Minimization of Deterministic Finite Automata Modulo the Edit Distance

Jakub Michaliszyn
University of Wrocław, Poland

Jan Otop
University of Wrocław, Poland

Abstract

We propose a novel approach to minimization of deterministic finite automata (DFA), in which the DFA is further minimized at the expense of relaxing equality of languages to merely a *similarity*. As the notion of similarity of languages, we consider the edit distance between languages $\mathcal{L}, \mathcal{L}'$, i.e., the minimal number of edits necessary to transform any word from $\mathcal{L}$ to some word from $\mathcal{L}'$ and vice versa.

In this paper we address two problems: minimization up to a predetermined edit distance given in the input, and minimization up to a bounded edit distance, in which there has to be an upper bound on the number of edits, but it is not specified. We show the first problem to be PSPACE-complete and that the second problem is in Σ_2^p, and both NP-hard and coNP-hard. We show that if we limit how many strongly connected components can be visited by a single run (i.e., bounded SCC-depth), the problem becomes NP-complete. We also establish maximal subclasses of DFA over which minimization up to a bounded edit distance can be performed in polynomial time.

Additionally, we provide a succinct overview of alternative metrics for assessing language similarity.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory

Keywords and phrases automata theory, automata minimization, edit distance

Digital Object Identifier 10.4230/LIPIcs.MFCS.2025.77

Funding This work was supported by the National Science Centre (NCN), Poland under grant 2020/39/B/ST6/00521.

1 Introduction

Finite automata are one of the most fundamental models of computation, playing a central role in theoretical computer science and numerous practical applications, including pattern recognition [27, 4], rule-based systems [9], planning [1], language recognition [24] or multi-agent systems [6, 23]. Among finite automata, *Deterministic Finite Automata (DFA)* provide one of the simplest formalisms for representing and reasoning about structured systems, patterns, rules, and processes.

Minimization of DFA is crucial for two reasons: complexity and interpretability. First, in applications involving DFA, it is often advantageous performance-wise to minimize DFA before proceeding with the actual algorithm.

Second, DFA are used as a formalism to approximate more complex models such as neural networks [26, 15]. Such applications operate in environments where inputs are inherently noisy or incomplete. In such cases, it is often sufficient to approximate the behavior of a system rather than achieving perfect fidelity. In this scenario the exact language equivalence is neither feasible nor necessary, and furthermore DFA of smaller size are typically easier to comprehend, which leads to better interpretability of the underlying model.

Furthermore, as DFA already approximate the underlying models, preserving the exact language of a given DFA in the minimization process is not imperative anymore. Instead, to boost state reduction, we can settle for a similarity notion between languages rather than the rigid language equivalence.

Various notions of similarity between languages have been considered. For instance, two languages are similar if their symmetric difference is finite. Minimization of DFA under this similarity notion is called *hyper-minimization* of DFA [10, 18]. However, this is a rather generic notion as it applies to any set of objects.

Another idea, which relies on the fact that languages are sets of words, is to take a distance $d(w, u)$ on words and lift it to languages in the canonical way: the distance between a word w and a language $\mathcal{L}$, denoted by $d(w, \mathcal{L})$, is the minimal $d(w, u)$ over all $u \in \mathcal{L}$. Then, the distance between $\mathcal{L}$ and $\mathcal{L}'$ is the maximum of numbers $d(w, \mathcal{L}')$, $d(w', \mathcal{L})$ over all $w \in \mathcal{L}, w' \in \mathcal{L}'$. This approach allows infinitely many words to be slightly changed, and is more specialized than the finite symmetric difference. We follow the latter approach employing the *edit distance* as the distance d.

The edit distance [17], also called the Levenshtein distance, between words w, w' is the minimal number of single letter insertions, deletions and substitutions necessary to transform w into w'. It is one of the fundamental string metrics used in natural language processing [14], DNA analysis [7] or analysis of neural networks [26]. Furthermore, it is closely related to dynamic time warping, which is a basic measure of similarity between timed series [14]. As a similarity measure between languages, edit distance has been considered on regular languages [3] and context-free languages [8].

1.1 Contributions

This paper aims to provide a fresh perspective to the field of DFA minimization, which balances the conflicting objectives of state reduction and similarity of languages.

We introduce the problem of *minimization of DFA modulo the edit distance*, a novel approach that balances compactness with approximation flexibility. Unlike classical minimization, which focuses on preserving exact language equivalence, this approach allows for slight deviations between the original and minimized automata, measured by the edit distance, which enables significant state reduction while maintaining practical usability.

The problem is considered in two variants. In the fixed threshold variant (MIN-ED[k]), the goal is to find a DFA $\mathcal{B}$ with the minimal number of states such that the edit distance between the languages of the original DFA $\mathcal{A}$ and $\mathcal{B}$ is bounded by a given threshold $k \in \mathbb{N}$.

In the bounded threshold variant (MIN-BED), the objective is to find a minimal DFA $\mathcal{B}$ such that the edit distance between the languages of $\mathcal{A}$ and $\mathcal{B}$ is finite. This variant further loosens the constraints, allowing for more aggressive state reduction while still capturing the core structure of the original language.

These two perspectives provide a flexible framework for tailoring DFA minimization to the needs of specific applications, particularly in noisy or approximate environments.

Our main technical results are for the bounded threshold variant MIN-BED. We show that this problem is in general Σ_2^P and both NP-hard and coNP-hard, but we identify subclasses of DFA, which enjoy better complexity. We consider automata with a bounded depth of strongly connected components. We prove that for any c, restricting the problem to automata whose accepting runs visit at most c different (maximal) strongly connected components reduces the complexity to NP. For cases $c \in \{1, 2\}$, we provide polynomial-time algorithms and show NP-hardness for $c = 3$.

For the fixed threshold variant MIN-ED[k], we prove that the problem is robustly PSPACE-complete, in the sense that it is unclear how to restrict the class of DFA to gain better complexity results. In particular, we discuss that the hardness proof can be adapted to automata in which accepting runs visit at most one (maximal) strongly connected component.

1.2 Related work

Minimization of finite automata has been extensively studied [22, Chapter 10]. The foundational algorithms in this area were designed to reduce the state complexity of DFA while preserving their exact language equivalence. Notable among these are the algorithms proposed by Hopcroft [11], which works in $O(n \log n)$, the algorithm by Moore [20], which is quadratic in the worst case but enjoys better average-case complexity, and the algorithm by Brzozowski [5], which is exponential in the worst case, but often outperforms other algorithms in practice [2].

Building on these foundational methods, researchers have explored extensions and generalizations of DFA minimization. One significant direction is hyper-minimization, which allows for a relaxation of exact language equivalence by permitting finite differences between the languages of the original and minimized DFA. This approach has been studied extensively, with algorithms achieving $O(n \log n)$ complexity [10, 18]. Hyper-minimization has also been adapted to other automata models, such as deterministic tree automata [13] and deterministic weighted tree automata [19].

Language inclusion modulo the edit distance has been studied between regular languages represented by DFA and NFA [3], and regular and context-free languages represented by DFA, NFA, and DPDA [8].

Despite extensive research on DFA minimization, the problem of minimization modulo the edit distance has not been addressed in the literature. Our work fills this gap.

2 Preliminaries

A *word* w over a finite alphabet Σ of letters is a finite sequence of letters. The set of all finite words over Σ is denoted by Σ^*.

A *deterministic finite-state automaton (DFA)* is a tuple $\langle \Sigma, Q, q_0, F, \delta \rangle$ consisting of the alphabet Σ, a finite set of states Q, the initial state $q_0 \in Q$, a subset $F \subseteq Q$ of accepting states, and a transition function $\delta \colon Q \times \Sigma \to Q$. The size of an automaton $\mathcal{A}$, denoted by $|\mathcal{A}|$, is its number of states.

Given a word $w_1 \dots w_n$, the *run* of a DFA $\mathcal{A}$ on w is a sequence of states $q_0 q_1 \dots q_n$ such that q_0 is the initial state and for each i we have $\delta(q_{i-1}, w_i) = q_i$. The run on w is accepting if its last state belongs to F. The *language* of $\mathcal{A}$, denoted by $\mathcal{L}(\mathcal{A})$, is the set of words whose runs are accepting. DFA $\mathcal{A}$ and $\mathcal{B}$ are *equivalent* if and only if $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{B})$.

Let $\mathcal{A}$ be a DFA and q be its state. We denote by $\mathcal{A}_q$ the DFA obtained from $\mathcal{A}$ by changing the initial state to q. The DFA $\mathcal{A}$ is *minimal* if and only if (a) all states of $\mathcal{A}$ are reachable from the initial state, and (b) for all states q, q' of $\mathcal{A}$, if $q \neq q'$, then $\mathcal{L}(\mathcal{A}_q) \neq \mathcal{L}(\mathcal{A}_{q'})$. Due to Myhill-Nerode theorem [12], every regular language is recognized by a minimal DFA $\mathcal{A}$, which is unique (up to state names), and every DFA language equivalent to $\mathcal{A}$ other than $\mathcal{A}$ has more states than $\mathcal{A}$. Furthermore, given a DFA $\mathcal{A}$, we can compute a minimal DFA equivalent to $\mathcal{A}$ in time $O(|\mathcal{A}| \log |\mathcal{A}|)$.

