The cost of traveling between languages

Michael Benedikt, Gabriele Puppis, and Cristian Riveros

Department of Computer Science, Oxford University
Parks Road, Oxford OX13QD UK

Abstract. We show how to calculate the maximum number of edits per character needed to convert any string in one regular language to a string in another language. Our algorithm makes use of a local determinization procedure applicable to a subclass of distance automata. We then show how to calculate the same property when the editing needs to be done in streaming fashion, by a finite state transducer, using a reduction to mean-payoff games. We show that the optimal streaming editor can be produced in PTIME.

1 Introduction

Edit distance is a well-studied metric between strings, measuring how many operations are needed to get from one string to another. In this paper we look for natural (asymmetric) analogs for regular languages: how many edits does it require to get from a word in regular language R to a word in regular language T, in the worst case? Our notation is motivated by considering R to be a *restriction* – a constraint that the input is guaranteed to satisfy – and T to be a *target* – a constraint that we want to enforce.

In a prior work [1], we considered the basic question of whether one can get from a word in R to a word in T with a finite (uniformly bounded) number of edits. One of the main results of [1] was a characterization of the pairs (R, T) for which such a uniform bound exists.

Example 1. Consider the languages $R = a^*b^*$ and $T = a^*cb^*$. Clearly, any string in R can be converted to a string in T with at most 1 edit.

Such a bound, when it exists, shows that the language R is "quite close to being a subset of T" – the gap between strings in R and strings in T is small. However, having a uniform bound on the number of edits is a strong requirement. In this paper we look not at the absolute number of edits required to get from R to T, but rather at the *percentage* of letters that need to be edited.

Example 2. Consider the languages $R = (a + b)^*$ and $T = (ab + b)^*$. Roughly, for any pair of consecutive occurrences of the letter a in the input, we will have to perform one edit in order to ensure alternation in the output. In particular, the number of edits required to get from a string in R to a string in T is unbounded. On the other hand, it is clear that we need to edit approximately half of the letters in the worst case (i.e. a^{2n}) in order to produce a string in T.

We measure the gap from R to T via the worst case, over all strings $w \in R$, of the number of edits needed to bring w into T divided by the length of w. Since we want the definition to be robust to a finite number of outliers, we take the limit of this quantity as the strings are of larger and larger length – this is the *asymptotic (normalized) cost* in getting from R to T. This gives us a measure of the distortion needed to get from R to T, lying always between 0 and 1.

Our first main result is that we can determine the asymptotic cost effectively. The algorithm relies on ideas from *distance automata* [9], and in particular on an application of determinization of distance automata, closely related to Mohri's determinization procedure [7].

We then turn to the setting where editing is required to be done in streaming fashion, producing the edits immediately on seeing the input letter. We measure a streaming edit processor by the number of edits per character it requires to get from any string in R to a string in T, again looking at the limit as the string length gets large. We define the *streaming asymptotic cost* to be the optimal cost of a streaming processor. We show that this quantity can also be calculated effectively, using techniques from mean-payoff games.

Example 3. Consider $R = (a + b)\, c^*(a^+ + b^+)$ and $T = a\, c^* a^+ + b\, c^* b^+$. One can get from R to T by only editing the initial letter: so the asymptotic cost is 0. However, a streaming strategy must commit to changing the initial letter or leaving it be: if it makes the "incorrect" choice, it will have to edit an unbounded final segment; thus the streaming asymptotic cost is 1.

The above two results give us the ability to compare the cost one should pay in editing strings in R to strings in T with an arbitrary processor with the cost when we are restricted to use a streaming processor. If these are the same, it shows that streaming processors that edit strings in R to T can approximate arbitrary processors in worst-case behavior.

In summary our contributions are:

- We present an algorithm for calculating the asymptotic cost of transforming strings in regular language R to strings in regular language T, based on locally determinizing a subclass of distance automata.
- We give an algorithm for calculating the optimal asymptotic cost achieved using a streaming editing algorithm.

Related Work. The problem of finding the minimal distance of a string to a regular language was first considered by Wagner in [10], who showed that the problem could be solved by adapting the dynamic programming approach to edit distance, giving a polynomial time algorithm. Several authors have extended the definition to deal with distances between languages. Mohri [8] defines a distance function between two sets of strings, and more generally between string distributions: in the case of languages, this is the minimum distance between two strings in the two respective languages, which is appropriate for many applications. Konstantinidis [4] focuses on the minimum distance between distinct strings within the same language, giving tractable algorithms for computing it.

Our notion of "cost" is quite distinct from this, since it is asymmetric in the two languages, focusing on the maximum of the distance of a string in one language to the other language. In our prior work [1] we have given an algorithm for determining when this distance is finite; again the paper deals with the streaming and the non-streaming setting, but the techniques used for the finiteness problem, particularly in the non-streaming case, are radically different from those used for asymptotic cost analysis. Further related work in the database area is overviewed in [1].

Organization. Section 2 defines the basic problems. Section 3 studies the non-streaming case, while Section 4 deals with the streaming case. Section 5 gives conclusions. Proofs are relegated to the full paper.

2 Problem setting

Given two words $w \in \Sigma^*$ and $u \in \Delta^*$, we denote by edit-dist(w, u) the *Levenshtein distance* (henceforth, edit distance) between w and u, which is defined as the length of a shortest sequence s of edit operations (e.g., deleting a single character, modifying a single character, and inserting a single character) that transforms w into u [11]. Following Wagner [10], we lift this to define the distance of a word to a regular language T.

$$\text{edit-dist}(w, T) \ =^{\text{def}} \ \min\big\{\, \text{edit-dist}(w, u) \ : \ u \in T \big\}.$$

We are interested in quantifying how difficult it is to edit a word in one language to obtain a word in another. That is, we have finite alphabets Σ and Δ and regular languages $R \subseteq \Sigma^*$ and $T \subseteq \Delta^*$, called the *restriction* and *target* languages, respectively. We would like to edit a string that is known to belong to the restriction language into a string in the target language.

