true. If r_1 is a catalytic reaction, then the species in cell $x-1$ does not change and must be the initial species, intermediate species, or the target species. We must also be careful with the ordering of the reactions. If r_1 or r_2 reacts with the intermediate species s' of the $(x-1)$th cell, then r_1 must be the second reaction for $x-1$. The run time to compute each cell of the table is $\mathcal{O}(|S| \cdot |R|^2)$.

If any $D(n-1, s, r_1, r_2)$ is true, then the answer to reconfiguration is true. $\square$

3.3 Constant Burnout

In this section, we consider the problem of reconfiguration for a surface CRN with n cells with at most k-burnouts on a $1 \times n$ board.

Theorem 3 *There is an $n^{1+k\log h}$-time algorithm for k-burnout degree-h 1D surface CRN reconfiguration, where each species is in at most h rules.*

Proof. We have a divide-and-conquer approach in our algorithm. A brute force method is used to enumerate all the possible transitions for the median position. The problem is split into two independent problems that can be solved independently.

Let p be the position of the median in a 1D surface CRN. We enumerate all the possible ways to burn out the position p at most k times. Since each species is in at most h rules, we have at most h^k combinations about the list of transitions involved by position p. Let $T(n)$ be the running time to solve the reconfiguration problem. We have the recursion $T(n) = h^k(2T(\frac{n}{2}))$. It brings a solution with $T(n) = h^{k\log n} \cdot n = n^{1+k\log h}$. $\square$

4 Non-constant Burnout on a Line

Here, we show that reconfiguration with k-burnout, where k is part of the input, is NP-hard. Without burnout (no bound on state changes), reconfiguration of a $1 \times n$ line is PSPACE-complete [15], but even with a burnout k given in binary, the problem may not be in the class NP since $O(kn)$ possible reactions could occur, which is exponential in $\log k$. This motivates looking at bounds on state changes that are polynomial in n and further motivates the other algorithms in the paper.

Reduction. We reduce from Vertex Cover (VC) by enumerating all vertices and using them as states on a $1 \times n$ line. A state "walks" back and forth choosing a vertex to add to the cover and crossing off instances it finds. Given a graph $G = (V, E)$ where $V = \{1, 2, \ldots, n\}$ and an edge $e \in E$ is defined as $e = \{v_i, v_j\}$ for $v_i, v_j \in V$ and $i \neq j$. An edge is listed as two states: 34 meaning an edge between vertices v_3 and v_4. Between any two edges we include a spacing state $-$.

Create the line representing the graph with edges in any order: $BS_0 - e_1 - e_2 - \cdots - e_m - E$, where the B state indicates the beginning of the line, E is the end of the line, and S_0 is a special state indicating no vertices are in the vertex cover. Example: $BS_0 - 34 - 13 - 21 - 14 - E$.

Basically, each edge independently and nondeterministically picks the vertex to cover it with both possible rules. Create rules for all $v_i, v_j \in V$ as $v_i + v_j \to v_i' + x$ and $v_i + v_j \to x + v_j'$ where x is an ignored state and the prime state is the chosen vertex for that edge. The spacing states ensure edges do not affect each other. Example: $3 + 4 \to 3' + x$ and $3 + 4 \to x + 4'$.

The S counting state sweeps back and forth k times to choose a vertex to add to the cover and ignores the other states. The S state takes the first picked vertex and removes all duplicates of it while remembering the count. There is a state S^{vertex}_{count} that exists for each vertex and count up to k. Thus, the rules $S_i + v_j' \to S^j_{i+1} + x$ are added for each vertex and count up to k. Example: $S_0 + 3' \to S^3_1 + x$ is used if v_3 is the first vertex added.

Once a vertex transitions to an S^i state, it ignores everything but v_i' states. Meaning it only swaps states, or "walks" in that direction. Thus, all rules $S^i + A \to A + S^i$ is added for any state X that is not v_i, B, or E. For v_i, $S^i_c + v_i' \to S^i_c + x$.

When a S^j_c vertex is next to the B or E states, it can transition to S_c. The rules $B + S^j_c \to B + S_c$ and $E + S^j_c \to E + S_c$ exist for all vertices v_j. This means we have removed all instances of the chosen vertex and can pick a new vertex for the cover.

This requires $O(kn)$ states to handle counting for each vertex. If k is odd, the final configuration, given a k VC exists, is $B - xx - xx - xx - \cdots - S_k E$. If k is even, then the final configuration is $BS_k - xx - xx - xx - \cdots - E$. S_k can not interact with anything. This requires $k+1$ burnout.