For convenience, throughout the paper, we assume that input DFAs are already minimal – the minimization can be done in polynomial time if needed. This in particular means that all the states are reachable.

A state s of DFA is a *sink* if it is not accepting and all the transitions from s lead to s. In a minimal DFA there is at most one sink, which is the only state from which no accepting state is reachable.

We extend δ to all words by defining $\delta(q, \epsilon) = q$ (where ϵ is the empty string) and $\delta(q, wx) := \delta(\delta(q, w), x)$ for each state q, word w and letter x. A subset of states C of a DFA $\mathcal{A}$ is a *(maximal) strongly connected component* (SCC) if for any two states $s_1, s_2 \in C$, there is a non-empty word w such that $\delta(s_1, w) = s_2$, and there is no $C' \supset C$ with this property. We say that $\mathcal{A}$ has the SCC-depth c if c is the maximal number such that there is an accepting run visiting c different (maximal) SCC of $\mathcal{A}$. Importantly, we only consider accepting runs and hence the sink state does not influence the SCC-depth; for example, both automata presented in Figure 2 (page 8) have the SCC-depth 3.

For convenience, we use *regular expressions* defined in the usual manner, i.e., expressions defined using the BNF $e := \epsilon \mid x \mid ee \mid e + e \mid e^* \mid e^+$, where x is a letter and ϵ stands for the empty word. The *language* $\mathcal{L}(e)$ of e is defined inductively: $\mathcal{L}(\epsilon) = \{\epsilon\}$, for each letter x, $\mathcal{L}(x) = \{x\}$; $\mathcal{L}(e_1 e_2) = \{vw \mid v \in \mathcal{L}(e_1), w \in \mathcal{L}(e_2)\}$; $\mathcal{L}(e_1 + e_2) = \{v \mid v \in \mathcal{L}(e_1) \vee v \in \mathcal{L}(e_2)\}$; $\mathcal{L}(e^*) = \{v_1 \dots v_n \mid \forall i\, v_i \in \mathcal{L}(e)\}$, and $\mathcal{L}(e^+) = \mathcal{L}(ee^*)$.

2.1 Distance between languages and decision problems

The *edit distance* between words w, w', denoted by $\mathrm{ED}(w, w')$, is the minimal number of operations: insertions of a letter, deletions of a letter, and substitutions of a letter, necessary to transform w into w'. For example, $\mathrm{ED}(ab, acd) = 2$ (one substitution and one insertion).

For a natural number k, we say that a language $\mathcal{L}$ is *contained in a language $\mathcal{L}'$ up to the edit distance k*, denoted as $\mathcal{L} \subseteq^k_{\mathrm{ED}} \mathcal{L}'$, if and only if for each $w \in \mathcal{L}$ there is $w' \in \mathcal{L}'$ such that $\mathrm{ED}(w, w') \leq k$. We extend this notion to $\subseteq^\infty_{\mathrm{ED}}$ by stating that $\mathcal{L} \subseteq^\infty_{\mathrm{ED}} \mathcal{L}'$ if and only if for some $k \in \mathbb{N}$ we have $\mathcal{L} \subseteq^k_{\mathrm{ED}} \mathcal{L}'$. The relation $\subseteq^k_{\mathrm{ED}}$ is transitive only for $k = \infty$.

We say that DFA $\mathcal{A}, \mathcal{B}$ are *equivalent up to the edit distance* $k \in \mathbb{N} \cup \{\infty\}$, denoted as $\mathcal{A} \equiv^k_{\mathrm{ED}} \mathcal{B}$, if and only if $\mathcal{L}(\mathcal{A}) \subseteq^k_{\mathrm{ED}} \mathcal{L}(\mathcal{B})$ and $\mathcal{L}(\mathcal{B}) \subseteq^k_{\mathrm{ED}} \mathcal{L}(\mathcal{A})$. Notice that $\mathcal{A} \equiv^k_{\mathrm{ED}} \mathcal{B}$ is an equivalence relation if and only if $k = \infty$; in other cases, it is symmetric and reflexive, but not necessarily transitive.

We study decision problems called MIN-ED[k] for $k \in \mathbb{N}$ and MIN-BED defined as follows:

MIN-ED[k]: minimization modulo a fixed edit distance k
Input: A DFA $\mathcal{A}$ and a number m.
Output: Is there a DFA $\mathcal{B}$ with at most m states such that $\mathcal{A} \equiv^k_{\mathrm{ED}} \mathcal{B}$?

MIN-BED: minimization modulo a bounded edit distance
Input: A DFA $\mathcal{A}$ and a number m.
Output: Is there a DFA $\mathcal{B}$ with at most m states such that $\mathcal{A} \equiv^\infty_{\mathrm{ED}} \mathcal{B}$?

2.2 Containment modulo a bounded edit distance

We recall a condition for checking containment modulo a bounded edit distance presented in [3, Theorem 4.1]. This condition is used in Sections 3 and 4.

Following [3], we consider minimal DFA, which recognize infinite languages. Note that all DFA that recognize finite non-empty languages are equivalent modulo a bounded edit distance, while the DFA that recognizes the empty language is only equivalent to itself. Conversely, a DFA that recognizes an infinite language can be equivalent modulo a bounded edit distance only to a DFA recognizing infinite language. Therefore, considering only DFA that recognize infinite languages removes only trivial cases.

For a DFA $\mathcal{A}$ (minimal and recognizing an infinite language), let $SCC(\mathcal{A})$ be the set of all (maximal) strongly connected components of $\mathcal{A}$ containing an accepting state or from which an accepting state is reachable.

We define $\textsc{dag}(\mathcal{A})$ as a directed acyclic graph whose set of nodes is $SCC(\mathcal{A})$, and there is an edge from C_1 to C_2 if and only if $C_1 \neq C_2$ and there is a transition from some state of C_1 to some state in C_2. Let $\textsc{dag}^*(\mathcal{A})$ be the reflexive and transitive closure of $\textsc{dag}(\mathcal{A})$.

For $C \in SCC(\mathcal{A})$, we define $\mathcal{L}(\mathcal{A}|C)$ as the language of words such that $w \in \mathcal{L}(\mathcal{A}|C)$ if, starting from some state in C, the automaton can read the whole word w without leaving C. More precisely, it is the language of words $w_1 \dots w_n$ such that there are $q^0 \dots q^n \in C$ satisfying, for all i, $\delta(q^{i-1}, w_i) = q^i$.

Now, consider two DFA $\mathcal{A}$ and $\mathcal{B}$. We say that a path $\pi = C_1 \dots C_n$ in $\textsc{dag}(\mathcal{A})$ is *covered* by a path $\pi' = D_1 \dots D_m$ in $\textsc{dag}^*(\mathcal{B})$ if and only if $n = m$ and for every $1 \leq i \leq n$ we have $\mathcal{L}(\mathcal{A}|C_i) \subseteq \mathcal{L}(\mathcal{B}|D_i)$. Checking whether a path is covered by a path can be decided in polynomial time [3, Lemma 4.4].

It has been shown in [3, Theorem 4.1] that $\mathcal{L}(\mathcal{A}) \subseteq^{\infty}_{\text{ED}} \mathcal{L}(\mathcal{B})$ if and only if every path in $\textsc{dag}(\mathcal{A})$ is covered by some path in $\textsc{dag}^*(\mathcal{B})$ (the result from [3, Theorem 4.1] holds even for non-deterministic finite automata). Furthermore, due to [3, Theorem 4.1], we have $\mathcal{L}(\mathcal{A}) \subseteq^{\infty}_{\text{ED}} \mathcal{L}(\mathcal{B})$ if and only if $\mathcal{L}(\mathcal{A}) \subseteq^{k}_{\text{ED}} \mathcal{L}(\mathcal{B})$ for any $k \geq (|\mathcal{A}| + 1) \cdot (|\mathcal{B}| + 1)$.

▶ Remark 1. The upper bound on the edit distance threshold in MIN-BED is $(|\mathcal{A}| + 1) \cdot m$, i.e., an instance $(\mathcal{A}, m)$ of MIN-BED is positive if and only if it is a positive instance of MIN-ED$[(|\mathcal{A}| + 1) \cdot m]$.

Note that a DFA $\mathcal{A}$ may contain a rejecting sink state, which has no counterpart in $\textsc{dag}(\mathcal{A})$. Furthermore, some states of $\mathcal{A}$ may not belong to any SCC (for example, the initial state of $\mathcal{A}$ if it is not reachable from any other state). In particular, if $\textsc{dag}(\mathcal{A})$ has no root, i.e., the node from which all other nodes are reachable, then the initial state of $\mathcal{A}$ does not belong to any SCC. We will use this fact further on.