How do we measure the cost of edits needed to get from R to T? One method is to look at the largest number of edit operations needed to get into T from strings in R: that is, the supremum over $w \in R$ of edit-dist(w, T). In an earlier work [1] we have studied for which pairs of languages (R, T) this cost is finite. However, many language pairs have infinite cost: the existence of a uniform bound to the number of edits is quite a strong property (see Example 2 in the introduction).

In this work we define an alternative notion of cost that looks at the *percentage* of symbols in a word that need to be edited. We define the *normalized cost* for editing a word w to a word in T as the fraction $\frac{\text{edit-dist}(w,T)}{|w|}$, that is, the ratio between the cost of editing w and its length. In order to measure the asymptotic behavior of the normalized cost, we define the *asymptotic cost* as the limit superior of the normalized cost when the length of words in the restriction tends to infinity. Formally, the asymptotic cost for two regular languages R and T is defined as

$$\mathbf{A}(R, T) \ =^{\text{def}} \ \lim_{n \to \infty} \sup \left\{ \frac{\text{edit-dist}(w,T)}{|w|} \ : \ w \in R, \ |w| \geq n \right\}$$

For the above definition to make sense, we always assume that R is infinite. It is easy to see that the asymptotic cost ranges over the interval $[0,1]$ of the real numbers. Indeed, for large words, one can modify and delete the letters to create shorter words in the target language and thus the resulting cost is always less than the length of the input word. We are ideally interested in computing the value $\mathbf{A}(R,T)$, provided that this number is rational.

Streaming vs non-streaming. The notion of "how much does it cost to edit a word in R to a word in T" assumes that an editing process could be any mapping from R to T (in principle, such a mapping could even fail to be computable). However, we know from [10] that there is a dynamic programming algorithm that, given a word w and a target language T represented by a deterministic finite state automaton (*DFA*) $\mathcal{T}$, computes in time $\mathcal{O}(|w|\cdot|\mathcal{T}|)$ an optimal edit sequence s such that $s(w) \in T$. In particular, this shows that optimal algorithms for editing a word can be described by functions of fairly low complexity. Sometimes it is desirable to have editing algorithms that are in even more limited classes. Perhaps the ideal case is when we can edit with a one-pass algorithm, that is, using a sequential transducer (note that we allow the transducer to have infinitely many states). Recall that a sequential transducer defines a word-to-word function; if this function happens to produce a word in T for every input $w \in R$, then we say that it is a *streaming edit strategy* for R and T. Similarly, we can consider k-lookahead transducers, with $k \in \mathbb{N}$: this type of transducer outputs words on the basis of its current state and an input $(k+1)$-character window that represents a substring of w of the form $w[i]\dots w[i+k]$, where $w[i]$ is either the i-th symbol of w, if $i \leq |w|$, or a dummy symbol $\perp$, if $i > |w|$. Accordingly, we talk about a k-lookahead streaming edit strategy.

Given a streaming edit strategy $\mathcal{S}$ for R and T and a word w, we define the cost of $\mathcal{S}$ on w to be the number of edits produced by $\mathcal{S}$ on w. Formally, letting $q_0 \xrightarrow{a_1/u_1} q_1 \xrightarrow{a_2/u_2} \dots \xrightarrow{a_n/u_n} q_n \xrightarrow{\varepsilon/u_{n+1}}$ be the run of the transducer $\mathcal{S}$ on a word $w = a_1 \dots a_n$, the *cost of* $\mathcal{S}$ *on* w, denoted $\mathrm{cost}(w,\mathcal{S})$, is the length of the final output u_{n+1} plus the sum of edit-dist(a_i,u_i) over all indices $1 \leq i \leq n$. Notice that the transducer $\mathcal{S}$ might output an additional string u_{n+1} at the end of its run in order to produce a word in the target language T. We can then define the asymptotic cost of a streaming (k-lookahead) edit strategy $\mathcal{S}$:

$$\mathbf{A}(R,\mathcal{S},T) \;=^{\mathrm{def}}\; \lim_{n\to\infty} \sup\left\{ \frac{\mathrm{cost}(w,\mathcal{S})}{|w|} \;:\; w \in R,\ |w| \geq n \right\}.$$

Finally, the *streaming* (k-lookahead) *asymptotic cost* for two languages R and T, denoted $\mathbf{SA}(R,T)$, is the *infimum* of $\mathbf{A}(R,\mathcal{S},T)$ taken over all streaming (k-lookahead) edit strategies $\mathcal{S}$ for R and T. We remark that, a priori, the infimum in the previous definition cannot be replaced by a minimum: it is conceivable that the asymptotic costs of the streaming edit strategies for R and T are arbitrary close to $\mathbf{SA}(R,T)$, but never achieve this value. In fact, in Section 4 we will show that this is not the case, as we can enforce, without loss of generality, a uniform bound to the memory of streaming edit strategies.

To stress the difference between the streaming and the non-streaming settings, we explicitly refer to the original problem as the asymptotic cost problem in the *non-streaming case*.

3 Asymptotic cost in the non-streaming case

In this section, we study the problem of computing the asymptotic cost in the non-streaming setting. We begin with some background on distance automata, which will play a key role in the main characterization result.

