IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 13, 2021, accepted August 13, 2021, date of publication August 16, 2021, date of current version August 24, 2021.

*Digital Object Identifier 10.1109/ACCESS.2021.3105607*

# Fast Algorithms for Computing the Statistics of Pattern Matching

**DANNA ZHANG**[1] **AND KAI JIN**[2]
[1]School of Intelligent Systems Engineering, Sun Yat-sen University, Shenzhen 510275, China
[2]Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong, China

Corresponding author: Kai Jin (cscjjk@gmail.com)

This work was supported in part by the Fundamental Research Funds for the Central Universities, Sun Yat-sen University, under Grant 19LGPY292, and in part by the National Natural Science Foundation of China under Grant 62002394.

**ABSTRACT** Pattern matching is a fundamental problem in theoretical computer science. The algorithms for pattern matching and the study on the statistics of pattern matching have found enormous applications in practical fields. In this paper, we revisit the Markov embedding approach for studying pattern matching in repeated experiments. For any pattern of length $m$ over alphabet $\Sigma$, we show that the mean and variance of the waiting time of the pattern in iid experiments can be computed in $O(m)$ time based on Markov embedding technique, improving over the $O(|\Sigma| \cdot m)$ and $O(m^2)$ naïve bounds. Our method extends to computing the $k$-th moment of the waiting time, and it extends to computing other related statistics about pattern matching in repeated experiments, and it also extends to the case of Markov dependent experiments.

**INDEX TERMS** Pattern matching, statistics, failure function, Markov chain, KMP algorithm, automaton.

## I. INTRODUCTION

The distribution and moments of waiting time of a pattern in repeated experiments are classic statistics which have been studied since 1960s [1], [2]. They have enormous applications in computer science [3]–[5], telecommunication [6], industrial quality control [7], and especially molecular biology [8]–[15]. Due to the practical importance, several approaches were invented for analyzing or computing them. Guibas and Odlyzko [16] invented the combinatorial approach. They computed the generating function of the number of strings with any fixed length which do not contain the given pattern. Breen *et al.* [17] obtained similar results via a probabilistic approach, in which the renewal theory of Feller [18] was used. Li [19] studied it via martingale theory and especially Doob's fundamental theorem on stopping times. His approach was further developed in [20]–[24].

A common drawback of the aforementioned approaches is that they entail tedious mathematics and heavy probability theory, even for the *single pattern case* where the pattern equals to a string (which is much easier than the *compound pattern case* where the pattern is formed by several strings).

To compute the $k$-th moment of the waiting time, another approach is based on *Markov embedding* [12], [25]–[28]. Once the *model* of the experiment is settled (either iid or Markov dependent) and a single pattern $A$ is fixed, we can build a corresponding Markov chain $\mathsf{Mc}(A)$ whose stopping time equals the waiting time of $A$, and it reduces to computing the $k$-th moment of the stopping time of $\mathsf{Mc}(A)$, which further reduces to solving a sequence of $k$ linear systems.

This approach is quite intuitive yet it would be slow for lengthy patterns like DNA sequences, since the dimension of each underlying linear system equals the length of the pattern, and a standard linear system solver is time consuming.

In this paper, we present an optimal $O(|A|)$ time algorithm for solving each of the linear systems invoked by the Markov embedding approach (for the iid model), where $|A|$ denotes the length of $A$, and thus obtain an efficient algorithm for computing the $k$-th moment of the waiting time of $A$.

On top of it, we investigate a class $\mathcal{C}$ of linear systems (see (1)) that contains all the linear systems mentioned above, and give an $O(|A|)$ time solver for systems in $\mathcal{C}$. (Note that it does not store explicitly the coefficient matrix of these systems.)

Apart from the $k$-th moment of the waiting time, many statistics related to pattern matching (specifically, searching $A$ in repeated experiments), e.g., the *fundamental matrix of* $\mathsf{Mc}(A)$ [29, p. 99] and the $k$-th moment of *the number of comparisons spent by the KMP (Knuth-Morris-Pratt) algorithm* [30] (denote this number by $\mathsf{K}$ henceforth) reduce to instances

The associate editor coordinating the review of this manuscript and approving it for publication was Usama Mir.

in $\mathcal{C}$. We thus obtain fast algorithms for computing these statistics as well. It is worth pointing out that the reduction of the $k$-th moment of $\mathsf{K}$ is nontrivial and is not shown elsewhere as far as we know, so we present it in IV-B. A key merit of our technique is that it not only applies to the iid model but also applies to the Markov model (see IV-C).

Technically, solving the linear system in $O(|A|)$ time and proving the reduction from the $k$-th moment of $\mathsf{K}$ to solving the linear systems are difficult and require innovations. The failure function $\pi$ exploited by the KMP algorithm will play an important role in obtaining these results (we will see several applications of $\pi$ throughout this paper). Moreover, some algorithmic skills are required in proving the reduction.

*Merits OF Our Approach Compared With Others:*

1. Simple. Although, we deduce many formulas, the mathematics in our approach are elementary and easy to follow.

2. Powerful. Unlike our approach, the previous approaches do not extend easily to the higher moment and the Markov model and are not applicable for finding the moments of $\mathsf{K}$.

### A. FORMAL DESCRIPTION OF THE PROBLEM

Consider an experiment whose *outcomes* belong to $\Sigma$, where $|\Sigma| = n$ (we usually denote the outcomes by distinct lowercase letters). Let the experiment be performed repeatedly and denote the sequence of outcomes by $\{Z_i (i \geq 1)\}$.

In most part of this manuscript, $Z_1, Z_2, \ldots$ are assumed to be independent and identical distributed (referred to as *iid model*). This means that for any fixed $\sigma \in \Sigma$, the probabilities $\Pr(Z_i = \sigma)$ for all $i$ are the same, denoted by $p_\sigma$. We address the case where the outcomes $Z_1, Z_2, \ldots$ are Markov dependent (referred to as *Markov model*) only in IV-C.

Fix a finite sequence of outcomes $A = A_1, \ldots, A_m$, where $A_i \in \Sigma$ and $p_{A_i} > 0$ for $1 \leq i \leq m$. The substring $A_i \ldots A_j$ of $A$ is denoted by $A(i, j)$. In particular, $A(1, i)$ $(i \geq 0)$ is a *prefix* of $A$. Note that $A(1, 0)$ is the empty string $\varepsilon$.

A search of $A$ in the iid random sequence $Z_1 Z_2 \cdots$ corresponds to a running of the following Markov chain $\mathsf{Mc}(A)$.