▶ **Fact 2.** *Let $\mathcal{A}$ be a DFA such that (1) its language is non-empty, (2) for some word w, its language does not contain any extension of w ($\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(w\Sigma^*) = \emptyset$), and (3) $\textsc{dag}(\mathcal{A})$ does not have a root, i.e., a node from which all other nodes are reachable. Then, $|\mathcal{A}|$ is at least the number of nodes of $\textsc{dag}(\mathcal{A})$ plus 2.*

2.3 Example

Consider a program that **r**eceives some data, **c**omputes something and **s**ends the response, after which it can start over or **t**erminate. Traces of such a program can be represented over the alphabet $\{r, c, s, t\}$ as depicted in Figure 1 (left).

Observe that **s** is a sink state; the sink state is present in all automata, but we do not draw it or discuss it for better clarity.

Assume that we want to learn the (target) automaton $\mathcal{A}^T$ depicted in Figure 1 (left) from noisy data, which contains as a positive example the word *rcsrcsrcs* instead of *rcsrcsrcst*. It is possible that the learning procedure produces the automaton that is depicted in Figure 1 (right).

Observe that the automaton $\mathcal{A}^L$ from Figure 1 (right) is minimal in the classical sense, and also minimal w.r.t. hyper-minimization [10, 18] as there is no language that differ from the language of $\mathcal{A}^L$ on some finite set of words whose automaton has less states.

Observe that $\mathcal{A}^T \equiv^{1}_{\text{ED}} \mathcal{A}^L$. Clearly $\mathcal{L}(\mathcal{A}^T) \subseteq^{1}_{\text{ED}} \mathcal{L}(\mathcal{A}^L)$ because $\mathcal{L}(\mathcal{A}^T) \subseteq \mathcal{L}(\mathcal{A}^L)$. To see $\mathcal{L}(\mathcal{A}^L) \subseteq^{1}_{\text{ED}} \mathcal{L}(\mathcal{A}^T)$, observe that any word accepted by $\mathcal{A}^L$ but not by $\mathcal{A}^T$ is of the form $(rcs)^{4k+3}$ for some k, and it can be modified to be accepted by $\mathcal{A}^T$ by appending t. In this case, the automaton $\mathcal{A}^L$ is the minimal automaton equivalent to $\mathcal{A}^T$ up to edit distance 1. Therefore, minimizing $\mathcal{A}^T$ allows to overcome the problem introduced by the noised data.

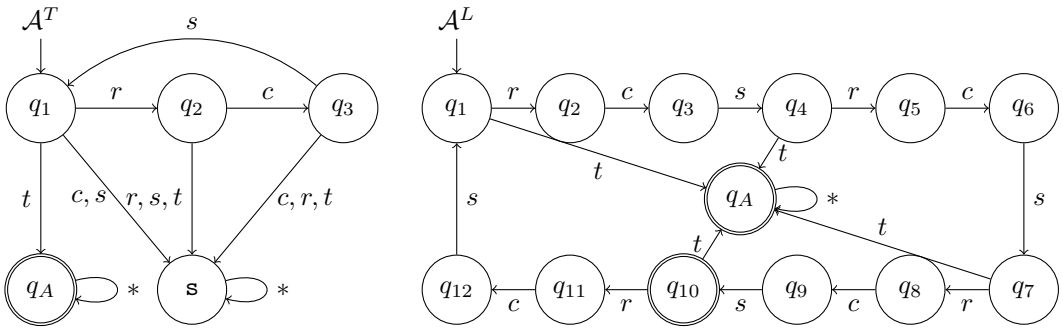

Figure 1 An example automaton $\mathcal{A}^T$ (left) and the automaton $\mathcal{A}^L$ (right) constructed from noised data. The remaining transitions of the right automaton lead to the sink state, which is also present in the automaton but not drawn here for better clarity.

Now consider an automaton $\mathcal{A}^K$ which is a copy of $\mathcal{A}^L$, but with the state q_2 accepting. In this case, this automaton accepts the word rc, which cannot be transformed into any word accepted by $\mathcal{A}^T$ with edit distance 1. Therefore, $\mathcal{A}^T \not\equiv^1_{\text{ED}} \mathcal{A}^K$. It can be checked, however, that $\mathcal{A}^T \equiv^2_{\text{ED}} \mathcal{A}^K$ and $\mathcal{A}^T \equiv^\infty_{\text{ED}} \mathcal{A}^K$.

3 Bounded edit distance

In this section, we study the minimization problem for DFA modulo a bounded edit distance. We show that the MIN-BED problem is in Σ^P_2 and it is both coNP-hard and NP-hard. We also identify a class of DFA for which the problem is NP-complete.

3.1 Membership in Σ^P_2 and coNP-hardness

We first show that the MIN-BED problem is in Σ^P_2. Recall that a language P is in Σ^P_2 if there is a language P' in coNP such that $x \in P$ if and only if there exists y of polynomial size in $|x|$ such that $(x, y) \in P'$.

Given two DFA $\mathcal{A}, \mathcal{B}$, checking whether $\mathcal{A} \equiv^\infty_{\text{ED}} \mathcal{B}$ amounts to checking two inclusions: $\mathcal{L}(\mathcal{A}) \subseteq^\infty_{\text{ED}} \mathcal{L}(\mathcal{B})$ and $\mathcal{L}(\mathcal{B}) \subseteq^\infty_{\text{ED}} \mathcal{L}(\mathcal{A})$. Since the inclusions can be checked in coNP [3, Theorem 5.2], checking equivalence can also be done in coNP.

To solve MIN-BED for an instance $(\mathcal{A}, m)$, assume that $m < |\mathcal{A}|$ (otherwise the answer is yes with $\mathcal{B} = \mathcal{A}$). Then, non-deterministically pick a DFA $\mathcal{B}$ of the size at most m and return whether $\mathcal{A} \equiv^\infty_{\text{ED}} \mathcal{B}$. It follows that the problem is in Σ^P_2.

▶ **Proposition 3.** *The MIN-BED problem is in* Σ^P_2.

Recall the condition for checking containment modulo a bounded edit distance (Section 2.2), which involves checking coverage of all paths in $\text{DAG}(\mathcal{A})$. As we strive for better complexity results, we will restrict the class of DFA to those with polynomially many paths in DAG. Later on, we consider DFA $\mathcal{A}$ such that $\text{DAG}(\mathcal{A})$ has bounded depth.

It has been shown in [3] that deciding language containment up to a bounded edit distance is coNP-complete. This does not directly imply that the MIN-BED problem or even deciding the equivalence up to a bounded edit distance is coNP-hard. For instance the language inclusion on languages represented by deterministic pushdown automata (DPDA) is undecidable [12], while the equivalence problem for DPDA is decidable [25]. Still, we show coNP-hardness of MIN-BED by adapting the reduction from [3].

▶ **Theorem 4.** *The MIN-BED problem is coNP-hard.*

3.2 MIN-ED[∞] over DFA of a bounded SCC-depth is NP-complete

We now restrict the class of DFA to those with polynomially many paths in DAG and show an improved complexity results.

▶ **Lemma 5.** *Fix $l \in \mathbb{N}$. Deciding $\mathcal{A} \equiv^{\infty}_{ED} \mathcal{B}$ over DFA $\mathcal{A}$ such that the number of paths of $\text{DAG}(\mathcal{A})$ is at most $|\mathcal{A}|^l$, is in NP.*

Notice that we do not require $\text{DAG}(\mathcal{B})$ to have polynomially many paths.

Proof. To check whether $\mathcal{A} \subseteq^{\infty}_{\text{ED}} \mathcal{B}$, for each path of $\text{DAG}(\mathcal{A})$, non-deterministically pick a path of $\text{DAG}^*(\mathcal{B})$ that covers it. Checking $\mathcal{B} \subseteq^{\infty}_{\text{ED}} \mathcal{A}$ is more complicated, as $\mathcal{B}$ may have exponentially many paths. Let Π be the set of all the paths of $\text{DAG}(\mathcal{A})$. We employ the following algorithm.

For each node C of $\text{DAG}(\mathcal{B})$, non-deterministically pick a non-empty subset $\mathit{covers}(C)$ of paths from Π such that for all $\pi \in \mathit{covers}(C)$, the last node D of π satisfies $\mathcal{L}(\mathcal{B}|C) \subseteq \mathcal{L}(\mathcal{A}|D)$. For nodes C, C' such that C' is a successor of C in $\text{DAG}(\mathcal{B})$ and $\pi \in \mathit{covers}(C)$, we say that π' is a *continuation* of π in C' if $\pi' \in \mathit{covers}(C')$ and π is a prefix of π' or $\pi = \pi'$. The algorithm accepts if for all nodes C, C' such that C' is a successor of C in $\text{DAG}(\mathcal{B})$, each $\pi \in \mathit{covers}(C)$ has a continuation in $\mathit{covers}(C')$.