Distance automata computing the edit cost. Intuitively, a *distance automaton* [9] is a transducer $\mathcal{D}$ that receives as input a finite word w and outputs a corresponding cost $\mathcal{D}(w)$ in $\mathbb{N} \cup \{\infty\}$. Formally, it is a tuple $\mathcal{D} = (\Sigma, Q, E, I, F)$, where Q is a finite set of states, $E \subseteq Q \times \Sigma \times \mathbb{N} \times Q$ is a finite transition relation, I and F are some initial and final conditions described by partial functions from Q to $\mathbb{N}$ and representing the costs of beginning and ending a run with certain states. A *run* of $\mathcal{D}$ on w is a sequence $\gamma = (q_0, a_1, c_1, q_1)\,(q_1, a_2, c_2, q_2)\,\ldots\,(q_{n-1}, a_n, c_n, q_n)$ of pairwise adjacent transitions in E that spell the input word $w = a_1 a_2 \ldots a_n$. The cost of the run γ is naturally defined by

$$\mathrm{cost}(\gamma) \;=^{\mathrm{def}}\; \sum_{1 \le i \le n} c_i.$$

We denote by $\mathcal{D}(w)$ the *minimum* value $I(q_0) + \mathrm{cost}(\gamma) + F(q_n)$ among all states q_0 in the domain $\mathcal{D}om(I)$ of I, all states q_n in the domain $\mathcal{D}om(F)$ of F, and all runs γ of $\mathcal{D}$ on w that start in q_0 and end in q_n. We let $\mathcal{D}(w) = \infty$ if there are no such states q_0 and q_n, or if there is no run from q_0 to q_n.

When considering the edit distance of a word $w \in \Sigma^*$ to a regular language $T \subseteq \Delta^*$, it is fairly natural to express this value in terms of the cost computed by a distance automaton. By default, we assume that the target language T is recognized by a DFA $\mathcal{T} = (\Delta, Q, \delta, q_0, F)$ where Q is a finite set of states, $\delta \subseteq Q \times \Delta \times Q$ is a finite transition relation, q_0 and F are the initial and final set of states. Given two states p, q of $\mathcal{T}$, we let $\mathcal{T}_{p,q}$ be the DFA obtained from $\mathcal{T}$ by letting p be the new initial state and q the new unique final state. The distance automaton that computes the edit distance of a word over Σ to the target language $\mathscr{L}(\mathcal{T})$ is defined as $\mathcal{D}^{\mathrm{edit}}_{\mathcal{T}} = (\Sigma, Q, E^{\mathrm{edit}}, I^{\mathrm{edit}}, F^{\mathrm{edit}})$, where

- E^{edit} is the set of all transitions of the form (p, a, c, q), with $p, q \in Q$, $a \in \Sigma$, q reachable from p, and $c = \min\{\text{edit-dist}(a, v) \;:\; v \in \mathscr{L}(\mathcal{T}_{p,q})\}$,
- I^{edit} is the partial function that maps a state $q \in Q$ to the minimum among the values edit-dist(ε, v), with $v \in \mathscr{L}(\mathcal{T}_{q_0,q})$ (if q is not reachable from the initial state q_0, then $I^{\mathrm{edit}}(q)$ is undefined),
- F^{edit} is the partial function that maps a state $p \in Q$ to the minimum among the values edit-dist(ε, v), with $v \in \bigcup_{q \in F} \mathscr{L}(\mathcal{T}_{p,q})$ (if p cannot reach a state in F, then $F^{\mathrm{edit}}(p)$ is undefined).

One can easily show that $\mathcal{D}^{\mathrm{edit}}_{\mathcal{T}}$ computes exactly the edit distance between a word $w \in \Sigma^*$ and $\mathscr{L}(\mathcal{T})$.

Proposition 1. *For every word $w \in \Sigma^*$, we have $\mathcal{D}_{\mathcal{T}}^{\text{edit}}(w) = \text{edit-dist}(w, \mathscr{L}(\mathcal{T}))$.*

Shortcut property and determinizable components. Distance automata of the form $\mathcal{D}_{\mathcal{T}}^{\text{edit}}$ are a proper sub-class of all distance automata. In particular, they satisfy the *shortcut property*, formalized just below. Given a symbol $a \in \Sigma$ and two states p, q of a distance automaton $\mathcal{D}$, we write $p \xrightarrow{a} q$ to denote the existence in $\mathcal{D}$ of a transition (p, a, c, q) with some cost $c \in \mathbb{N}$.

Definition 1. *A distance automaton $\mathcal{D}$ satisfies the* shortcut property *if for all symbols a, b and all states p, q, r, $p \xrightarrow{a} q \xrightarrow{b} r$ implies $p \xrightarrow{a} r$ and $p \xrightarrow{b} r$.*

The following lemma shows that, in particular, $\mathcal{D}_{\mathcal{T}}^{\text{edit}}$ satisfies the shortcut property.

Lemma 1. *For every DFA $\mathcal{T}$, $\mathcal{D}_{\mathcal{T}}^{\text{edit}}$ satisfies the shortcut property.*

We call a *strongly connected component* (SCC) of a distance automaton $\mathcal{D}$ any maximal set of mutually reachable states. Given a SCC C of $\mathcal{D}$, we denote by $\mathcal{D}|C$ the sub-automaton obtained from $\mathcal{D}$ by restricting the set of states and transitions to C and by letting the initial and final conditions map any state of C to 0. Note that the transition graph of $\mathcal{D}|C$ is a clique when $\mathcal{D}$ satisfies the shortcut property.