*Definition 1:* For each $i$ $(0 \leq i \leq m)$, build a state $s_i$ labeled with prefix $A(1, i)$, where $s_m$ is the unique *absorbing state* and $s_0, \ldots, s_{m-1}$ are *transient state*. For each *transient state* $s_i$ and $\sigma \in \Sigma$, construct an edge from $s_i$ to $s_{\delta(i,\sigma)}$ with *associated probability* $p_\sigma$, where $\delta(i, \sigma) = \max\{h \mid A(1, h)$ is a suffix of $(A(1, i)+\sigma)\}$ (operation '+' means concatenation). This gives the markov chain $\mathsf{Mc}(A)$. Fig. 1 shows an example where $\Sigma = \{$'$a$','$b$','$c$'$\}$ and $A =$ "$abac$".

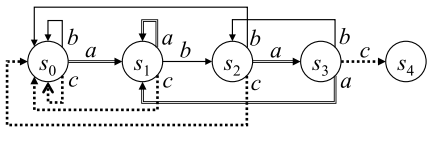


**FIGURE 1.** The Markov chain $\mathsf{Mc}(A)$ for $A =$ "$abac$" and $\Sigma = \{$'$a$','$b$','$c$'$\}$.

After running the experiment for $t$ times, $Z_1, \ldots, Z_t$ are generated. The Markov chain arrives at $s_i$ if $A(1, i)$ is the longest prefix of $A$ that is a suffix of $Z_1 \cdots Z_t$. The chain arrives at $s_m$ and is absorbed when $A$ appears in $Z_1 \cdots Z_t$. Clearly, the *waiting time of* $A$ until its first occurrence in $Z_1 Z_2 \cdots$, denoted by $\mathsf{L}$, is the *stopping time of* $\mathsf{Mc}(A)$.

The number of comparisons spent by the KMP algorithm during its search of $A$ in $Z_1 Z_2 \cdots$ (Line 2 of Algorithm 1), denoted by $\mathsf{K}$, equals the total length of edges (see below) travelled by the Markov chain (proved in Appendix A).

*Definition 2:* The edge from $s_i$ to $s_{i,\sigma}$ has an *associated length* $\ell(i, \sigma)$ which equals the number of $j$'s in $\{0, \ldots, i\}$ such that $A(1, j)$ is a prefix of $A(1, i)$ and that $j \geq \delta(i, \sigma) - 1$.

Our problem is: Given $A = A_1, \ldots, A_m$ and $\{p_\sigma \mid \sigma \in \Sigma\}$, how can we compute the $k$-th moment of $\mathsf{L}$ and $\mathsf{K}$?

### B. MORE PRELIMINARIES

Throughout, $[\cdot]$ is the Iverson bracket, and $\mathbb{E}X$ and $\mathbb{D}X$ denote the mean and variance of random variable $X$, respectively. Let $\pi : \{1, .., m\} \to \{0, .., m-1\}$ be the *failure function* (a.k.a. *prefix function*) of $A$ [31, p. 1003]. This means that $\pi_i = \max\{h \mid 0 \leq h < i$ and $A(1, h)$ is a suffix of $A(1, i)\}$. It is well known that $\pi$ can be computed in $O(m)$ time [30], [31]. Henceforth in this paper, assume $\pi$ is precomputed.

The KMP algorithm utilizes $\pi$ to search a pattern in a text. The kernel of the searching process is shown in Algorithm 1.

**Algorithm 1** Kernel of the KMP Algorithm for Searching $A$

**Require:** We are at $s_i$ $(i < m)$ and the next outcome is $\sigma$

```
1: while TRUE do
2:   if σ = A[i + 1] then
3:     i ← i + 1; break;
4:   else
5:     if i > 0 then i ← π_i; else break; end if
6:   end if
7: end while
```

### C. OUR RESULTS

We study the following linear system of $m$ unknowns and $m$ equations. ($\sigma$ is taken over $\Sigma$ when its scope is unspecified.)

$$x_i = \sum_\sigma p_\sigma \cdot x_{\delta(i,\sigma)} + c_i \ (0 \leq i \leq m-1). \tag{1}$$

Note: This system employs a "dummy variable" $x_m$ for simplicity; $x_m$ is assigned with 0 and is **not** a variable of the system; it would appear in (1) since $\delta(m-1, A_m) = m$.

1) We solve system (1) for any given vector $\mathbf{c}$ in $O(m)$ time, improving over a naïve $O(mn)$ time algorithm and an $O(m+n)$ time algorithm given in [32].
2) Built upon the first result, we give an $O(k^2 m)$ time algorithm for computing the $k$-th moment of $\mathsf{L}$. The method applies to the Markov model case (see IV-C).
3) We also design an $O(k^3 m)$ time algorithm for computing the $k$-th moment of $\mathsf{K}$ based on our first result.
4) We show applications of our algorithm in computing the fundamental matrix of $\mathsf{Mc}(A)$ (see II-E) and in solving the Penney's game problem (see IV-D).

5) We show elementary proofs for two known formulas (11), (12) about the mean and variance of $\mathsf{L}$ (see III).

Our algorithm utilizes interesting properties of $\mathsf{Mc}(A)$. The theory of Markov chain is **not** applied in our method.

### D. RELATED WORK

Many aspects of pattern occurrence in repeated experiments have been studied. Books on this topic include [4], [27], [28]. We list some results concerning a fixed number of experiments in the following. Fudos *et al.* [33] studied the probability of exact $r$ occurrences of a pattern in $k$ experiments under iid model. Extension to Markov model is conducted in [23], [34], [35]. Régnier [36], Régnier and Szpankowski [10] Régnier studied the mean and variance of the number of occurrence of a given pattern in $k$ experiments under iid or Markov model, for non-overlap or overlap counting.

Some other related work will be mentioned in IV-D.

## II. SOLVE THE LINEAR SYSTEM

First, we state an equivalent form of (1):

$$y_i = \sum_{\sigma} p_\sigma \cdot y_{\delta(i,\sigma)} - c_i \ (0 \le i \le m-1). \tag{2}$$

In (2), we assume that $y_0$ *is a dummy variable which equals* 0. Thus, there are only $m$ unknowns $y_1, \ldots, y_m$. (The dummy variable $y_0$ appears in (2) since $\delta(i,\sigma)$ could be 0.)

Lemma 1 shows the equivalence between (1) and (2).

*Lemma 1: Assume* $\mathbf{x} = (x_0, \ldots, x_{m-1}) \in \mathbb{R}^m$ *and* $x_m = 0$. *Let* $\phi(x_0, \ldots, x_{m-1})$ *be the unique* $\mathbf{y} = (y_1, \ldots, y_m) \in \mathbb{R}^m$ *such that the following holds:* $y_i = x_0 - x_i$ $(0 \le i \le m)$ *(note that* $y_0 = 0$*). We claim that* $\mathbf{x}$ *is a solution to* (1) *if and only if* $\mathbf{y} = \phi(\mathbf{x})$ *is a solution to* (2).