The algorithm works in polynomial time, as the size of Π is polynomial. If the algorithm accepts, each path is covered. Indeed, consider a path $C_1 \dots C_n$ in $\text{DAG}(\mathcal{B})$. Consider π_1 in $\mathit{covers}(C_1)$, which exists as the subsets are non-empty. Then, inductively, let π_{i+1} be a continuation of π_i in C_i. From π_n in $\text{DAG}(\mathcal{A})$, we can extract a path in $\text{DAG}^*(\mathcal{A})$ that covers the path $C_1 \dots C_n$.

Now we prove that if every path is covered, then the algorithm accepts. To do so, construct $\mathit{covers}(\cdot)$ starting with $\mathit{covers}(C) = \emptyset$ for each node. Then, for each path $C_1 \dots C_n$ of $\text{DAG}(\mathcal{B})$, which is covered by a path $D_1 \dots D_n$ of $\text{DAG}^*(\mathcal{A})$, and for each i we construct a path π_i for $\mathit{covers}(C_i)$ as follows. First, remove repeating nodes from $D_1, \dots, D_i$, obtaining π'_i. Since π'_i is a path of $\text{DAG}^*(\mathcal{A})$, one can compute a path π_i of $\text{DAG}(\mathcal{A})$ such that π_i can be obtained from π'_i by possibly inserting some nodes. We add π_i to $\mathit{covers}(C_i)$. Then $\mathit{covers}(\cdot)$ is as required. ◀

▶ **Lemma 6.** *For every c, the MIN-BED problem over DFA of SCC-depth at most c is in NP.*

Proof. Given a DFA $\mathcal{A}$ and $m \in \mathbb{N}$, the algorithm first non-deterministically picks a DFA $\mathcal{B}$ with at most m states (if $m > |\mathcal{A}|$ then the algorithm accepts straight away). Notice that $\text{DAG}(\mathcal{A})$ has at most $|\mathcal{A}|^c$ paths as the length of a path is bounded by c. Then, the algorithm returns whether $\mathcal{A} \equiv^{\infty}_{\text{ED}} \mathcal{B}$ using the algorithm from Lemma 5. ◀

We now show NP-hardness of MIN-BED over DFA of SCC-depth 3. For a (directed) bipartite graph G, let $RB(G)$ be the minimal number of complete bipartite graphs contained in G (not necessarily disjoint), which together cover all edges of G. The following problem is NP-complete [21, Theorem 8.1].

R-bicontent covering
Input: a bipartite graph G without isolated nodes and $m \in \mathbb{N}$
Output: whether $RB(G) \leq m$

We employ the R-bicontent covering problem to prove the following hardness result.

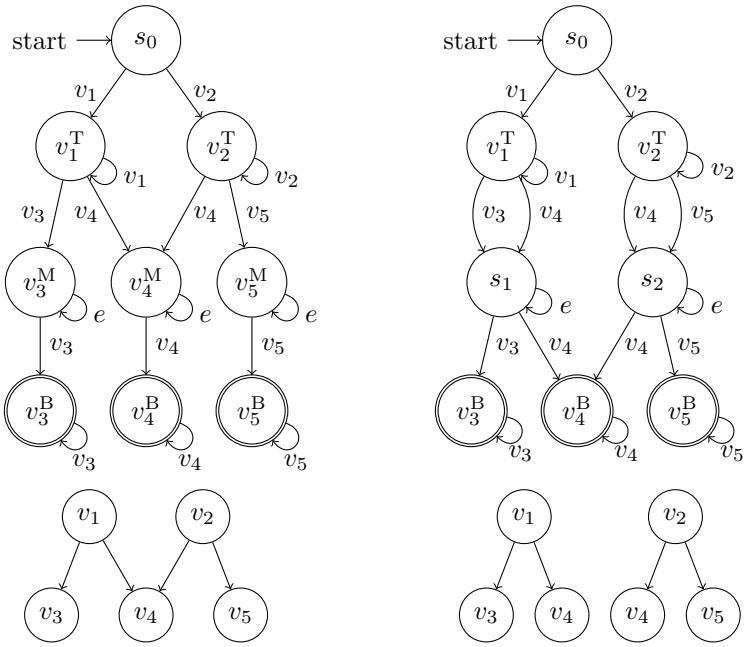

Figure 2 A bipartite graph G (bottom left), the corresponding automaton $\mathcal{A}_G$ (top left) and a minimal automaton equivalent modulo a bounded edit distance (top right) and the corresponding cover of G (bottom right). For readability, we omit sinks and the transitions leading to the sinks. Notice that the SCC-depth of both automata is 3, as the initial state is not in an SCC, and the sink does not count.

▶ **Lemma 7.** *The MIN-BED problem over DFA of SCC-depth at most* 3 *is NP-hard.*

Proof. Let $G = (V_{\mathrm{in}}, V_{\mathrm{out}}, E)$, where $E \subseteq V_{\mathrm{in}} \times V_{\mathrm{out}}$, be a bipartite graph without isolated nodes, and $m \in \mathbb{N}$. Consider a DFA $\mathcal{A}_G$ over the alphabet $V_{\mathrm{in}} \cup V_{\mathrm{out}} \cup \{e\}$ accepting exactly words of the form $v^+ue^*u^+$ such that $(v, u) \in E$.

The DFA $\mathcal{A}_G$ has the following states: an initial state s_0, a sink s_r, a state v^{T} for each node $v \in V_{\mathrm{in}}$, and two states v^{M}, v^{B} for each node $v \in V_{\mathrm{out}}$.

For each $(v, u) \in E$, $\mathcal{A}_G$ has the following transitions:

- From s_0 to v^{T} over v.
- From v^{T}, a loop over v and a transition to u^{M} over u.
- From u^{M}, a loop over e and a transition to u^{B} over u.
- A loop in u^{B} over u.

All the remaining transitions lead to the sink. The states v^{B} are accepting, while all other states are rejecting. Figure 2 contains an example of this reduction with a corresponding minimal automaton.

Assume that $l = RB(G)$. We claim that the minimal size of a DFA equivalent modulo a bounded edit distance to $\mathcal{A}_G$ is $|V_{\mathrm{in}}| + |V_{\mathrm{out}}| + l + 2$.

Let $S = \{(C^1_{\mathrm{in}}, C^1_{\mathrm{out}}, E^1), \ldots, (C^l_{\mathrm{in}}, C^l_{\mathrm{out}}, E^l)\}$ be a minimal R-bicontent covering of G. Each element of S is a complete bipartite subgraph of G. We construct a DFA $\mathcal{A}_S$ from $\mathcal{A}_G$ by removing all states v^{M} and introducing states $s_1, \ldots, s_l$. For each i, the state s_i has a loop over e and for all $u \in C^i_{\mathrm{out}}$, a transition over u to u^{B}. Moreover, for each $(v, u) \in E$, we pick one i such that $(v, u) \in E^i$ and we add a transition from v^{T} to s_i over u. The remaining transitions lead to the sink.

For $\mathcal{L}(\mathcal{A}_G) \subseteq^{\infty}_{\text{ED}} \mathcal{L}(\mathcal{A}_S)$ observe that $\mathcal{L}(\mathcal{A}_G) \subseteq \mathcal{L}(\mathcal{A}_S)$; each word $v^x u e^y u^z \in \mathcal{L}(\mathcal{A}_G)$ is accepted by $\mathcal{A}_S$ as there is a transition from v^{T} to some s_i such that $(v,u) \in E^i$, and from s_i there is a transition to u^{B}.

For $\mathcal{L}(\mathcal{A}_S) \subseteq^{\infty}_{\text{ED}} \mathcal{L}(\mathcal{A}_G)$, observe that $\mathcal{A}_S$ accepts words of the form $w = v^x u' e^y u^z$ such that $(v,u) \in E$. For each such a word, there is $w = v^x u e^y u^z$ accepted by $\mathcal{A}_G$ such that $\text{ED}(w,w') \leq 1$.

We now show that if for some l there is a DFA $\mathcal{B} \equiv^{\infty}_{\text{ED}} \mathcal{A}_S$ of the size $|V_{\text{in}}| + |V_{\text{out}}| + l + 2$, then $RB(G)$ is at most l. Due to Fact 2, $\text{DAG}(\mathcal{B})$ has at most $|V_{\text{in}}| + |V_{\text{out}}| + l$ nodes.

Consider $\text{DAG}(\mathcal{B})$. Since $\mathcal{B}$ is equivalent modulo a bounded edit distance to $\mathcal{A}_G$, every path in $\text{DAG}(\mathcal{B})$ is covered in $\text{DAG}^*(\mathcal{A}_G)$ and vice versa. Therefore, $\text{DAG}(\mathcal{B})$ contains three sets of nodes $V_1 = \{D \mid \mathcal{L}(\mathcal{B}|D) = \mathcal{L}(v^*) \text{ for some } v \in V_{\text{in}}\}$, $V_2 = \{D \mid \mathcal{L}(\mathcal{B}|D) = \mathcal{L}(e^*)\}$, and $V_3 = \{D \mid \mathcal{L}(\mathcal{B}|D) = \mathcal{L}(u^*) \text{ for some } u \in V_{\text{out}}\}$. Observe that $|V_1| \geq |V_{\text{in}}|$ and $|V_3| \geq |V_{\text{out}}|$, as otherwise some paths of $\text{DAG}(\mathcal{A}_G)$ would not be covered. Hence $|V_2| \leq l$.