A crucial property entailed by the shortcut property is the following one. Consider two runs ρ and ρ' of $\mathcal{D}|C$ that spell the same word w, but end in different states q and q'. If ρ and ρ' have optimal cost among all runs on $\mathcal{D}|C$ on w that end in q and q' respectively, then one can show that the difference in cost between ρ and ρ' is uniformly bounded by a constant. This implies that we can determinize $\mathcal{D}|C$ by using a subset construction, maintaining the difference between the optimal cost of reaching each state q and the overall optimal cost; this is exactly Mohri's determinization procedure [7]. Since this difference is always uniformly bounded by a constant, we get a finite-state distance automaton:

Proposition 2. *For every distance automaton $\mathcal{D}$ that satisfies the shortcut property and every SCC C of $\mathcal{D}$, the sub-automaton $\mathcal{D}|C$ can be determinized.*

The above result allows us to denote by $\det(\mathcal{D}|C)$ some deterministic distance automaton equivalent to $\mathcal{D}|C$, namely, such that $\det(\mathcal{D}|C)(w) = \mathcal{D}|C(w)$ for all $w \in \Sigma^*$. The automaton $\det(\mathcal{D}|C)$ can be computed from $\mathcal{D}|C$ by a direct exponential-time algorithm [7].

Example 4. Consider the distance automaton $\mathcal{D}$ of Figure 1, which computes the edit distance of any word to the target language $T = (ab + b)^* \; a^*$. As $\mathcal{D}$ satisfies the shortcut property and consists of two SCCs C_1 and C_2, the two sub-automata $\mathcal{D}|C_1$ and $\mathcal{D}|C_2$ can be turned into equivalent deterministic distance automata $\det(\mathcal{D}|C_1)$ and $\det(\mathcal{D}|C_2)$, depicted to the right of Figure 1.

We remark that the above result does not imply that the entire distance automaton $\mathcal{D}$ is determinizable. Consider, for instance, a distance automaton that computes the edit distance of a word w to the target language $\mathscr{L}(\mathcal{T}) =$

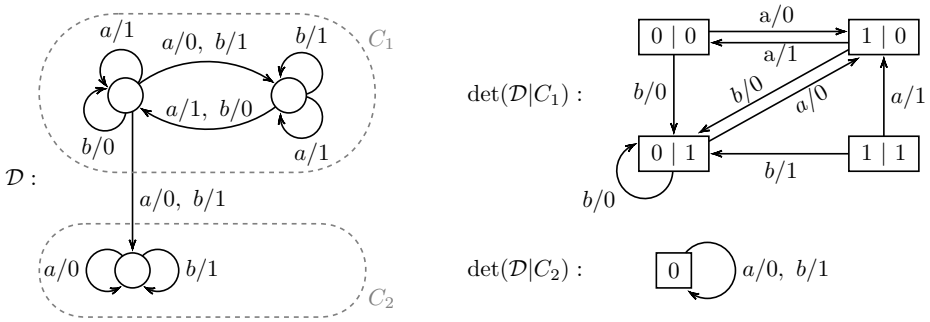

Fig. 1. A distance automaton with two SCCs and its determinized sub-automata.

$a^* + b^*$. This distance is given by the symmetric difference between the number of occurrences of a and the number of occurrences of b and hence any deterministic device that computes edit-dist$(w, \mathscr{L}(\mathcal{T}))$ must use unbounded memory.

The asymptotic cost. We give an effective characterization of the asymptotic cost $\mathbf{A}(\mathcal{D})$ of a distance automaton $\mathcal{D}$ satisfying the shortcut property:

$$\mathbf{A}(\mathcal{D}) \;=^{\text{def}}\; \lim_{n \to \infty} \sup \left\{ \tfrac{\mathcal{D}(w)}{|w|} \;:\; w \in \Sigma^*,\ |w| \geq n \right\}.$$

The characterization will imply that the above value is rational and computable from $\mathcal{D}$. Before turning to the characterization, we remark that computability of asymptotic costs does not hold for arbitrary distance automata:

Proposition 3. *The problem of deciding, given an arbitrary distance automaton $\mathcal{D}$, whether or not $\mathbf{A}(\mathcal{D}) \leq \frac{1}{2}$ is undecidable.*

We use the undecidability of the $\frac{1}{2}$*-threshold problem* for normalized costs induced by distance automata [5], which consists of deciding, given a distance automaton $\mathcal{D}$, whether $\frac{\mathcal{D}(w)}{|w|} \leq \frac{1}{2}$ holds for all words $w \in \Sigma^*$. The reduction is done by transforming a given distance automaton $\mathcal{D}$ into a new distance automaton $\mathcal{D}'$ such that $\mathbf{A}(\mathcal{D}') = \sup \left\{ \frac{\mathcal{D}(w)}{|w|} \;:\; w \in \Sigma^* \right\}$.

Next we explain how the shortcut property helps in computing the asymptotic cost. One can show that the problem of computing $\mathbf{A}(\mathcal{D})$ for a distance automaton $\mathcal{D}$ that is *deterministic* is reducible to the problem of computing normalized costs of simple cycles. Formally, a *simple cycle* is any run of $\det(\mathcal{D})$ that is a cycle (i.e., that starts and ends in the same state) but that does not contain proper sub-cycles. It is then easy to show that for a deterministic distance automaton $\mathcal{D}$, $\mathbf{A}(\mathcal{D})$ coincides with the maximum of $\frac{\text{cost}(L)}{|L|}$ among all simple cycles L of $\mathcal{D}$, where cost(L) denotes the cost of the simple cycle L. Thus by Proposition 2, calculation with simple cycles suffices to compute the asymptotic cost of any distance automaton satisfying the shortcut property and having a *single* SCC.