Theorem 4 *Reconfiguration of a $1 \times n$ configuration in sCRNs with k-burnout is NP-hard, even when $k < n$, and NP-complete as long as k is polynomial in n.*

Proof. Given a VC with graph $G = (V, E)$ and $k \in \mathbb{N}$, we create a surface CRN system with configuration C and rules R as described above. We define the output configuration D based on the number of edges and parity of k as described. G has a VC of size k if and only if C can reach configuration D with burnout $k + 1$. Note that $k \leq n$ as input from VC, so the number of states and rules in the reduction is polynomial.

Given that the graph G has a k vertex cover, in the sCRN system, the only transitions possible at first are for each edge to pick a vertex to cover it. Then the counting state walks across, increases the count and selects the vertex from the first edge, and that state continues walking and removes any other instance of that vertex. In the best case, all locations but the first and last have changed twice. If this continues, and it always adds the correct vertices, then after k passes only x's are left. S_k does not interact with anything, so nothing else transitions. The k passes and the initial choice requires $k + 1$ burnout.

If the sCRN system ends in the output configuration with x's on every edge state, which can only occur if the k passes chose vertices that appeared in the other edges and were crossed out. Thus, every edge correctly chose the right vertex to cover it so that only k different vertices were used. □

5 Extension to 2D Graphs

As an extension to the 1D case, we now consider reconfiguration and 1-reconfiguration for 2D surfaces. In the case of reconfiguration, we study a restricted version of the problem where all reactions are non-catalytic.

Theorem 5 *Reconfiguration in 1-burnout for a planar graph $G = (V, E)$ is solvable in $O(|V|^{1.5} + |R|)$ time if every reaction is non-catalytic.*

Proof. Given a planar graph $G = (V, E)$, construct a subgraph G' from G such that there is an edge between pairs of vertices if there exists a non-catalytic reaction that transitions both vertices to their final states. Run maximum matching on G'. If all vertices are either matched or in their final state, then reconfiguration is possible. Otherwise, reconfiguration is not possible.

Since non-catalytic reactions transition both vertices to their final states, a vertex must be involved in at most one reaction. Edges represent these non-catalytic reactions between two vertices. As a result, limiting a vertex to one reaction is the equivalent of matching each vertex in G' to at most one other vertex it shares an edge with, which is a perfect matching problem. For planar graphs, this can be solved using a maximum matching algorithm. If any unmatched vertex is not in its final state, then reconfiguration is not possible because this vertex is unable to react.

Constructing G' takes $O(V + |R|)$ time. Running the maximum matching algorithm takes $O(V^{1.5})$ time. A last check of G' for any unmatched vertices that are not in their final state takes $O(V)$ time. Therefore, the runtime is $O(V^{1.5} + |R|)$. □

Corollary 6 *Reconfiguration in 1-burnout for general graphs is solvable in $O(V^4 + |R|)$ time if every reaction is non-catalytic, where V is the number of vertices.*

Proof. Proof follows from Theorem 5. Maximum matching on general graphs runs in $O(V^4)$ time. □

5.1 Arbitrary Graphs with 1-Burnout

We now consider surface CRNs that allow catalytic as well as non-catalytic rules. With this additional rule type, we prove the problem of reconfiguration is NP-complete on an arbitrary graph with 1-burnout.

Theorem 7 *Reconfiguration with 1-burnout of an arbitrary surface in surface CRNs is NP-complete.*

Proof. We reduce from the dominating set problem to sCRN reconfiguration with 1-burnout. Let $G = (V, E)$ be an arbitrary graph and k be an integer parameter.

We need to decide if graph G has a dominating set of size k. Note that a subset $U \subseteq V$ is a dominating set of G if each vertex $v \in V - U$ has $(u, v) \in E$ for some $u \in U$ (vertex u dominates v).

Let $v_1, \cdots, v_n$ be the n vertices of G. We design a surface CRN system. For each edge (v_i, v_j) in E, create two rules $v_i + v_j \to v_i + v'_j$ and $v_i + v_j \to v'_i + v_j$. We introduce k additional species $u_1, \cdots, u_k$. The target configuration is to let each v_i enter v'_i for $i = 1, \cdots, n$ and each u_t enter u'_t. We set up the rules $u_t + v_j \to u'_t + v'_j$ for all $t \le k$ and all $j \le n$.