We claim that each node in V_2 corresponds to a (possibly empty) complete bipartite subgraph of G. For $D \in V_2$, let $D_{\text{in}} \subseteq V_{\text{in}}$ be the labels of the predecessors of D from V_1, and $D_{\text{out}} \subseteq V_{\text{out}}$ be the labels of the successors of D from V_3. Observe that for any $v \in D_{\text{in}}$, $u \in D_{\text{out}}$ we have $(v,u) \in E$, because every path in $\text{DAG}(\mathcal{B})$ is covered in $\text{DAG}^*(\mathcal{A}_G)$, which in turn encodes edges of G, and hence the complete bipartite graph $(D_{\text{in}}, D_{\text{out}}, D_{\text{in}} \times D_{\text{out}})$ is a subgraph of G. Furthermore, since every path in $\text{DAG}(\mathcal{A}_G)$ is covered by $\text{DAG}^*(\mathcal{B})$, every edge of G is contained in some complete subgraph. Thus, $X = \{(D_{\text{in}}, D_{\text{out}}, D_{\text{in}} \times D_{\text{out}}) \mid D \in V_2\}$ is a set of complete bipartite graphs which together cover all edges of G. Finally, $RB(G) \leq |X| \leq |V_2| \leq l$. ◀

By combining the containment in NP from Lemma 6 and the hardness from Lemma 7, we have the following.

▶ **Theorem 8.** *For any $c \geq 3$, the MIN-BED problem restricted to DFA of SCC-depth at most c is NP-complete.*

4 Effective algorithms for bounded edit distance

In this section, we show that the MIN-BED problem restricted to DFA of SCC-depth at most 2 can be solved in polynomial time. In order to avoid technicalities, we allow the minimized automaton to operate over an extended alphabet with additional fresh letters. We use up to $|\mathcal{A}|$ fresh letters and the number of occurrences of each fresh letter is bounded among all accepted words.

We first investigate automata, for which the minimization problem is trivial. There are two reasons for that: either $\mathcal{L}(\mathcal{A})$ is finite or (conversely) the language is equivalent modulo bounded edit distance to Σ^*. In the former case, the DFA has only sink and possibly some transient states, i.e., the states that do not belong to any SCC and hence can be visited at most once along any run. In the latter case, the DFA has a state from which no sink is reachable. We argue that such automata are equivalent either to a single-state automaton accepting/rejecting all the words or to a two-state automaton recognizing the language $\{\epsilon\}$. All finite non-empty languages are equivalent modulo bounded edit distance, but $\{\epsilon\}$ has the DFA with the least number of states.

▶ **Proposition 9.** *Consider a DFA $\mathcal{A}$. (a) If $\mathcal{L}(\mathcal{A})$ is finite, then either $\mathcal{L}(\mathcal{A}) = \emptyset$ or $\mathcal{L}(\mathcal{A}) \equiv^{\infty}_{ED} \{\epsilon\}$. (b) If $\mathcal{A}$ has a state from which no sink is reachable, then $\mathcal{L}(\mathcal{A}) \equiv^{\infty}_{ED} \Sigma^*$.*

To see (a), observe that the empty language is equivalent modulo bounded edit distance only to the empty language, and all finite non-empty languages are equivalent.

To see (b), observe that if there is a state from which no sink is reachable, then there is a state q in some bottom SCC C of $\mathcal{A}$ from which no sink is reachable. Observe that since $\mathcal{A}$ is minimal, C has an accepting state q_f – otherwise C would be reduced to a sink. Let w_0 be a shortest word such that $\mathcal{A}$ reading w_0 from q_0 ends in q. For each word $w \in \Sigma^*$, we now consider the word w_0w. Let q' be the state of $\mathcal{A}$ after reading w_0w, and let w_f be a shortest word such that the state of $\mathcal{A}$ after reading w_f from q' is q_f.

Clearly, $\mathcal{A}$ accepts w_0ww_f. Moreover, the sizes of w_0 and w_f can be bounded by the number of states of $\mathcal{A}$ (using a pumping argument). This shows that $\Sigma^* \subseteq^{2|\mathcal{A}|}_{\text{ED}} \mathcal{L}(\mathcal{A})$. Obviously $\mathcal{L}(\mathcal{A}) \subseteq^{0}_{\text{ED}} \Sigma^*$, and thus $\mathcal{A}$ is equivalent modulo a bounded edit distance to a single state DFA, which accepts every word.

Due to Proposition 9, in the remainder of this section we assume that a given DFA $\mathcal{A}$ recognizes an infinite language (which can be easily checked) and from every state of $\mathcal{A}$ the rejecting sink state is reachable. We call such DFA *non-trivial*. Observe that for a non-trivial DFA $\mathcal{A}$, the DAG $SCC(\mathcal{A})$ is non-empty.

4.1 Single SCC with a sink

We now focus on the case of DFA $\mathcal{A}$ such that $\mathcal{A}$ is non-trivial and $SCC(\mathcal{A})$ is a singleton $\{C\}$. This means that $\mathcal{A}$ may contain states that are not in any SCC and the sink $\{s_r\}$.

▶ **Theorem 10.** *For any non-trivial DFA $\mathcal{A}$ with $\text{DAG}(\mathcal{A})$ consisting of a single node, $\text{MIN-SCC}(\mathcal{A})$ returns an automaton equivalent to $\mathcal{A}$ modulo a bounded edit distance such that any DFA equivalent to $\mathcal{A}$ modulo a bounded edit distance has at least as many states as $\text{MIN-SCC}(\mathcal{A})$.*

Proof sketch. The automaton $\mathcal{A}$ can be minimized using the following steps: (1) remove all that states except for C and s_r, (2) make all states of C accepting, (3) pick any state of C as initial, and (4) minimize the resulting automaton using standard DFA minimization.

The automaton $\mathcal{A}'$ obtained this way is equivalent modulo a bounded edit distance to $\mathcal{A}$, and by construction $\mathcal{A}'$ and $\text{MIN-SCC}(\mathcal{A})$ are language equivalent.

We argue that $\text{MIN-SCC}(\mathcal{A})$ is minimal. Let $\mathcal{B} = \text{MIN-SCC}(\mathcal{A})$ and assume that $\mathcal{C}$ is a minimal automaton such that $\mathcal{C} \equiv^{\infty}_{\text{ED}} \mathcal{A}$. Recall that $\mathcal{A}_q$ stands for $\mathcal{A}$ with the initial state changed to q. One can show that there is a state s such that $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\mathcal{C}_s)$. Since $\mathcal{B}$ is by construction a minimal automaton recognizing $\mathcal{L}(\mathcal{B}_{q_0})$, we have $|\mathcal{B}| \leq |\mathcal{C}|$, and since $\mathcal{C}$ is minimal, $\mathcal{B} = \text{MIN-SCC}(\mathcal{A})$ is minimal as well. ◀

A heuristic for the general case

The described algorithm gives us a polynomial-time procedure to minimize SCCs in non-trivial DFA with multiple SCCs. The idea is that one can minimize each SCC independently and then connect the results according to the connections in $\text{DAG}(\mathcal{A})$, as described below.

Consider $\mathcal{A}$ whose $\text{DAG}(\mathcal{A})$ consists of nodes $C_1, \dots C_n$. For each C_i, consider a DFA $\mathcal{A}_i$ which is a restriction of $\mathcal{A}$ to the states C_i and the sink, where the initial state is any state of C_i and each transition from state of C_i that in $\mathcal{A}$ does not lead to a state in C_i is diverted to the sink. Moreover, let $\mathcal{A}_0$ be a two-state automaton where all the transitions from the accepting initial state lead to the sink. We now consider the DFA $\mathcal{A}^S$ obtained by taking the disjoint union of automata $\mathcal{A}_0, \mathcal{A}_1, \dots, \mathcal{A}_n$ and unifying the sinks. Consider fresh letters $c_1, \dots, c_n$. Now for each C_i that has no predecessor in $\text{DAG}(\mathcal{A})$, we add a transition from all the states of $\mathcal{A}_0$ (except the sink) to all the states of $\mathcal{A}_i$ (except the sink). Moreover, for each edge (C_i, C_j) of $\text{DAG}(\mathcal{A})$, we add a transition from all the states of $\mathcal{A}_i$ (except the sink) to all the states of $\mathcal{A}_j$ (except the sink). The remaining edges are to the sink.

The procedure described above constructs in polynomial time a DFA $\mathcal{A}^S$ that is equivalent modulo a bounded edit distance to the initial one.