*Proof:* "Only if": Assume $\mathbf{x}$ is a solution to (1). For $0 \le i \le m-1$,

$$\begin{aligned} y_i &= x_0 - x_i \\ &= x_0 - \left( \sum_{\sigma} p_\sigma \cdot x_{\delta(i,\sigma)} + c_i \right) \quad \text{(due to (1))} \\ &= \sum_{\sigma} p_\sigma \cdot (x_0 - x_{\delta(i,\sigma)}) - c_i \quad \left(\text{because } \sum_{\sigma} p_\sigma = 1\right) \\ &= \sum_{\sigma} p_\sigma \cdot y_{\delta(i,\sigma)} - c_i. \end{aligned}$$

"If" part: Assume $\mathbf{y}$ is a solution to (2). For $0 \le i \le m-1$,

$$\begin{aligned} x_i &= x_0 - y_i \\ &= x_0 - \left( \sum_{\sigma} p_\sigma \cdot y_{\delta(i,\sigma)} - c_i \right) \quad \text{(due to (2))} \\ &= \sum_{\sigma} p_\sigma \cdot (x_0 - y_{\delta(i,\sigma)}) + c_i \quad \left(\text{because } \sum_{\sigma} p_\sigma = 1\right) \\ &= \sum_{\sigma} p_\sigma \cdot x_{\delta(i,\sigma)} + c_i. \end{aligned}$$

□

As a corollary of Lemma 1, solving (1) reduces to solving (2): After computing $y_1, \ldots, y_m$, we can first compute $x_0 = y_m + x_m = y_m$, and then compute $x_i = x_0 - y_i$ $(i > 0)$.

### A. SOLVE THE ABOVE SYSTEM ABOUT y

In this subsection, we show that there is a unique solution to (2) and we compute the solution in linear time. Denote

$$z_i = \sum_{\sigma \ne A_{i+1}} p_\sigma \cdot y_{\delta(i,\sigma)} \quad (0 \le i < m). \tag{3}$$

We can reformulate (2) as follows.

$$y_{i+1} = (y_i - z_i + c_i)/p_{A_{i+1}} \quad (0 \le i \le m-1). \tag{4}$$

*Proof:* Because $\delta(i, A_{i+1}) = i+1$,

$$\sum_{\sigma} p_\sigma \cdot y_{\delta(i,\sigma)} = z_i + p_{A_{i+1}} \cdot y_{i+1}.$$

So, the equation $y_i = \sum_\sigma p_\sigma \cdot y_{\delta(i,\sigma)} - c_i$ in (2) is equivalent to $y_i = z_i + p_{A_{i+1}} \cdot y_{i+1} - c_i$, i.e., the one in (4). □

According to (3), $z_i$ is a linear combination of $y_0, y_1, \ldots, y_i$, because $\delta(i,\sigma) \le i$ when "$i < m$ and $\sigma \ne A_{i+1}$". This implies that (4) has a unique solution. First, the dummy variable $y_0$ equals 0. Moreover, $y_1$ is uniquely determined by the equation taking $i = 0$ in (4), and $y_2$ is uniquely determined by the equation taking $i = 1$. So on and so forth, $y_m$ is uniquely determined by the equation taking $i = m-1$. Consequently, both (2) and (1) have a unique solution.

The above process of determining $y_1, \ldots, y_m$ in sequence implies a trivial algorithm for solving (4). Assume $y_0, \ldots, y_i$ have been computed. We can compute $z_i$ from $y_0, \ldots, y_i$ using (3) and compute $y_{i+1}$ from $z_i, y_i, c_i$. This algorithm runs in $O(n \cdot m)$ time, as it takes $O(n)$ time to compute $z_i$.

Next, we optimize the algorithm by using a faster method to compute $z_i$, which is based on the following equation.

$$z_i = z_{\pi_i} + p_{A_{\pi_i+1}} \cdot y_{\pi_i+1} - p_{A_{i+1}} \cdot y_{\delta_i}, \quad (1 \le i < m) \tag{5}$$

where $\delta_i$ denotes $\delta(\pi_i, A_{i+1})$ for $1 \le i < m$.

To prove (5), we need the following equation which is the basis of the KMP algorithm (proof can be found in [30]).

$$\delta(i,\sigma) = \begin{cases} i+1, & \sigma = A_{i+1}; \\ \delta(\pi_i, \sigma), & \sigma \ne A_{i+1}. \end{cases} \quad \begin{pmatrix} 1 \le i < m \\ \sigma \in \Sigma \end{pmatrix} \tag{6}$$

*Proof of* (5):

$$\begin{aligned} z_i &= \sum_{\sigma \ne A_{i+1}} p_\sigma \cdot y_{\delta(i,\sigma)} \overset{\text{by (6)}}{=} \sum_{\sigma \ne A_{i+1}} p_\sigma \cdot y_{\delta(\pi_i,\sigma)} \\ &= \sum_{\sigma} p_\sigma \cdot y_{\delta(\pi_i,\sigma)} - p_{A_{i+1}} y_{\delta(\pi_i, A_{i+1})} \\ &= \sum_{\sigma} p_\sigma \cdot y_{\delta(\pi_i,\sigma)} - p_{A_{i+1}} y_{\delta_i} \ \text{(by definition of } \delta_i) \\ &= \sum_{\sigma \ne A_{\pi_i+1}} p_\sigma \cdot y_{\delta(\pi_i,\sigma)} + p_{A_{\pi_i+1}} y_{\pi_i+1} - p_{A_{i+1}} y_{\delta_i} \\ &\qquad \text{(apply the fact that } \delta(\pi_i, A_{\pi_i+1}) = \pi_i + 1 \\ &= z_{\pi_i} + p_{A_{\pi_i+1}} y_{\pi_i+1} - p_{A_{i+1}} y_{\delta_i}. \quad \text{(apply (3) for } \pi_i) \end{aligned}$$

□

According to (5), we can compute $z_i$ from $z_{\pi_i}$ (instead of computing it directly using the definition (3)), and our final algorithm is shown in Algorithm 2.

**Algorithm 2** Algorithm for Solving (4)

```
Compute δ_1, ..., δ_{m-1};
y_0, z_0 ← 0;
y_1 ← (y_0 − z_0 + c_0)/p_{A_1} = c_0/p_{A_1};    (applying (4))
for i = 1 to m − 1 do
    z_i ← z_{π_i} + p_{A_{π_i+1}} y_{π_i+1} − p_{A_{i+1}} y_{δ_i};    (use (5))
    y_{i+1} ← (y_i − z_i + c_i)/p_{A_{i+1}};    (applying (4))
end for
```

Clearly, $\pi_i < i, \pi_i + 1 \le i$ and $\delta_i \le i$. These inequalities respectively imply that $z_{\pi_i}, y_{\pi_i+1}$, and $y_{\delta_i}$ have been computed before computing $z_i$. Therefore, Algorithm 2 is correct.

The first step of Algorithm 2 preprocesses $\delta_1, \ldots, \delta_{m-1}$. This step will be elaborated later in this section.