We consider now the more general case of a distance automaton $\mathcal{D}$ satisfying the shortcut property and having many SCCs, say $C_1, \ldots, C_k$. The situation in

this case is slightly more complicated, as $\mathbf{A}(\mathcal{D})$ cannot be expressed as a function of $\mathbf{A}(\mathcal{D}|C_1), \ldots, \mathbf{A}(\mathcal{D}|C_k)$. We define $\bar{\mathcal{D}}$ as the deterministic *multi-distance* automaton obtained from the synchronous product of $\det(\mathcal{D}|C_1)$, ..., $\det(\mathcal{D}|C_k)$ and we denote by $L_1, \ldots, L_m$ the simple cycles of $\bar{\mathcal{D}}$. Moreover, given $1 \leq i \leq m$ and $1 \leq j \leq k$, we denote by $\mathrm{cost}_j(L_i)$ the cost of the projection of the simple cycle L_i into the j-th component of $\bar{\mathcal{D}}$. Assuming that $\mathcal{D}$ is *trim*, namely, all its states are reachable from some states in $\mathcal{D}om(I)$ and they can reach some states in $\mathcal{D}om(F)$, we can characterize the asymptotic cost of $\mathcal{D}$ as follows:

Theorem 1. *For every distance automaton $\mathcal{D}$ satisfying the shortcut property,*

$$\mathbf{A}(\mathcal{D}) \;=\; \underbrace{\max_{\alpha_1,\ldots,\alpha_m \geq 0} \; \min_{1 \leq j \leq k} \; \frac{\sum_{1 \leq i \leq m} \alpha_i \cdot \mathrm{cost}_j(L_i)}{\sum_{1 \leq i \leq m} \alpha_i \cdot |L_i|}}_{\mathbf{E}(\mathcal{D})}. \tag{1}$$

The idea underlying the above characterization is that the asymptotic cost $\mathbf{A}(\mathcal{D})$ is achieved by repetitions of simple cycles in $\bar{\mathcal{D}}$. Indeed, the parameters $\alpha_1, \ldots, \alpha_m$ represent a correlation between the numbers of repetitions of the various simple cycles, and the index j represents the SCC of $\mathcal{D}$ that optimizes the normalized cost of these repetitions. The proof will consist of establishing two inequalities. In one case, we argue that all words can be approximated in cost by repetitions of simple cycles, and that the cost of editing these words is at most the cost of a "homogeneous strategy" that edits all cycles in the same component of $\mathcal{D}$. For the other inequality, we present a large family of words for which the best strategy is nearly homogeneous. The words will consist of nested repetitions of simple cycles in such a way that any edit strategy stabilizes by editing in the same component.

Example 5. Consider again the distance automaton $\mathcal{D}$ of Figure 1, with the two SCCs C_1 and C_2. The determinized sub-automaton $\det(\mathcal{D}|C_1)$ has four different simple cycles: one spelling aa with cost 1, one spelling ab with cost 0, one spelling b with cost 0, and one spelling aba with cost 1. Similarly, the determinized sub-automaton $\det(\mathcal{D}|C_2)$ has two simple cycles: one spelling a with cost 0, and the other spelling b with cost 1. Hence $(aa)^n$ is a family of words achieving a worst-case asymptotic cost of $\lim \frac{n}{2n} = \frac{1}{2}$ for the sub-automaton $\mathcal{D}|C_1$, and b^n is a family of words achieving a worst-case asymptotic cost of $\lim \frac{n}{n} = 1$ for the sub-automaton $\mathcal{D}|C_2$. However, a^{2n} is not a worst-case for $\mathcal{D}|C_2$ (as it can be repaired with asymptotic cost 0) and, symmetrically, b^n is not a worst-case for $\mathcal{D}|C_1$. This means that the worst-case asymptotic cost for $\mathcal{D}$ is achieved by a suitable combination of both families, namely, $(aab)^n$. This gives the asymptotic cost $\mathbf{A}(\mathcal{D}) = \lim \frac{n}{3n} = \frac{1}{3}$.

We make a few remarks related to the effectiveness of the characterization. First of all, we observe that the right handside term $\mathbf{E}(\mathcal{D})$ of Equation (1) can be rewritten as the following instance of a linear programming problem:

$$\texttt{maximize} \quad y \quad \texttt{subject to} \quad \begin{array}{ll} \sum_{1 \leq i \leq m} c_{i,j} \cdot x_i \geq y & \forall 1 \leq j \leq k \\ \sum_{1 \leq i \leq m} x_i \leq 1, \quad x_i \geq 0 & \forall 1 \leq i \leq k. \end{array}$$

where, for every $1 \leq i \leq m$ and every $1 \leq j \leq k$, $c_{i,j} = \frac{\mathrm{cost}_j(L_i)}{|L_i|}$. Intuitively, the variables $x_1, \ldots, x_m$ represent the values $\alpha_1 \cdot |L_1|$, ..., $\alpha_m \cdot |L_m|$ normalized in such a way that they sum up to 1, and the variable y represents an under-approximation of the value $\mathbf{E}(\mathcal{D})$. It is also known [6] that the optimal choices for the parameters $x_1, ..., x_m, y$ can be found at the 'corners' of the $(m+1)$-dimensional polyhedron that results from the intersection of the finitely many half-spaces defined by the above linear inequalities. This explains why we put $\max_{\alpha_1,...,\alpha_m \geq 0}$ instead of $\sup_{\alpha_1,...,\alpha_m \geq 0}$ in Equation (1). Moreover, it also implies that the asymptotic cost $\mathbf{A}(\mathcal{D})$ is a rational number.