▶ **Corollary 11.** *Let $\mathcal{A}$ be a non-trivial DFA. Then, the DFA $\mathcal{A}^S$ is equivalent modulo a bounded edit distance to $\mathcal{A}$ and can be constructed in polynomial time.*

This gives us a possible heuristic to reduce the size of a given automaton. The procedure described above does not guarantee that the constructed automaton is minimal – for example, it may be possible to eliminate some SCCs. This comes as no surprise as that the minimization problem is both NP-hard and coNP-hard, and the above algorithm works in polynomial time. Nevertheless, this procedure is optimal whenever the SCC structure of the input automaton matches that of any minimal automaton; in particular, this holds if each SCC recognizes a distinct language.

4.2 DFA of SCC-depth 2

For DFA of SCC-depth 3, the MIN-BED problem is NP-hard. We present a complementary result that for DFA of SCC-depth at most 2, minimization modulo a bounded edit distance can be computed in polynomial time.

Recall the example in Figure 2: the main idea there is to have many SCCs in the "middle layer" of the automaton – SCCs that have predecessors and successors in the corresponding DAG. Because of that, the states (being single-state maximal SCCs) s_1 and s_2, that have the same language ($\mathcal{L}(e^*)$), cannot be merged. However, for SCCs with no successors, it holds that they can be merged if they have the same language – and the same holds for SCCs with no predecessors. We build on this idea to solve the minimization modulo a bounded edit distance for DFA of SCC-depth 2, where each SCC has either no successors or no predecessors. One needs to be careful though; even if the input automaton has the SCC depth 2, it might happen that the minimal equivalent one has SCC depth 3, as we will show later on (Example 13).

For a path π, by $\pi[i]$ we denote the ith element of π (starting from 1). We say that a path π is *weakly covered* by π' if there is a sequence $i_1 \leq \cdots \leq i_k$ such that $\pi'[i_1] \dots \pi'[i_k]$ covers π. For example, the path CCC is weakly covered by CC' because of the sequence $1, 1, 1$.

Let $\mathcal{A}$ be a DFA of SCC-depth at most 2. A subset E of paths of $\text{DAG}(\mathcal{A})$ is:

- *noncollapsible* if for each path C, D in $\text{DAG}(\mathcal{A})$ of length 2 we have $\mathcal{L}(\mathcal{A}|C) \not\subseteq \mathcal{L}(\mathcal{A}|D)$ and $\mathcal{L}(\mathcal{A}|C) \not\supseteq \mathcal{L}(\mathcal{A}|D)$.
- *irredundant* if there is no proper subset E' of E such that every path in E is weakly covered by a path in E'.
- a *minimal cover* if it is noncollapsible, irredundant, and paths from E weakly cover all paths in $\text{DAG}(\mathcal{A})$.

Let $\mathcal{A}, \mathcal{B}$ be DFA. A path $C_1, \dots, C_k$ is *tightly covered* by a path $D_1, \dots, D_k$ if for every i we have $\mathcal{L}(\mathcal{A}|C_i) = \mathcal{L}(\mathcal{B}|D_i)$.

We show that if $\mathcal{A}, \mathcal{B}$ are equivalent modulo a bounded edit distance, then every path in a minimal cover of paths in $\text{DAG}(\mathcal{A})$ is tightly covered by some path from $\text{DAG}^*(\mathcal{B})$. The intuition is that each paths in the minimal cover defines a "maximal" possible language, that has to be covered by some path of $\text{DAG}^*(\mathcal{B})$, which in turn has to be covered by some path of $\text{DAG}^*(\mathcal{A})$. The language maximality then guarantees the equality of the languages.

Algorithm 1 Function MIN-DEPTH-2($\mathcal{A}$) computing a minimal equivalent automaton in the 2 SCC-depth case. All transitions that are not explicitly defined lead to the sink.

Input: A DFA $\mathcal{A} = \langle \Sigma, Q, q_0, F, \delta \rangle$.
$P \leftarrow$ a minimal cover of paths in DAG($\mathcal{A}$)
$L_1 \leftarrow \{C \mid (C) \in P \vee \exists C'.(C, C') \in P\}$
$L_2 \leftarrow \{C \mid \exists C'.(C', C) \in P\}$
$L'_i \leftarrow$ a minimal subset of L_i s.t. for every $C \in L_i$ there is $C' \in L_i$ with $\mathcal{L}(\mathcal{A}|C) = \mathcal{L}(\mathcal{A}|C')$, for $i \in \{1, 2\}$
$\tilde{L}_i \leftarrow \{\text{MIN-SCC}(\mathcal{A}|C) \mid C \in L'_i\}$ for $i \in \{1, 2\}$
$\tilde{\mathcal{A}} \leftarrow$ the disjoin union of DFA $\tilde{L}_1 \cup \tilde{L}_2$
for every $C \in L'_1$, $D \in L'_2$ such that the path (C, D) is tightly covered by some path from P **do**
    connect all states of MIN-SCC($\mathcal{A}|C$) to all states of MIN-SCC($\mathcal{A}|D$)
add a sink to $\tilde{\mathcal{A}}$, set all states but sink as accepting
for each state of $\tilde{L}_1$ **do**
    $\tilde{\mathcal{A}}_q \leftarrow \tilde{\mathcal{A}}$ with q as the initial state
    add transitions from q to all the states of $\tilde{L}_1$ that are in a different SCC than q
    if $\mathcal{A} \equiv^{\infty}_{\text{ED}} \tilde{\mathcal{A}}_q$ **then** $\tilde{\mathcal{A}} \leftarrow \tilde{\mathcal{A}}_q$
if $\tilde{\mathcal{A}}$ still has no initial state **then**
    $q'_0 \leftarrow$ a fresh state (accepting)
    add q'_0 to $\tilde{\mathcal{A}}$
    add transitions from q'_0 to all states of $\tilde{L}_1$
return $\tilde{\mathcal{A}}$

▶ **Lemma 12.** *Let DFA $\mathcal{A}, \mathcal{B}$ be equivalent modulo a bounded edit distance, SCC depth of $\mathcal{A}$ be 2, and P be a minimal cover of paths in* DAG($\mathcal{A}$)*. Then, every path in P is tightly covered by some path from* DAG*($\mathcal{B}$)*.*

Now, we construct a DFA equivalent modulo a bounded edit distance to $\mathcal{A}$ with the minimal number of states. To do so, we first compute a minimal cover of paths in DAG($\mathcal{A}$). This is done iteratively in polynomial time, by starting from all the noncollapsible paths of DAG($\mathcal{A}$), and removing paths that can be covered by other paths. Then, we split the SCCs of the minimal cover into two categories: *starters*, which are the first SCCs of the paths (including paths of length 1), and *finishers*, which are the second SCCs of the paths of lengths 2. We remove duplicates in each category, minimize each SCC, and construct the target automaton $\tilde{\mathcal{A}}$ that consists of a disjoint union of all the remaining starters and finishers.

For each path of length 2 in the minimal cover, we find a corresponding starter and a finisher, and connect all the states of the starter to all the states of the finisher (connecting any state from the starter to any state of the finisher would also work). This may require extending the alphabet with additional letters; we need at most $|P|$ such letters.

If it is possible to pick a state of some starter as an initial state, then we do; otherwise we add one additional starting state. We connect the initial state to all the starters.

▶ **Example 13.** Consider the automaton $\mathcal{A}$ depicted in Figure 3. Without the red dashed edge, this automaton is minimal, and a separate initial state q_0 is needed. However, if we consider $\mathcal{A}$ with with the red dashed edge, then it is not minimal – one can remove q_0, make q_1 initial and connect q_1 to q_3 over c, obtaining a smaller equivalent automaton. Interestingly, in this case the SCC-depth of the minimal automaton is 3.

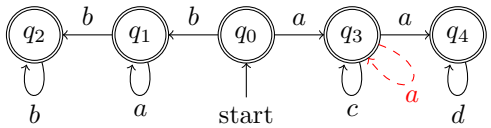

Figure 3 Example of the initial state selection process. For readability, the sink and the transitions that lead to sinks are omitted.

We add transitions from the initial state to all the starters (possibly using another letter). The detailed algorithm is presented in Algorithm 1. A quick check shows that it works in polynomial time.

The DFA MIN-DEPTH-2($\mathcal{A}$) is constructed in such a way that DAG(MIN-DEPTH-2($\mathcal{A}$)) consists of nodes from $L_1 \cup L_2$ with edges from C to D if and only if $C \in L'_1, D \in L'_2$ and the path C, D is tightly covered by some path from P.

The automaton MIN-DEPTH-2($\mathcal{A}$) is not unique as the transitions between different SCCs may use different letters; the choice of letters is irrelevant as it can be fixed with a bounded edit distance.