Algorithm 2 runs in $O(m)$ time after the first step.

### B. EXPECTATION AND VARIANCE OF L

For $0 \le i \le m$, denote by $e_i$ the expected stopping time (of $\mathsf{Mc}(A)$) starting from $s_i$; formally,

$$e_i := \sum_{t\ge 0} t \cdot \Pr(\text{It takes } t \textit{ steps} \text{ to reach } s_m \text{ from } s_i).$$

For $0 \le i \le m$, denote by $f_i$ the expectation of the square of stopping time starting from state $s_i$; formally,

$$f_i := \sum_{t\ge 0} t^2 \cdot \Pr(\text{It takes } t \textit{ steps} \text{ to reach } s_m \text{ from } s_i).$$

*Lemma 2: 1.* $(e_0, \ldots, e_{m-1})$ *is the solution to* (1) *fixing* $(c_0, \ldots, c_{m-1}) = (1, \ldots, 1)$. *2.* $(f_0, \ldots, f_{m-1})$ *is the solution fixing* $(c_0, \ldots, c_{m-1}) = (2e_0 - 1, \ldots, 2e_{m-1} - 1)$.

*Proof:* For any $i$ $(0 \le i < m)$, by the First-Step-Analysis,

$$\begin{aligned} e_i &= \sum_\sigma p_\sigma \sum_{t\ge 0}(t+1) \\ &\quad \cdot \Pr(\text{It takes } t \text{ steps to reach } s_m \text{ from } s_{\delta(i,\sigma)}) \\ &= \sum_\sigma p_\sigma \cdot (e_{\delta(i,\sigma)} + 1) = \sum_\sigma p_\sigma \cdot e_{\delta(i,\sigma)} + 1. \end{aligned}$$

This proves part 1 of this lemma.

For any $i$ $(0 \le i < m)$, by the First-Step-Analysis,

$$\begin{aligned} f_i &= \sum_\sigma p_\sigma \sum_{t\ge 0}(t+1)^2 \\ &\quad \cdot \Pr(\text{It takes } t \text{ steps to reach } s_m \text{ from } s_{\delta(i,\sigma)}) \\ &= \sum_\sigma p_\sigma \cdot (f_{\delta(i,\sigma)} + 2e_{\delta(i,\sigma)} + 1) \\ &= (\sum_\sigma p_\sigma \cdot f_{\delta(i,\sigma)}) + 2(e_i - 1) + 1. \end{aligned}$$

This proves part 2 of this lemma. □

By Lemma 2, we can compute $\mathbf{e} = (e_0, \ldots, e_{m-1})$ by solving the linear system (1) fixing $(c_0, \ldots, c_{m-1}) = (1, \ldots, 1)$. Moreover, after the computation of $\mathbf{e}$, we can compute $\mathbf{f} = (f_0, \ldots, f_{m-1})$ by solving the linear system (1) fixing $(c_0, \ldots, c_{m-1}) = (2e_0 - 1, \ldots, 2e_{m-1} - 1)$.

Be aware that $\mathbb{E}(\mathsf{L}) = e_0$, $\mathbb{E}(\mathsf{L}^2) = f_0$ and $\mathbb{D}(\mathsf{L}) = \mathbb{E}(\mathsf{L}^2) - \mathbb{E}(\mathsf{L})^2$. It means that we can compute the mean and variance of $\mathsf{L}$ by solving (1) (for two instances).

In fact, we can compute the first $k$-th moments of $\mathsf{L}$ by solving $k$ instances of (1). This will be shown in IV-A.

### C. EXPECTATION OF K

For ease of presentation, we abuse the notation a little bit. Throughout this subsection, let $e_i$ be the expected number of *comparison consumed by the KMP algorithm* starting from $s_i$ (until the algorithm ends); formally,

$$e_i := \sum_{t\ge 0} t \cdot \Pr(\text{It takes } t \textit{ comparisons} \text{ to reach } s_m \text{ from } s_i).$$

Clearly, $\mathbb{E}(\mathsf{K}) = e_0$. We show in the following how to compute $(e_0, \ldots, e_{m-1})$ by solving an instance of (1).

Recall the definition of $\ell(i, \sigma)$ in Definition 2.

*Lemma 3: Let* $\bar{c}_i = \sum_\sigma p_\sigma \cdot \ell(i, \sigma)$. *Then,* $(e_0, \ldots, e_{m-1})$ *is the solution to* (1) *fixing* $(c_0, \ldots, c_{m-1}) = (\bar{c}_0, \ldots, \bar{c}_{m-1})$. *In other words,*

$$e_i = \sum_\sigma p_\sigma \cdot e_{\delta(i,\sigma)} + \bar{c}_i. \quad (0 \le i < m) \tag{7}$$

*Proof:* Consider any $i$ $(0 \le i < m)$. Denote $q_{\sigma,t} =$

$\Pr(\text{It takes } t \text{ comparsions to reach } s_m \text{ from } s_{\delta(i,\sigma)})$.

By the First-Step-Analysis,

$$\begin{aligned} e_i &= \sum_{t\ge 0} t \cdot \Pr(\text{It takes } t \text{ comparisons to reach } s_m \text{ from } s_i) \\ &= \sum_\sigma p_\sigma \sum\nolimits_{t\ge 0} q_{\sigma,t}(t + \ell(i,\sigma)) \\ &= \sum_\sigma p_\sigma \left(\sum\nolimits_{t\ge 0} q_{\sigma,t} t + \sum\nolimits_{t\ge 0} q_{\sigma,t}\ell(i,\sigma)\right) \\ &= \sum_\sigma p_\sigma \cdot (e_{\delta(i,\sigma)} + \ell(i,\sigma)) = \sum_\sigma p_\sigma \cdot e_{\delta(i,\sigma)} + \bar{c}_i. \end{aligned}$$

□

For any $i$ $(1 \le i < m)$ and $\sigma \in \Sigma$, it holds trivially that

$$\ell(i,\sigma) = \begin{cases} 1, & \sigma = A_{i+1}; \\ \ell(\pi_i, \sigma) + 1, & \sigma \ne A_{i+1}. \end{cases} \tag{8}$$

Recall that $\delta_i$ denotes $\delta(\pi_i, A_{i+1})$ for $1 \le i < m$. Moreover, denote $\ell_i = \ell(\pi_i, A_{i+1})$ for $1 \le i < m$. We will preprocess $\ell_1, \ldots, \ell_{m-1}$ together with $\delta_1, \ldots, \delta_{m-1}$. The preprocessing is shown in the next subsection.