Regarding the complexity of the problem of computing $\mathbf{A}(\mathcal{D})$, we observe that (i) the size $|\bar{\mathcal{D}}|$ of the multi-distance automaton $\bar{\mathcal{D}}$ is exponential in $|\mathcal{D}|$, (ii) each simple cycle L_i has length at most linear in $|\bar{\mathcal{D}}|$, (iii) the number m of simple cycles is exponential in $|\bar{\mathcal{D}}|$, and (iv) each constant $c_{i,j} = \frac{\mathrm{cost}_j(L_i)}{|L_i|}$ can be computed in time polynomial in $|\bar{\mathcal{D}}|$ and $|L_i|$. Overall, the problem of computing the asymptotic cost of $\mathcal{D}$ is reduced, in time doubly exponential, to an instance of a linear programming problem. The latter problem is known to be in PTIME [3], which proves that $\mathbf{A}(\mathcal{D})$ can be computed in doubly exponential time.

From the cost of distance automata to the cost of editing. Theorem 1, together with Proposition 1 and Lemma 1, gives a way of computing the asymptotic cost $\mathbf{A}(\Sigma^*, \mathcal{T})$ of editing arbitrary words in Σ^* to words in $\mathscr{L}(\mathcal{T})$. Here we show how to generalize to our original problem, which involves the presence of both a restriction and a target language. We first modify the definition of asymptotic cost for a distance automaton to include the presence of a restriction language $\mathscr{L}(\mathcal{R})$ recognized by a DFA $\mathcal{R}$:

$$\mathbf{A}(\mathcal{R}, \mathcal{D}) \ =^{\mathrm{def}} \lim_{n \to \infty} \sup \left\{ \frac{\mathcal{D}(w)}{|w|} \ : \ w \in \mathscr{L}(\mathcal{R}),\ |w| \geq n \right\}.$$

Given a DFA $\mathcal{R}$ and a distance automaton $\mathcal{D}$ satisfying the shortcut property, we denote by $\mathcal{D}ag(\mathcal{R})$ (resp., $\mathcal{D}ag(\mathcal{D})$) the directed acyclic graph of the SCCs of $\mathcal{R}$ (resp., $\mathcal{D}$). The paths in $\mathcal{D}ag(\mathcal{R})$ (resp., $\mathcal{D}ag(\mathcal{D})$) are the sequences of SCCs of the form $\pi = C_1 \ldots C_h$, where each SCC C_{l+1} is reachable from the previous SCC C_l. Given a SCC B of $\mathcal{R}$, we denote by $L_{B,1}, \ldots, L_{B,m_B}$ the simple cycles of the automaton $\bar{\mathcal{D}} \times (\mathcal{R}|B) = \det(\mathcal{D}|C_1) \times \ldots \times \det(\mathcal{D}|C_k) \times (\mathcal{R}|C)$, where $C_1, \ldots, C_k$ are the SCCs of $\mathcal{D}$ and $\mathcal{R}|B$ is the sub-automaton obtained from $\mathcal{R}$ by restricting the set of states to B (it does not matter which state is chosen to be initial in $\mathcal{R}|B$). Finally, given a simple cycle $L_{B,i}$ of $\bar{\mathcal{D}} \times (\mathcal{R}|B)$ and a SCC C of $\mathcal{D}$, we denote by $\mathrm{cost}_C(L_{B,i})$ the cost of a the projection of $L_{B,i}$ into the component C of $\bar{\mathcal{D}} \times (\mathcal{R}|B)$. The generalized characterization result is as follows:

Theorem 2. *For every (trim) DFA $\mathcal{R}$ and every distance automaton $\mathcal{D}$ satisfying the shortcut property,*

$$\mathbf{A}(\mathcal{R}, \mathcal{D}) \ = \max_{\substack{\tau = B_1 \ldots B_h \in \mathcal{D}ag(\mathcal{R}) \\ \alpha_{1,1},\ldots,\alpha_{1,m_1} \geq 0 \\ \ldots \\ \alpha_{h,1},\ldots,\alpha_{h,m_h} \geq 0}} \ \min_{\pi = C_1 \ldots C_h \in \mathcal{D}ag(\mathcal{D})} \frac{\sum_{1 \leq l \leq h} \sum_{1 \leq i \leq m_l} \alpha_{l,i} \cdot \mathrm{cost}_{C_l}(L_{B_l,i})}{\sum_{1 \leq l \leq h} \sum_{1 \leq i \leq m_l} \alpha_{l,i} \cdot |L_{B_l,i}|}$$

Using arguments similar to the complexity analysis in the unrestricted case, we obtain that the asymptotic cost $\mathbf{A}(\mathcal{R},\mathcal{T})$ ($= \mathbf{A}(\mathcal{R},\mathcal{D}^{\text{edit}}_{\mathcal{T}})$) for two DFA $\mathcal{R}$ and $\mathcal{T}$ is computable in 2EXPTIME.