▶ **Theorem 14.** *Given a non-trivial DFA $\mathcal{A}$ of SCC-depth at most 2, we can construct in polynomial time a DFA MIN-DEPTH-2($\mathcal{A}$) equivalent to $\mathcal{A}$ modulo bounded edit distance such that any DFA equivalent to $\mathcal{A}$ modulo a bounded edit distance has at least as many states as MIN-DEPTH-2($\mathcal{A}$).*

Proof. Observe that DAG(MIN-DEPTH-2($\mathcal{A}$)) consists of nodes from $L'_1 \cup L'_2$ with edges from C to D if and only if $C \in L'_1, D \in L'_2$ and the path C, D is tightly covered by some path from P. Hence each path in DAG(MIN-DEPTH-2($\mathcal{A}$)) is tightly covered by some path in P and hence it is covered by some path in DAG$^*(\mathcal{A})$. Conversely, every path in P is exactly covered by a path in DAG($\mathcal{B}$) and hence every path in DAG($\mathcal{A}$) is covered by a path in DAG*(MIN-DEPTH-2($\mathcal{A}$)). It follows that $\mathcal{A}$ and MIN-DEPTH-2($\mathcal{A}$) are equivalent modulo a bounded edit distance.

Minimality of MIN-DEPTH-2($\mathcal{A}$) follows from Lemma 12. Indeed, if $\mathcal{B}$ is a DFA, which is equivalent modulo a bounded edit distance to $\mathcal{A}$, then Lemma 12 implies that for each SCC C from $L'_1 \cup L'_2$, the DFA $\mathcal{B}$ contains an SCC D with $\mathcal{L}(\mathcal{A}|C) = \mathcal{L}(\mathcal{B}|D)$. Therefore, all SCCs present in MIN-DEPTH-2($\mathcal{A}$) need to have their counterparts in $\mathcal{B}$ with the same language. It follows that $\mathcal{B}$ has at least as many states as MIN-DEPTH-2($\mathcal{A}$). ◀

5 Fixed threshold edit distance

In this section we show that for any fixed k the MIN-ED[k] problem is PSPACE-complete. The hardness result is robust in the sense that the subclasses of DFA discussed earlier do not admit a better complexity of the MIN-ED[k] problem, and it is not clear how to restrict the class of DFA to obtain better complexity results.

For membership, one can non-deterministically pick a target automaton and verify its equivalence modulo the edit distance k in polynomial space using the algorithm presented in [3, Theorem 5.12]. For PSPACE-hardness we reduce the following problem.

Automata universality
Input: DFA $\mathcal{A}_1, \ldots, \mathcal{A}_n$ over an alphabet Σ, such that $\mathcal{A}_1$ contains one sink state and it accepts all words of length at most 1, and for all i we have that $\mathcal{L}(\mathcal{A}_i) \neq \Sigma^*$.
Output: whether $L(\mathcal{A}_1) \cup \cdots \cup L(\mathcal{A}_n) = \Sigma^*$

This problem is PSPACE-complete, which follows from the classic result [16] by adding an appropriate automaton $\mathcal{A}_1$.

▶ **Theorem 15.** *For each k, the MIN-ED[k] problem is PSPACE-hard.*

Proof. Fix $k \geq 1$. We reduce the automata universality problem. Let $\Sigma' = \{b_1, \ldots, b_n, \triangleright\}$ be disjoint from Σ. We construct a DFA $\mathcal{A}$ over the alphabet $\Sigma \cup \Sigma'$ that accepts exactly words of the form $(b_i)^k \triangleright w$ such that $w \in \mathcal{L}(\mathcal{A}_i)$.

The automaton $\mathcal{A}$ over the alphabet $\Sigma \cup \Sigma'$ consists of the disjoint union of the automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$ and fresh non-accepting states q_0 (initial) and q_i^j for $i, j \in \{1, \ldots, k\}$.

The automaton $\mathcal{A}$ has the following additional transitions:

1. $\delta(q_0, b_i) = q_i^1$ for any i;
2. $\delta(q_i^j, b_i) = q_i^{j+1}$ for any $j < k$;
3. $\delta(q_i^k, \triangleright)$ is the initial state of $\mathcal{A}_i$, for any j.

The remaining transitions, including transitions over Σ' from the states of $\mathcal{A}_1, \ldots, \mathcal{A}_n$, are to the sink state of $\mathcal{A}_1$. Figure 4 contains an example of such an automaton.

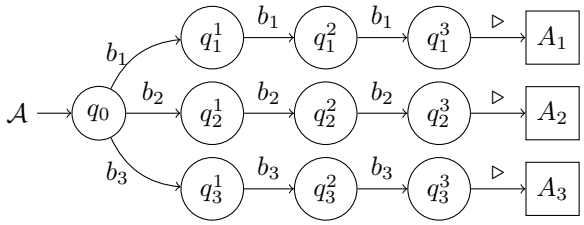

Figure 4 Example of the reduction in Theorem 15 for $k = 3$. For readability, sinks and the transitions that lead to sinks are omitted.

We conclude the reduction by fixing $m = 3$, i.e., we ask whether the input DFA is equivalent modulo the edit distance k to some automaton with at most 3 states.

Let $\mathcal{B}$ be the minimal automaton over $\Sigma \cup \Sigma'$ recognizing the language $\{\triangleright w \mid w \in \Sigma^*\}$. It has three states: q_0 (initial), q (accepting) and s (sink). From q_0 the transition over $\triangleright$ is to q, for each $x \in \Sigma$ the transition from q over x is to q, and all the remaining transitions are to s.

▶ **Lemma 16.** *For all three-state automata $\mathcal{C}$ such that $\mathcal{A} \equiv_{ED}^k \mathcal{C}$ we have $\mathcal{C} = \mathcal{B}$ (up to state renaming).*

▶ **Lemma 17.** *$\mathcal{A} \equiv_{ED}^k \mathcal{B}$ if and only if $L(\mathcal{A}_1) \cup \cdots \cup L(\mathcal{A}_n) = \Sigma^*$.*

Proof. Assume $\mathcal{A} \equiv_{\mathrm{ED}}^k \mathcal{B}$ and let $w \in \Sigma^*$. Since $\mathcal{B}$ accepts $\triangleright w$, for some i, $\mathcal{A}$ accepts $b_i^k \triangleright w$, which means that $\mathcal{A}_i$ accepts w. Thus $L(\mathcal{A}_1) \cup \cdots \cup L(\mathcal{A}_n) = \Sigma^*$.

Now assume $L(\mathcal{A}_1) \cup \cdots \cup L(\mathcal{A}_n) = \Sigma^*$. Each word accepted by $\mathcal{A}$ is of the form $b_i^k \triangleright w$, where $w \in \Sigma^*$. Then, $\mathcal{B}$ accepts $\triangleright w$ and $\mathrm{ED}(b_i^k \triangleright w, \triangleright w) \leq k$.

On the other hand, each word accepted by $\mathcal{B}$ is of the form $\triangleright w$, where $w \in \Sigma^*$. Since $L(\mathcal{A}_1) \cup \cdots \cup L(\mathcal{A}_n) = \Sigma^*$, there is i such that $w \in L(\mathcal{A}_i)$. Therefore, for $w' = b_i^k \triangleright w$ we have that $\mathcal{A}$ accepts w' and $\mathrm{ED}(w, w') \leq k$. Thus $\mathcal{A} \equiv_{\mathrm{ED}}^k \mathcal{B}$. ◀

Lemmas 16 and 17 together imply that the instance $(\mathcal{A}, 3)$ of the MIN-ED[k] problem is positive iff we have $L(\mathcal{A}_1) \cup \cdots \cup L(\mathcal{A}_n) = \Sigma^*$. This concludes the proof of PSPACE-hardness. ◀

▶ Remark 18. One can assume that the input automata of the automata universality problem consists of one SCC, for example, by adding an additional reset letter to each automaton (i.e., a letter such that the transition from each state over this letter leads to the initial state). This shows that the problem is PSPACE-hard even for automata of SCC-depth 1.

6 Other metrics

The paper focuses on minimizing DFA modulo similarity of languages, where the similarity measure is based on the edit distance between words. We briefly discuss alternative approaches to define similarity of languages: the Hamming distance and asymmetric relations.

6.1 Minimization modulo the Hamming distance

Hamming distance calculates the minimum number of single-character substitutions required to transform one word into another, assuming both words are of the same length. In contrast, edit distance also allows insertions and deletions of characters, making it applicable to words of different lengths. Hamming distance could be more advantageous in scenarios where fixed-length words are present, or situations where insertions and deletions are unrealistic, like comparing fixed-length DNA sequences. However, its limitation to equal-length words makes it less suitable than edit distance in cases where inputs may vary in length or include additional elements.

We now briefly discuss how our results could be adapted for the Hamming distance metrics. For fixed $k \in \mathbb{N}$, the PSPACE-hardness result from Theorem 15 can be easily adapted to the Hamming distance. For the bounded Hamming distance case, i.e, $k = \infty$, the result would be different from those obtained for equivalence modulo a bounded edit distance. First, our results for a bounded edit distance rely on the condition from [3, Theorem 4.1] presented in Section 2.2 for DFA to be equivalent modulo a bounded edit distance. Up to our best knowledge, there is no counterpart of this condition for the Hamming distance. Second, due to Proposition 9, we have focused on non-universal DFA, for which a rejecting sink state is reachable from every state, as all universal DFA are equivalent modulo finite edit distance to a single-state DFA. Proposition 9 does not extend to language equivalence modulo a bounded Hamming distance. For example, the language consisting of words of even length is equivalent modulo a bounded edit distance to the language of all words, whereas it is not equivalent modulo a bounded Hamming distance. For these reasons, we leave the study of minimization modulo a bounded Hamming distance as future work.