The following equation is useful for computing $\bar{c}_i$.

$$\bar{c}_i = \bar{c}_{\pi_i} - p_{A_{i+1}}\ell_i + 1. \quad (1 \le i < m) \tag{9}$$

*Proof:*

$$\begin{aligned} \bar{c}_i &= \sum_\sigma p_\sigma \cdot \ell(i,\sigma) \quad \text{(by definition)} \\ &= \sum_{\sigma \ne A_{i+1}} p_\sigma \ell(i,\sigma) + p_{A_{i+1}}\ell(i, A_{i+1}) \end{aligned}$$

$$= \sum_{\sigma \neq A_{i+1}} p_\sigma(\ell(\pi_i, \sigma) + 1) + p_{A_{i+1}} \quad \text{(apply (8))}$$
$$= \sum_{\sigma \neq A_{i+1}} p_\sigma \ell(\pi_i, \sigma) + 1 \quad \text{(becasue } \textstyle\sum_\sigma p_\sigma = 1\text{)}$$
$$= \sum_{\sigma} p_\sigma \ell(\pi_i, \sigma) - p_{A_{i+1}} \ell(\pi_i, A_{i+1}) + 1$$
$$= \bar{c}_{\pi_i} - p_{A_{i+1}} \ell_i + 1. \quad \text{(by definition of } \bar{c}_{\pi_i} \text{ and } \ell_i\text{)}$$

□

Based on (9), we can compute $\bar{c}_0, \ldots, \bar{c}_{m-1}$ in sequence in $O(m)$ time (given $\sigma_1, \ldots, \sigma_{m-1}$, $\ell_1, \ldots, \ell_{m-1}$) (note that $\bar{c}_0 = 1$), and then solve (7) to obtain $e_0, \ldots, e_{m-1}$.

The above method extends to computing the $k$-th moment of $\mathsf{K}$. We present the extension in subsection IV-B.

### D. THE PREPROCESSING STEP

In our main algorithm (Algorithm 2), we need to use $\delta_1, \ldots, \delta_{m-1}$. Moreover, we also need to use $\ell_1, \ldots, \ell_{m-1}$ in computing $\mathbb{E}(\mathsf{K})$ (as shown in the previous subsection). We show how to preprocess $\delta_1, \ldots, \delta_{m-1}, \ell_1, \ldots, \ell_{m-1}$ efficiently by a Depth-First-Search (DFS) in this subsection.

**Procedure 3** **Get-sigma-D** ($j$)

1: **for** $i:\ i < m$ and $\pi_i = j$ **do**
2: $\quad \mathsf{D}_i \leftarrow \mathsf{D}_j + 1$;
3: $\quad \delta_i \leftarrow \mathsf{Tr}[A_{i+1}]$;
4: $\quad \mathsf{Tr}[A_{i+1}] \leftarrow i + 1$; (Here, $\mathsf{Tr}[*]$ becomes $\delta(i, *)$)
5: $\quad$ **Get-sigma-D** ($i$);
6: $\quad \mathsf{Tr}[A_{i+1}] \leftarrow \delta_i$; (Let $\mathsf{Tr}[*]$ return to $\delta(j, *)$)
7: **end for**

**Algorithm 4** Compute $\delta_1, \ldots, \delta_{m-1}$ and $\ell_1, \ldots, \ell_{m-1}$

$\mathsf{Tr}[1], \ldots, \mathsf{Tr}[n] \leftarrow 0$; $\mathsf{Tr}[A_1] \leftarrow 1$;
$\mathsf{D}_0 \leftarrow 0$; $\mathsf{D}_{-1} \leftarrow 0$;
**Get-sigma-D** (0);
Compute $\ell_i \leftarrow \mathsf{D}_i - \mathsf{D}_{\delta_i - 1}$ for each $i (0 < i < m)$.

The preprocessing algorithm is described in Algorithm 4.

*Lemma 4: 1. Algorithm 4 computes $\delta_1, \ldots, \delta_{m-1}$ correctly. 2. Algorithm 4 computes $\ell_1, \ldots, \ell_{m-1}$ correctly.*

*Proof:* Let us first prove the claim on $\delta_1, \ldots, \delta_{m-1}$. (To this end, we can ignore array $\mathsf{D}$ for a moment.) Observe two facts:

A. *The lines inside the for-loop (in Procedure 3) will be executed exactly once for each $i$ $(0 < i < m)$.* (trivial)

B. *Upon the for-loop in any call of* **Get-sigma-D**, *the array* $\mathsf{Tr}$ *would have been assigned with $\delta(j, 1), \ldots, \delta(j, n)$, and moreover, Line 4 updates* $\mathsf{Tr}$ *to $\delta(i, 1), \ldots, \delta(i, n)$.*

Fact B follows immediately from the recursive definition of $\delta$ in (6). Indeed, (6) implies that if $\pi_i = j$ $(0 < i < m)$, then $\delta(i, 1), \ldots, \delta(i, n)$ are almost the same as $\delta(j, 1), \ldots, \delta(j, n)$, with the only exception that $\delta(i, A_{i+1}) = i + 1$.

As a result, $\mathsf{Tr}[A_{i+1}] = \delta(j, A_{i+1}) = \delta(\pi_i, A_{j+1}) = \delta_i$ upon Line 3, therefore $\delta_i$ is computed correctly at Line 3.

We now move on to the claim on $\ell_1, \ldots, \ell_{m-1}$. First, recall the definition of $\ell(i, \sigma)$ in Definition 2.

For $i$ $(0 \le i < m)$ and $\sigma \in \Sigma$, we claim that

$$\ell(i, \sigma) = \mathsf{D}_i - \mathsf{D}_{\delta(i,\sigma)-1} + 1. \tag{10}$$

This equation can be obtained as follows. Build a sequence

$$(i, \pi(i), \pi(\pi(i)), \ldots, 0) := (j_0, j_1, \ldots, j_r),$$

which contains exactly those $j$'s for which $A(1, j)$ is a prefix of $A(1, i)$. Moreover, notice that $\mathsf{D}(j_a) = \mathsf{D}(j_{a+1}) + 1$.

It holds either (i) $\delta(i, \sigma) = j_a + 1$ for some $0 \le a \le r$ (this occurs when $\sigma \in \{A[j_0+1], \ldots, A[j_r+1]\}$), or (ii) $\delta(i, \sigma) = 0$ (this occurs if $\sigma \notin \{A[j_0 + 1], \ldots, A[j_r + 1]\}$).

In case (i), length $\ell(i, \sigma)$ equals the number of elements in $\{j_0, j_1, \ldots, j_a\}$ (according to Definition 2), which equals $\mathsf{D}_{j_0} - \mathsf{D}_{j_a} + 1 = \mathsf{D}_i - \mathsf{D}_{\delta(i,\sigma)-1} + 1$. In case (ii), length $\ell(i, \sigma)$ equals the number of elements in $\{j_0, j_1, \ldots, j_r\}$ (according to Definition 2), which equals $\mathsf{D}_{j_0} - \mathsf{D}_{j_r} + 1 = \mathsf{D}_i + 1 = \mathsf{D}_i - \mathsf{D}_{\delta(i,\sigma)-1} + 1$ (note that $\delta(i, \sigma) - 1 = -1$ and $\mathsf{D}_{-1} = 0$).