4 Asymptotic cost in the streaming case

Here we characterize the asymptotic cost in the streaming setting in terms of the value of a mean-payoff game [2]. A *mean-payoff game* is an infinite, turn-based game played over an arena $\mathcal{M} = (V, E, v_0)$, where V is the union of two disjoint finite sets of vertices, V_{Adam} (owned by player Adam) and V_{Eve} (owned by player Eve), $E \subseteq V \times \mathbb{N} \times V$ is a finite set of weighted edges, and $v_0 \in V$ is an initial vertex. The game starts at v_0 and, at each round, the player who owns the current vertex v moves along an edge $(v, c, v') \in E$. The reward for Adam (resp., Eve) in an infinite play $\pi = (v_0, c_1, v_1)\ (v_1, c_2, v_2)\ldots$ is given by the value ν^{π}_{Adam} (resp., $-\nu^{\pi}_{\text{Eve}}$), where

$$\nu^{\pi}_{\text{Adam}} =^{\text{def}} \liminf_{n\to\infty} \frac{\sum_{i=1}^{n} c_i}{n} \qquad \nu^{\pi}_{\text{Eve}} =^{\text{def}} \limsup_{n\to\infty} \frac{\sum_{i=1}^{n} c_i}{n}$$

Intuitively, Adam wants to maximize ν^{π}_{Adam} and Eve wants to minimize ν^{π}_{Eve}.

It is known from [2] that mean-payoff games are positionally determined, namely, to each mean-payoff game corresponds a value ν such that Adam (resp., Eve) has a positional strategy that guarantees $\nu^{\pi}_{\text{Adam}} \geq \nu$ (resp., $\nu^{\pi}_{\text{Eve}} \leq \nu$) for all plays π that respect his (resp., her) strategy.

Let $\mathcal{R} = (\Sigma, Q, \delta, q_0, F)$ and $\mathcal{T} = (\Delta, Q', \delta', q'_0, F')$ be two trim DFA. To compute the streaming asymptotic cost $\mathbf{SA}(\mathcal{R},\mathcal{T})$, we construct the arena $\mathcal{M}^{\text{edit}}_{\mathcal{R},\mathcal{T}}$, where Adam's vertices are pairs of the form (q, q'), with $q \in Q$ and $q' \in Q'$, and Eve's vertices are pairs of the form (q, q', a), with $q \in Q$, $q' \in Q'$, and $a \in \Sigma$. The edges of the arena are triples of the form $\big((q, q'), 0, (p, q', a)\big)$, where $p = \delta(q, a)$, or of the form $\big((q, q', a), c, (q, p')\big)$, where $c = \min\{\text{edit-dist}(a, v) : v \in \mathscr{L}(\mathcal{T}_{q',p'})\}$. The initial vertex of the arena is the pair (q_0, q'_0) (so Adam moves first). Observe that the final states of $\mathcal{R}$ and $\mathcal{T}$ do not play any relevant role in this definition: this is because $\mathcal{R}$ and $\mathcal{T}$ are assumed to be trim and the costs of moving from non-final states to final states are irrelevant for the asymptotic behaviour. Furthermore, note that the game alternates between Adam and Eve, and only the second player can incur positive costs.

Below, we show that the value of the mean-payoff game over $\mathcal{M}^{\text{edit}}_{\mathcal{R},\mathcal{T}}$, multiplied by 2, coincides with the asymptotic cost in the streaming setting.

Theorem 3. *Given two DFA $\mathcal{R}$ and $\mathcal{T}$, we have $\mathbf{SA}(\mathcal{R},\mathcal{T}) = 2 \cdot \nu$, where ν is the value of the mean-payoff game over $\mathcal{M}^{\text{edit}}_{\mathcal{R},\mathcal{T}}$. Moreover, $\mathbf{SA}(\mathcal{R},\mathcal{T})$ is rational, it can be computed in polynomial time, and it is achieved by a single streaming edit strategy for $\mathscr{L}(\mathcal{R})$ and $\mathscr{L}(\mathcal{T})$ – which can also be computed in PTIME.*

Even if it seems natural that the value of the mean-payoff game over $\mathcal{M}^{\text{edit}}_{\mathcal{R},\mathcal{T}}$ determines the asymptotic cost $\mathbf{SA}(\mathcal{R},\mathcal{T})$, we remark that the proof of the above theorem is not trivial. Indeed, the mean-payoff game corresponds directly to a

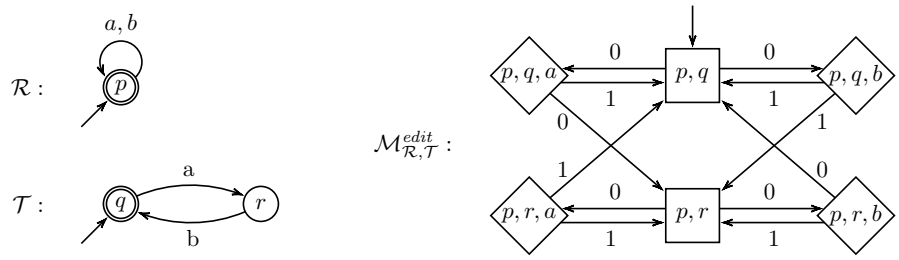

Fig. 2. Two DFA and the arena for the associated mean-payoff game.

version of the streaming edit problem where the input to the edit strategy is a sequence of prefixes of a single infinite word spelled by a run of $\mathcal{R}$. The core of the proof is to show a correspondence between this infinitary version of the streaming edit problem and the original problem as stated in Section 2. This is done by showing that $(\star)$ for the optimal strategy of Eve $\mathcal{S}$ in the mean-payoff game, one can construct a streaming edit processor $\mathcal{S}'$ of $\mathcal{R}$ into $\mathcal{T}$ such that $\frac{\mathbf{A}(\mathcal{R},\mathcal{S}',\mathcal{T})}{2}$ does not exceed the reward of $\mathcal{S}$. $\mathcal{S}'$ just mimics $\mathcal{S}$ until the string terminates, at which point it performs additional insertions to get to a final state. For the other direction we take any streaming edit processor $\mathcal{S}'$ of $\mathcal{R}$ into $\mathcal{T}$ with value $\mathbf{A}(\mathcal{R},\mathcal{S}',\mathcal{T})$ and show that no strategy $\mathcal{S}$ for Adam can guarantee a reward of more than $\frac{\mathbf{A}(\mathcal{R},\mathcal{S}',\mathcal{T})}{2}$. By the result from [2] mentioned above, this shows that Eve can guarantee a reward of at least this amount. The limit on Adam's ability is shown by combating his strategy $\mathcal{S}$ using the edit processor $\mathcal{S}'$. Putting these two directions together, we see that the optimal streaming edit processor is produced by first computing Eve's optimal strategy, then applying the transformation $(\star)$ described above: we will argue below that this is a PTIME procedure.