6.2 Asymmetric similarity relations

While we have discussed symmetric similarity measure, in some applications only one-sided errors are allowed. For instance, in formal verification, abstraction methods reduce a given model size by constructing an over-approximation, which is verified against a specification. In such a case, there are no false positives, i.e., the answers that the specification is met are correct, while there may be false negatives due to over-approximation. Over-approximation should balance the size of the minimized model and the false-negatives rate. We can address it with similarity relations.

Consider the following relation: $\mathcal{L} \preceq^k_{\mathrm{ED}} \mathcal{L}'$ if $\mathcal{L} \subseteq \mathcal{L}'$ and $\mathcal{L}' \subseteq^k_{\mathrm{ED}} \mathcal{L}$. For $k = \infty$, this is a transitive relation. This leads to the following minimization problem: given a DFA $\mathcal{A}$, find a minimal $\mathcal{B}$ such that $\mathcal{L}(\mathcal{A}) \preceq^\infty_{\mathrm{ED}} \mathcal{L}(\mathcal{B})$. This approach gives us a close over-approximation; an automaton that accepts all the words that $\mathcal{A}$ accepts and perhaps some additional words, but only if the additional words are close (w.r.t. the edit distance) to some accepted words.

Again, for fixed $k \in \mathbb{N}$, the PSPACE-hardness result from Theorem 15 can be easily adapted to the asymmetric distance $\preceq^{\infty}_{\mathrm{ED}}$. For $k = \infty$, most results can be easily adapted to the asymmetric distance $\preceq^{\infty}_{\mathrm{ED}}$: containment in Σ^P_2 (Proposition 3), coNP-hardness (Theorem 4), NP-completeness (Theorem 8), triviality of minimization with unrachable sink state (Proposition 9), and minimization of a DFA being a single SCC with a sink (Theorem 10). However, minimization of DFA of SCC-depth 2 does not extend easily to the asymmetric distance $\preceq^{\infty}_{\mathrm{ED}}$.

While potentially useful in applications like formal verification, these asymmetric relations exhibit challenges such as counter-intuitive behaviour, and require further investigation into their properties and practical applications.

7 Conclusions

We investigated the problem of minimizing DFA modulo the edit distance. We proved that this problem is PSPACE-complete for fixed edit distance and in Σ^P_2 for a bounded edit distance. Then, we proposed a restriction of the latter problem which improves the complexity: we showed that for a fixed SCC-depth, the minimization problem is in NP, and for SCC-depth at most 2 the problem can be solved in polynomial time.

An interesting future work are possible heuristics and approximations for the fixed-edit-distance case. In particular, if the edit distance is fixed, but it is not strictly enforced (for example, it can be slightly violated by some fraction of words).

The bounded edit distance minimization is at least as good as hyperminimization in terms of the number of states (except in some corner cases of finite languages). In practice, we observed that it sometimes minimize too much (see, e.g., Proposition 9). To mitigate this, and also preserve the modular structure of the automata, one could restrict the minimization algorithm to work at some subsets of automata states. In our future work we want to explore algorithms for this scenario.

References

1 Diego Aineto, Sergio Jimenez, and Eva Onaindia. Generalized Temporal Inference via Planning. In *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning*, pages 22–31, November 2021. doi:10.24963/kr.2021/3.

2 Marco Almeida, Nelma Moreira, and Rogério Reis. On the performance of automata minimization algorithms. In *Proceedings of the 4th Conference on Computation in Europe: Logic and Theory of Algorithms*, pages 3–14, 2007.

3 Michael Benedikt, Gabriele Puppis, and Cristian Riveros. Bounded repairability of word languages. *J. Comput. Syst. Sci.*, 79(8):1302–1321, 2013. doi:10.1016/j.jcss.2013.06.001.

4 Anat Bremler-Barr, David Hay, and Yaron Koral. Compactdfa: Scalable pattern matching using longest prefix match solutions. *IEEE/Acm Transactions On Networking*, 22(2):415–428, 2013. doi:10.1109/TNET.2013.2253119.

5 Janusz A Brzozowski. Canonical regular expressions and minimal state graphs for definite events. In *Proc. Symposium of Mathematical Theory of Automata*, pages 529–561, 1962.

6 David Carmel and Shaul Markovitch. Opponent modeling in multi-agent systems. In *International Joint Conference on Artificial Intelligence*, pages 40–52. Springer, 1995. doi:10.1007/3-540-60923-7_18.

7 William I. Chang and Eugene L. Lawler. Sublinear approximate string matching and biological applications. *Algorithmica*, 12(4/5):327–344, 1994. doi:10.1007/BF01185431.

8 Krishnendu Chatterjee, Thomas A. Henzinger, Rasmus Ibsen-Jensen, and Jan Otop. Edit distance for pushdown automata. *Log. Methods Comput. Sci.*, 13(3), 2017. doi:10.23638/LMCS-13(3:23)2017.

9 Massimiliano de Leoni, Paolo Felli, and Marco Montali. Strategy Synthesis for Data-Aware Dynamic Systems with Multiple Actors. In *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning*, pages 315–325, September 2020. doi:10.24963/kr.2020/32.

10 Pawel Gawrychowski and Artur Jeż. Hyper-minimisation made efficient. In *MFCS 2009*, pages 356–368, 2009. doi:10.1007/978-3-642-03816-7_31.

11 John Hopcroft. An n log n algorithm for minimizing states in a finite automaton. In *Theory of machines and computations*, pages 189–196. Elsevier, 1971.

12 John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation, Second Edition*. Addison-Wesley, 2000.

13 Artur Jeż and Andreas Maletti. Hyper-minimization for deterministic tree automata. In *CIAA 2012*, pages 217–228, 2012. doi:10.1007/978-3-642-31606-7_19.

14 Dan Jurafsky and James H. Martin. *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition, 2nd Edition*. Prentice Hall series in artificial intelligence. Prentice Hall, Pearson Education International, 2009. URL: https://www.worldcat.org/oclc/315913020.

15 Igor Khmelnitsky, Daniel Neider, Rajarshi Roy, Xuan Xie, Benoît Barbot, Benedikt Bollig, Alain Finkel, Serge Haddad, Martin Leucker, and Lina Ye. Analysis of recurrent neural networks via property-directed verification of surrogate models. *International Journal on Software Tools for Technology Transfer*, 25(3):341–354, 2023. doi:10.1007/s10009-022-00684-w.

16 Dexter Kozen. Lower bounds for natural proof systems. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pages 254–266. IEEE, 1977. doi:10.1109/SFCS.1977.16.

17 Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics doklady*, 10(8):707–710, 1966.

18 Andreas Maletti and Daniel Quernheim. Optimal hyper-minimization. *Int. J. Found. Comput. Sci.*, 22(8):1877–1891, 2011. doi:10.1142/S0129054111009094.

19 Andreas Maletti and Daniel Quernheim. Hyper-minimization for deterministic weighted tree automata. In *AFL 2014*, pages 314–326, 2014. doi:10.4204/EPTCS.151.22.

20 Edward F Moore et al. Gedanken-experiments on sequential machines. *Automata studies*, 34:129–153, 1956.

21 James Orlin. Contentment in graph theory: Covering graphs with cliques. *Indagationes Mathematicae (Proceedings)*, 80(5):406–424, 1977. URL: https://www.sciencedirect.com/science/article/pii/1385725877900555.

22 Jean-Éric Pin, editor. *Handbook of Automata Theory*. European Mathematical Society Publishing House, Zürich, Switzerland, 2021. doi:10.4171/Automata.

23 Senthil Rajasekaran and Moshe Y. Vardi. Verification and Realizability in Finite-Horizon Multiagent Systems. In *Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning*, pages 278–287, August 2022. doi:10.24963/kr.2022/28.

24 Emmanuel Roche and Yves Schabes. Deterministic part-of-speech tagging with finite state transducers. *Computational linguistics*, 21(2):227–253, 1995.

25 Géraud Sénizergues. The equivalence problem for deterministic pushdown automata is decidable. In *Automata, Languages and Programming, 24th International Colloquium, ICALP'97, Bologna, Italy, 7-11 July 1997, Proceedings*, pages 671–681, 1997. doi:10.1007/3-540-63165-8_221.

26 Qinglong Wang, Kaixuan Zhang, Xue Liu, and C Lee Giles. Verification of recurrent neural networks through rule extraction. *arXiv preprint arXiv:1811.06029*, 2018. doi:10.48550/arXiv.1811.06029.

27 Erkang Zhu, Silu Huang, and Surajit Chaudhuri. High-performance row pattern recognition using joins. *Proc. VLDB Endow.*, 16(5):1181–1195, January 2023. doi:10.14778/3579075.3579090.