Therefore, (10) holds in either way. As a corollary of (10),

$$\ell_i = \ell(\pi_i, A_{i+1}) \overset{\text{by (10)}}{=} \mathsf{D}_{\pi_i} - \mathsf{D}_{\delta(\pi_i, A_{i+1})-1} + 1$$
$$= \mathsf{D}_{\pi_i} - \mathsf{D}_{\delta_i - 1} + 1 = \mathsf{D}_i - \mathsf{D}_{\delta_i - 1}.$$

So, Algorithm 4 computes $\ell_1, \ldots, \ell_{m-1}$ correctly. □

Algorithm 4 uses a table $\mathsf{Tr}$ of space $O(n)$ which is initialized in $O(n)$ time before **Get-sigma-D**(0) is invoked. In fact, this is the only step which may exceed $O(m)$ time; all the other steps in the algorithm sum up to $O(m)$ time. Therefore, the algorithm consumes $O(n + m)$ time.

Nevertheless, we can improve the algorithm to $O(m)$ time using perfect hash. The idea is that we only take care of those $\mathsf{Tr}[\sigma]$ for which $\sigma$ is a character in $S = \{A_1, \ldots, A_m\}$ and discard all the other useless information in $\mathsf{Tr}$. To this end, we can use the perfect hash table (with two layers) introduced in [37]. In our case, the universe is $\Sigma$ and the dictionary $S$ includes all characters that appear in $A$ ($|S| \le m$). We can build an $O(S) = O(m)$ space table for $S$, so that each looking up and inserting operation is handled in $O(1)$ time.

### E. ON THE FUNDAMENTAL MATRIX OF Mc(A)

Recall that $\mathsf{Mc}(A)$ is the Markov chain searching $A$. Let $\mathbf{P} = (P_{i,j})_{(m+1)\times(m+1)}$ be the *transition matrix* of $\mathsf{Mc}(A)$; $P_{i,j} (i, j \in \{0, \ldots, m\})$ is the probability that it moves to $s_j$ at the next step when the chain is currently at $s_i$. Let $\mathbf{Q}$ be the leading principle submatrix of $\mathbf{P}$ with order $m$, i.e., $\mathbf{Q}$ is obtained from $\mathbf{P}$ by deleting the last column and last row.

Obviously, fixing any $\mathbf{c} = (c_0, \ldots, c_{m-1})^\top \in \mathbb{R}^m$, the linear system (1) is equivalent to $\mathbf{x} = \mathbf{Q}\mathbf{x} + \mathbf{c}$. Consequently, our main result can equivalently be stated as follows:

*For any $\mathbf{c} = (c_0, \ldots, c_{m-1})^\top \in \mathbb{R}^m$, we can solve $\mathbf{x} = \mathbf{Q}\mathbf{x} + \mathbf{c}$, i.e., $\mathbf{x} = (\mathbf{I} - \mathbf{Q})^{-1}\mathbf{c}$, in $O(m)$ time.*

With this connection, we are now ready to state another important application of our main result:

*Corollary 1:* The *fundamental matrix of* $\mathsf{Mc}(A)$, *which is defined as $\mathbf{N} = (\mathbf{I} - \mathbf{Q})^{-1}$, can be computed in $O(m^2)$ time.*

Suppose we want to compute the $j$-th column of $\mathbf{N}$. Set $\mathbf{c} = (c_0, \ldots, c_{m-1})^\top$ where $c_k = [k = j]$ $(0 \le k \le m-1)$. Then, the $j$-th column of $\mathbf{N}$ is given by $\mathbf{N}\,\mathbf{c} = (\mathbf{I} - \mathbf{Q})^{-1}\,\mathbf{c}$.

*Remark 1:* According to the theory of Markov chain [29], [38], many statistics about $\mathsf{Mc}(A)$ can be computed once the fundamental matrix $\mathbf{N}$ is computed. For example, we can compute the mean and variance of *the number of visits* to each state $s_j$ (before being absorbed) starting at each state $s_i$. Also, we can compute the expected *number of visited states* and the *transient probabilities*. See the details in Appendix B.

## III. ELEMENTARY PROOFS OF TWO FORMULAS

Let $w_j = (p_{A_1})^{-1} \cdots (p_{A_j})^{-1}$ $(0 \le j \le m)$; $w_0 = 1$. Let $T_{0,i}$ be the waiting time of $s_i$ starting from $s_0$. The following two formulas of $T_{0,i}$ are proved in [19] and [3, p. 407] by applying Doob's fundamental theorem on stopping times of martingales or generating function of $T_{0,i}$. This section gives a more elementary proof using the basic properties of $\mathsf{Mc}(A)$.

$$\mathbb{E}\,T_{0,i} = \sum_{j=1}^{i} w_j \Delta, \tag{11}$$

$$\mathbb{D}\,T_{0,i} = (\mathbb{E}\,T_{0,i})^2 - \sum_{j=1}^{i} (2j-1) w_j \cdot \Delta, \tag{12}$$

where $\Delta := \big[A(1,j) = A(i-j+1, i)\big]$.

*Proof of* (11)*:* Denote by $e_i$ the expected waiting time of $s_m$ starting from $s_i$. Denote $y_i = \mathbb{E}\,T_{0,i}$, which equals $e_0 - e_i$ (this follows from the trivial fact that $\mathbb{E}\,T_{0,i} + e_i = e_0$).

Let $\mathbf{y} = (y_1, \ldots, y_m)$ and $\mathbf{x} = (e_0, \ldots, e_{m-1})$ and recall $\phi$ in Lemma 1. Since $y_i = e_0 - e_i$, we have $\mathbf{y} = \phi(\mathbf{x})$.

By Lemma 2, $\mathbf{x}$ is the unique solution to (1) fixing $\mathbf{c} = (1, \ldots, 1)$. Therefore, according to Lemma 1, $\mathbf{y} = \phi(\mathbf{x})$ is the unique solution to (2) (or (4)) fixing $\mathbf{c} = (1, \ldots, 1)$, i.e.,

$$y_{i+1} = (y_i - \sum_{\sigma \ne A_{i+1}} p_\sigma \cdot y_{\delta(i,\sigma)} + 1)/p_{A_{i+1}}. \quad (i < m) \tag{13}$$

Next, by using (13), we prove (11) by induction.

*Initial condition*: Because $y_0 = 0$, (11) holds for $i = 0$. Taking $i = 0$ in (13), we get $y_1 = (y_0 - \sum_{\sigma \ne A_1} p_\sigma \cdot y_0 + 1)/p_{A_1}$, namely, $y_1 = p_{A_1}^{-1} = w_1$. So, (11) holds for $i = 1$.