We recall that the problem of deciding whether the value of an arbitrary mean-payoff game $\mathcal{M} = (V, E, v_0)$ is below a certain threshold is in coNP $\cap$ NP; much recent work has focused on improving the exponential bounds on deterministic algorithms; for example, it can be done in $\mathcal{O}(|V|^2 \cdot |E| \cdot c_{\max})$, where $c_{\max}$ is the maximum weight of an edge of $\mathcal{M}$ [12]. Even though the parameter $c_{\max}$ is exponential when the weights are represented in binary notation, when restricting to arenas of the form $\mathcal{M}^{\text{edit}}_{\mathcal{R},\mathcal{T}}$, this value never exceeds the total number of states of the DFA $\mathcal{T}$. This gives the polynomial bound on the complexity of the problem of computing the value of the mean-payoff game over $\mathcal{M}^{\text{edit}}_{\mathcal{R},\mathcal{T}}$, and the PTIME bound in Theorem 3 follows.

Example 6. Consider the restriction and target language $R : (a + b)^*$ and $T : (ab)^*$ which automata and respectively mean-payoff arena $\mathcal{M}^{\text{edit}}_{\mathcal{R},\mathcal{T}}$ are shown in Figure 2. Here, diamond nodes are owned by Eve and square nodes are owned by Adam. One can easily see that an optimal positional strategy for Adam is to play $(p, q) \longrightarrow (p, q, b)$ and $(p, r) \longrightarrow (p, r, a)$. With this strategy we get that for every Eve's strategy the value ν of the mean-payoff game over $\mathcal{M}^{\text{edit}}_{\mathcal{R},\mathcal{T}}$ is equal to

$\frac{1}{2}$ and then $\mathbf{SA}(R,T) = 1$. This value definitely contrasts with the non-streaming asymptotic cost between R and T which is equal to $\frac{1}{2}$.

There is a natural generalization of the above theorem for computing the asymptotic cost of streaming edits with k-lookahead: it is indeed sufficient to modify the definition of the arena $\mathcal{M}^{\mathrm{edit}}_{\mathcal{R},\mathcal{T}}$ in such a way that Adam plays $(k+1)$-character windows. Note that this requires extending the set of vertices of $\mathcal{M}^{\mathrm{edit}}_{\mathcal{R},\mathcal{T}}$ from $\big(Q\times Q'\big)\cup\big(Q\times Q'\times\Sigma\big)$ to $\big(Q\times Q'\times(\Sigma\cup\{\bot\})^k\big)\cup\big(Q\times Q'\times\Sigma\times(\Sigma\cup\{\bot\})^k\big)$.

5 Conclusions

We have addressed the problem of computing the asymptotic cost between regular languages in the non-streaming and streaming settings. It is surprising that the asymptotic cost in both settings is rational and computable. In the streaming setting this gives us optimal online algorithms for editing one language into another, which are quite distinct from traditional edit distance algorithms based on dynamic programming. We leave as an open problem whether the algorithms for computing asymptotic cost in the nonstreaming setting are optimal.

Acknowledgments. We thank the anonymous referees, Thomas Colcombet, and Slawek Staworko for many helpful comments. The authors were supported by EPSRC (UK) grant EP/G004021/1.

References

[1] Benedikt, M., Puppis, G., Riveros, C.: Regular repair of specifications. In: LICS (2011)
[2] Ehrenfeucht, A., Mycielski, J.: Positional strategies for mean payoff games. J. of Game Theory 8, 109–113 (1979)
[3] Karmarkar, N.: A new polynomial-time algorithm for linear programming. In: STOC. pp. 302–311 (1984)
[4] Konstantinidis, S.: Computing the edit distance of a regular language. Inf. and Comp. 205(9), 1307–1316 (2007)
[5] Krob, D.: The equality problem for rational series with multiplicities in the tropical semiring is undecidable. In: ICALP. pp. 101–112 (1992)
[6] Matouek, J., Gärtner, B.: Understanding and Using Linear Programming (2006)
[7] Mohri, M.: Finite-state transducers in language and speech processing. J. of Comp. Ling. 23(2), 269–311 (1997)
[8] Mohri, M.: Edit-distance of weighted automata: general definitions and algorithms. J. of Found. of Comp. Sc. 14(6), 957–982 (2003)
[9] Simon, I.: Recognizable sets with multiplicities in the tropical semiring. In: MFCS. vol. 324, pp. 107–120 (1988)
[10] Wagner, R.: Order-n correction for regular languages. CACM 17(5), 265–268 (1974)
[11] Wagner, R., Fischer, M.: The string-to-string correction problem. JACM 21(1), 168–173 (1974)
[12] Zwick, U., Paterson, M.: The complexity of mean payoff games on graphs. Theor. Comput. Sci. 158, 343–359 (1996)