If graph G has a dominating set of size k, the target configuration is reachable. Assume that $v_{i_1}, \cdots, v_{i_k}$ dominate all the vertices in the graph G. For each v_j with $j \in \{1, \cdots, n\} - \{i_1, \cdots, i_k\}$, it can be transformed into v'_j by a rule $v_{i_s} + v_j \to v_{i_s} + v'_j$. Each v_{i_s} can enter v'_{i_s} by a rule $u_s + v_{i_s} \to u'_s + v'_{i_s}$. Here, the burnout is 1. Similarly, if the target configuration is reachable, there is a dominating set of size k. If the target configuration is reachable, we have at most $v_{i_1}, \cdots, v_{i_h}$ with $(h \le k)$ such that each v_{i_r} enters v'_{i_r} via the type of rule $u_t + v_{i_r} \to u'_t + v'_{i_r}$ as there is only one burnout for each v_i and u_j. Clearly, $v_{i_1}, \cdots, v_{i_h}$ dominate all the other vertices in the graph G.

This is a polynomial-time reduction and membership is known from [1]. □

5.2 1-Burnout 1-Reconfiguration

Theorem 8 *1-Reconfiguration in 1-burnout of a $w \times n$ rectangle for surface CRNs is solvable in $O\left(n \cdot (|S||R|)^{2w} \cdot f(w)\right)$ time.*

Proof. We use a dynamic programming approach similar to that in Theorem 2, defining a table D with Boolean entries $D(x, \vec{s}, \vec{r}, \pi)$, where x is a column index, $\vec{s} = [s_1, s_2, \ldots, s_w]$, $\vec{r} = [r_1, r_2, \ldots, r_w]$, and π a permutation of $[1, w]$. Each $s_y \in S$ is a potential final species of cell (x, y), which changes from its initial species into (x, y) due to reaction $r_y \in R$, and π gives the order in which the reactions occur. As before, r_y specifies which of its up-to-four neighboring cells participated in the reaction, and s_y and r_y may be null if the cell never changes species.

Since only one cell (x_t, y_t) of the target configuration is fixed, the top-level of the dynamic program will be column x_t, and it will symmetrically recurse outwards in both directions, with base-cases at both ends. So, for $x < x_t$ $D(x, \vec{s}, \vec{r}, \pi)$ is true if the cells in columns $0, 1, \ldots, x$ can reach a target configuration in which column x reaches species $\vec{s}$ using reactions $\vec{r}$ occurring in order π, for $x > x_t$ we consider columns $x, x{+}1, \ldots, n{-}1$ instead, and for $x = x_t$ we consider the entire surface.

To compute $D(x, \vec{s}, \vec{r}, \pi)$, say when $x < x_t$, we search for a smaller subproblem $D(x-1, \vec{s}', \vec{r}', \pi')$ which has value true and $(\vec{r}', \vec{r})$ together are a chain of reactions that actually transform columns $x-1$ and x into species $(\vec{s}', \vec{s})$ from their initial species, given that they must occur in relative orders π' and π. Specifically, we can search each possible interleaving of $\pi(\vec{r})$ and $\pi'(\vec{r}')$, simulate the reactions in that order, and verify that the reactions within these two columns can actually be performed and do result in $(\vec{s}', \vec{s})$. Notably, for reactions between columns $x-2$ and $x-1$, we do not need to validate the species in column $x-2$ because the smaller subproblem already performed that validation, and for reactions between columns x and $x+1$, the species in column $x{+}1$ are assumed to be validated later in a larger subproblem. For $x > x_t$, the recursion is symmetric.

For top-level subproblems $D(x_t, \vec{s}, \vec{r}, \pi)$, we only consider $\vec{s}$ that include the fixed target species s_{y_t}, and we search for both $D(x_t - 1, \vec{s}', \vec{r}', \pi')$ and $D(x_t + 1, \vec{s}'', \vec{r}'', \pi'')$ and validate between all three columns $x_t{-}1, x_t, x_t{+}1$ in a similar manner. If any $D(x_t, \vec{s}, \vec{r}, \pi)$ is true, then the answer to 1-reconfiguration is true.

The size of D is $O\left(n \cdot |S|^w \cdot (4|R|)^w \cdot w!\right)$. Computing each entry involves checking $O\left(|S|^w \cdot (4|R|)^w \cdot w!\right)$ subproblems, and each check considers $\binom{2w}{w}$ interleavings of orderings and runs an simulation taking $O(w)$ time. Combined, the total time is $O\left(n \cdot (|S||R|)^{2w} \cdot f(w)\right)$ for a function f only depending on w. Therefore, for constant w, this is polynomial time. □

6 Conclusion

In this paper, we have shown that the reconfiguration problem on $1 \times n$ surface CRNs with k-burnout is in P when $k = 1$ or $k = 2$. To show this, we have given algorithms that output a sequence of reactions to achieve the given configuration. Further, we show that for any $k = \mathcal{O}(1)$, there exists an algorithm that has a polynomial runtime in k. To conclude our investigation of 1-Dimensional surface CRNs, we prove that when the burnout number, k, is part of the input (in unary), the problem of reconfiguration is NP-complete.