*Induction*. Assume (11) holds for $0, 1, \ldots, i$, we shall prove that (11) also holds for $i+1$. Consider all $j$ such that $A(1,j)$ is a suffix of $A(1,i)$. Denote them by $j_0, j_1, \ldots, j_r$, where $j_0 = i > j_1 > \ldots > j_r = 0$. We say $j_k$ is *alive* if there is **no** $j_h$ larger than $j_k$ such that $A_{j_h+1} = A_{j_k+1}$.

$$\begin{aligned}
y_{i+1} &= \Big(y_i - \sum_{\sigma \ne A_{i+1}} p_\sigma \cdot y_{\delta(i,\sigma)} + 1\Big)/p_{A_{i+1}} \quad \text{(by (13))} \\
&= \Big(y_i + 1 - \sum_{\sigma \ne A_{i+1}} p_\sigma \Big( \sum_{\substack{A_{j_k+1}=\sigma \\ j_k \text{ is alive}}} y_{j_k+1} \Big)\Big)/p_{A_{i+1}} \\
&= \Big(\sum_{k=0}^{r} w_{j_k} - \sum_{\sigma \ne A_{i+1}} p_\sigma \sum_{A_{j_k+1}=\sigma} w_{j_k+1}\Big)/p_{A_{i+1}} \\
&\qquad \text{(Expending terms } y_i \text{ and } y_{j_k+1} \text{ using (11))} \\
&= \Big(\sum_{j_k} w_{j_k} - \sum_{\sigma \ne A_{i+1}} \sum_{A_{j_k+1}=\sigma} w_{j_k}\Big)/p_{A_{i+1}} \\
&= \sum_{A_{j_k+1}=A_{i+1}} w_{j_k}/p_{A_{i+1}} = \sum_{A_{j_k+1}=A_{i+1}} w_{j_k+1} \\
&= \sum_{j=1}^{i+1} w_j \Big[A(1,j) = A(i-j+2, i+1)\Big].
\end{aligned}$$

□

*Proof of* (12)*:* Let $\mathbf{x}'$ be the unique solution to (1) fixing $\mathbf{c} = (y_0, \ldots, y_{m-1})$. Let $\mathbf{y}' = \phi(\mathbf{x}') = (y'_0, \ldots, y'_{m-1})$. Applying Lemma 1, $\mathbf{y}'$ is the unique solution to (2) (or (4)) fixing $\mathbf{c} = (y_0, \ldots, y_{m-1})$. Therefore,

$$y'_{i+1} = \Big(y'_i - \sum_{\sigma \ne A_{i+1}} p_\sigma \cdot y'_{\delta(i,\sigma)} + y_i\Big)/p_{A_{i+1}}. \tag{14}$$

To prove (12), we first prove the following formula of $y'$.

$$y'_i = \sum_{j=1}^{i} (j-1) w_j \Big[A(1,j) = A(i-j+1, i)\Big]. \tag{15}$$

We prove (15) by induction. By definition, $y'_0 = 0$. Take $i = 0$ in (14), we get $y'_1 = y_0/p_{A_1} = 0$. So $y'_1 = 0$. Since $y'_0 = 0$ and $y'_1 = 0$, (15) holds for $i = 0$ and $i = 1$.

Next, assume that (15) holds for $0, 1, \ldots, i$, we shall prove that (15) also holds for $i+1$. Let $j_0, \ldots, j_r$ and the concept "alive" be the same as in the proof of (11).

$$\begin{aligned}
y'_{i+1} &= \Big(y'_i - \sum_{\sigma \ne A_{i+1}} p_\sigma \cdot y'_{\delta(i,\sigma)} + y_i\Big)/p_{A_{i+1}} \text{ (by (14))} \\
&= \Big(y'_i + \sum_{j_k > 0} w_{j_k} - \sum_{\sigma \ne A_{i+1}} p_\sigma \cdot y'_{\delta(i,\sigma)}\Big)/p_{A_{i+1}} \\
&= \Big(y'_i + \sum_{j_k > 0} w_{j_k} - \sum_{\sigma \ne A_{i+1}} p_\sigma \sum_{\substack{A_{j_k+1}=\sigma \\ j_k \text{ is alive}}} y'_{j_k+1}\Big)/p_{A_{i+1}} \\
&= \Big(\sum_{j_k > 0} j_k w_{j_k} - \sum_{\sigma \ne A_{i+1}} p_\sigma \sum_{A_{j_k+1}=\sigma} j_k w_{j_k+1}\Big)/p_{A_{i+1}} \\
&\qquad \text{(Expending terms } y'_i \text{ and } y'_{j_k+1} \text{ using (15))} \\
&= \Big(\sum_{j_k} j_k w_{j_k} - \sum_{\sigma \ne A_{i+1}} \sum_{A_{j_k+1}=\sigma} j_k w_{j_k}\Big)/p_{A_{i+1}} \\
&= \sum_{A_{j_k+1}=A_{i+1}} j_k w_{j_k}/p_{A_{i+1}} = \sum_{A_{j_k+1}=A_{i+1}} j_k \cdot w_{j_k+1} \\
&= \sum_{j=1}^{i+1} (j-1) w_j \Big[A(1,j) = A(i-j+2, i+1)\Big].
\end{aligned}$$

Let $T_{i,m}$ be the waiting time of $s_m$ starting from $s_i$. Denote $f_i = \mathbb{E}(T_{i,m}^2)$ and $f = (f_0, \ldots, f_{m-1})$. We have

$$\begin{aligned}
f &= (\mathbf{I} - \mathbf{Q})^{-1}(2e_0 - 1, \ldots, 2e_{m-1} - 1)^\top \text{ (by Lemma 2)} \\
&= (\mathbf{I} - \mathbf{Q})^{-1}\Big(2(e_0 - y_0) - 1, \ldots, 2(e_0 - y_{m-1}) - 1\Big)^\top
\end{aligned}$$

$$= (2e_0 - 1)(\mathbf{I} - \mathbf{Q})^{-1}(1, \ldots, 1)^\top - 2(\mathbf{I} - \mathbf{Q})^{-1}(y_0, \ldots, y_{m-1})^\top$$
$$= (2e_0 - 1)(e_0, \ldots, e_{m-1})^\top - 2(x'_0, \ldots, x'_{m-1})^\top.$$

Therefore, $f_0 - f_i = (2e_0 - 1)(e_0 - e_i) - 2(x'_0 - x'_i) = (2e_0 - 1)y_i - 2y'_i$ $(1 \le i \le m)$. Combining this formula with (11) and (15), the result is that $\mathbb{D}\,T_{0,i}$ equals to

$$= \mathbb{D}\,T_{0,m} - \mathbb{D}\,T_{i,m} = (f_0 - f_i) + e_i^2 - e_0^2$$
$$= ((2e_0 - 1)y_i - 2y'_i) + (e_0 - y_i)^2 - e_0^2 = y_i^2 - y_i - 2y'_i$$
$$= y_i^2 - \sum_{j=1}^{i}(1 + 2(j-1))w_j\Big[A(1,j) = A(i-j+1,i)\Big].$$