Following by exploring 2-Dimensional surface CRNs and showing that a restricted case of 1-burnout reconfiguration can be seen as perfect matching, showing this case of the problem to still be in P. Finishing with a proof that the problem of 1-Reconfiguration in 1-burnout can be solved in polynomial time on a $w \times n$ rectangle when w is constant.

Some of the open questions are then:

- What is the lower bound for a given k-burnout?
- In a rectangle/grid graph, what is the lower/upper bound for k-burnout?
- Most of our complexity is in terms of the size of the surface. Are there interesting results looking at the complexity of other aspects of an sCRN such as states, rules, and burnout?
- We have a direct NP-complete reduction but does there exist an L-reduction for some inapproximability result?

References

[1] R. M. Alaniz, J. Brunner, M. Coulombe, E. D. Demaine, Y. Diomidov, T. Gomez, E. Grizzell, R. Knobel, J. Lynch, A. Rodriguez, R. Schweller, and T. Wylie. Complexity of reconfiguration in surface chemical reaction networks. In *Proc. of the 29th International Conference on DNA Computing and Molecular Programming*, DNA'23, 2023. To appear.

[2] T. Brailovskaya, G. Gowri, S. Yu, and E. Winfree. Reversible computation using swap reactions on a surface. In *Proc. of the International Conference on DNA Computing and Molecular Programming*, DNA'19, pages 174–196. Springer, 2019.

[3] C. Chalk, A. Luchsinger, E. Martinez, R. Schweller, A. Winslow, and T. Wylie. Freezing simulates non-freezing tile automata. In *DNA Computing and Molecular Programming: 24th International Conference, DNA 24, Jinan, China, October 8–12, 2018, Proceedings 24*, pages 155–172. Springer, 2018.

[4] G. Chatterjee, N. Dalchau, R. A. Muscat, A. Phillips, and G. Seelig. A spatially localized architecture for fast and modular DNA computing. *Nature nanotechnology*, 12(9):920–927, 2017.

[5] H.-L. Chen, D. Doty, and D. Soloveichik. Deterministic function computation with chemical reaction networks. *Natural computing*, 13:517–534, 2014.

[6] C. T. Chou. Chemical reaction networks for computing logarithm. *Synthetic Biology*, 2(1):ysx002, Jan. 2017.

[7] S. Clamons, L. Qian, and E. Winfree. Programming and simulating chemical reaction networks on a surface. *Journal of the Royal Society Interface*, 17(166):20190790, 2020.

[8] M. Cook et al. Universality in elementary cellular automata. *Complex systems*, 15(1):1–40, 2004.

[9] M. Cygan, F. V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized algorithms*, volume 5. Springer, 2015.

[10] F. Dannenberg, M. Kwiatkowska, C. Thachuk, and A. J. Turberfield. DNA walker circuits: computational potential, design, and verification. *Natural Computing*, 14(2):195–211, 2015.

[11] E. Goles, D. Maldonado, P. Montealegre, and M. Ríos-Wilson. On the complexity of asynchronous freezing cellular automata. *Information and Computation*, 281:104764, 2021.

[12] E. Goles, N. Ollinger, and G. Theyssier. Introducing freezing cellular automata. In *Cellular Automata and Discrete Complex Systems, 21st International Workshop (AUTOMATA 2015)*, volume 24, pages 65–73, 2015.

[13] R. A. Muscat, K. Strauss, L. Ceze, and G. Seelig. DNA-based molecular architecture with spatially localized components. *ACM SIGARCH Computer Architecture News*, 41(3):177–188, 2013.

[14] T. Neary and D. Woods. P-completeness of cellular automaton rule 110. In *Automata, Languages and Programming: 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part I 33*, pages 132–143. Springer, 2006.

[15] L. Qian and E. Winfree. Parallel and scalable computation and spatial dynamics with DNA-based chemical reaction networks on a surface. In *DNA Computing and Molecular Programming: 20th International Conference, DNA 20, Kyoto, Japan, September 22-26, 2014. Proceedings*, volume 8727, page 114. Springer, 2014.

[16] D. Soloveichik, M. Cook, E. Winfree, and J. Bruck. Computation with finite stochastic chemical reaction networks. *natural computing*, 7:615–633, 2008.

[17] G. Theyssier and N. Ollinger. Freezing, bounded-change and convergent cellular automata. *Discrete Mathematics & Theoretical Computer Science*, 24, 2022.

[18] A. J. Thubagere, W. Li, R. F. Johnson, Z. Chen, S. Doroudi, Y. L. Lee, G. Izatt, S. Wittman, N. Srinivas, D. Woods, et al. A cargo-sorting DNA robot. *Science*, 357(6356):eaan6558, 2017.