(The last equation applies (11) and (15).) □

## IV. EXTENSIONS AND APPLICATIONS

### A. ON THE k-TH MOMENT OF L

For $k \ge 0$ and $0 \le i \le m$, denote

$$\begin{cases} e_i^{(k)} = \sum_{t\ge 0} t^k \cdot \Pr(\text{It takes } t \textit{ steps} \text{ to reach } s_m \text{ from } s_i), \\ c_i^{(k)} = -\sum_{l=0}^{k-1}\binom{k}{l}(-1)^{k-l}e_i^{(l)}. \end{cases}$$

*Lemma 5: For $k \ge 0$, $(e_0^{(k)}, \ldots, e_{m-1}^{(k)})$ is the solution to (1) fixing $(c_0, \ldots, c_{m-1}) = (c_0^{(k)}, \ldots, c_{m-1}^{(k)})$. In other words,*

$$e_i^{(k)} = \left(\sum_\sigma p_\sigma \cdot e_{\delta(i,\sigma)}^{(k)}\right) - \sum_{l=0}^{k-1}\binom{k}{l}(-1)^{k-l}e_i^{(l)}. \quad (16)$$

We need the following identity to prove Lemma 5. (A proof of this identity can be found in Appendix A.)

$$\sum_{h=l}^{k}\binom{k}{h}\binom{h}{l}(-1)^h = 0. \quad (k > l \ge 0) \quad (17)$$

Equivalently,

$$\sum_{h=l}^{k-1}\binom{k}{h}\binom{h}{l}(-1)^h = -\binom{k}{l}(-1)^k. \quad (k > l \ge 0) \quad (18)$$

*Proof of Lemma 5:* We prove it by induction. First, noticing that $(c_0^{(1)}, \ldots, c_{m-1}^{(1)}) = (1, \ldots, 1)$, the case $k = 1$ holds according to Lemma 2. The case $k = 0$ also holds trivially. Assume now for any $h < k$, $(e_0^{(h)}, \ldots, e_{m-1}^{(h)})$ is the solution to (1) fixing $(c_0, \ldots, c_{m-1}) = (c_0^{(h)}, \ldots, c_{m-1}^{(h)})$, and we are going to prove (16). By the First-Step-Analysis,

$$e_i^{(k)} = \sum_\sigma p_\sigma \sum_{t\ge 0}(t+1)^k \cdot \Pr(\text{It takes } t \text{ steps to reach } s_m \text{ from } s_{\delta(i,\sigma)})$$
$$= \sum_\sigma p_\sigma \sum_{t\ge 0}\left(t^k + \sum_{h=0}^{k-1}\binom{k}{h}t^h\right) \cdot \Pr(\text{It takes } t \text{ steps to reach } s_m \text{ from } s_{\delta(i,\sigma)})$$
$$= \left(\sum_\sigma p_\sigma e_{\delta(i,\sigma)}^{(k)}\right) + \sum_{h=0}^{k-1}\binom{k}{h}\sum_\sigma p_\sigma e_{\delta(i,\sigma)}^{(h)}$$
$$= \left(\sum_\sigma p_\sigma e_{\delta(i,\sigma)}^{(k)}\right) + \sum_{h=0}^{k-1}\binom{k}{h}\left(e_i^{(h)} + \sum_{l=0}^{h-1}\binom{h}{l}(-1)^{h-l}e_i^{(l)}\right)$$

(obtain the last equality from induction hypothesis)

$$= \left(\sum_\sigma p_\sigma e_{\delta(i,\sigma)}^{(k)}\right) + \sum_{h=0}^{k-1}\binom{k}{h}\sum_{l=0}^{h}\binom{h}{l}(-1)^{h-l}e_i^{(l)}$$
$$= \left(\sum_\sigma p_\sigma e_{\delta(i,\sigma)}^{(k)}\right) + \sum_{l=0}^{k-1}e_i^{(l)}\cdot\sum_{h=l}^{k-1}\left(\binom{k}{h}\binom{h}{l}(-1)^{h-l}\right)$$
$$= \left(\sum_\sigma p_\sigma e_{\delta(i,\sigma)}^{(k)}\right) - \sum_{l=0}^{k-1}e_i^{(l)}\left(\binom{k}{l}(-1)^{k-l}\right).$$

Note that the last step applies (18). □

Based on Lemma 5, we can compute $e^{(k)}$ as follows.

First, compute $c^{(k)}$ from $e^{(0)}, \ldots, e^{(k-1)}$. Then, solve the system (1) fixing $(c_0, \ldots, c_{m-1}) = (c_0^{(k)}, \ldots, c_{m-1}^{(k)})$.

Assume $e^{(1)}, \ldots, e^{(k-1)}$ have been computed. Then, it takes $O(mk)$ time for computing the constant vector $(c_0^{(k)}, \ldots, c_{m-1}^{(k)})$ according to the definition $c_i^{(k)} = -\sum_{l=0}^{k-1}\binom{k}{l}(-1)^{k-l}e_i^{(l)}$. We then obtain $e^{(k)}$ in $O(m)$ time.

To sum up, we can compute $e^{(1)}, \ldots, e^{(k)}$ altogether in $O(mk^2)$ time, and thus obtain $\mathbb{E}(\mathsf{L}^1) = e_0^{(1)}, \ldots, \mathbb{E}(\mathsf{L}^k) = e_0^{(k)}$. Furthermore, the $k$-th moment of L can easily be computed from $\mathbb{E}(\mathsf{L}^1), \ldots, \mathbb{E}(\mathsf{L}^k)$ (see [3, p. 397–399]).

### B. ON THE k-TH MOMENT OF K

For ease of presentation, we abuse the notation a little bit. In this subsection, $e_i^{(k)}$ and $c_i^{(k)}$ have different meanings compared with the last subsection. Here, we denote $e_i^{(k)} = \sum_{t\ge 0} t^k \cdot \Pr(\text{It takes } t \textit{ comparisons} \text{ to reach } s_m \text{ from } s_i)$.

For $a \ge 0, b \ge 0, 0 \le i < m$, denote $w_i^{a,b} = \sum_\sigma p_\sigma\,(\ell(i,\sigma))^a\, e_{\delta(i,\sigma)}^{(b)}$, and $c_i^{(k)} = \sum_{l=1}^{k}\binom{k}{l}w_i^{l,k-l}$.

*Lemma 6: $(e_0^{(k)}, \ldots, e_{m-1}^{(k)})$ is the solution to (1) fixing $(c_0, \ldots, c_{m-1}) = (c_0^{(k)}, \ldots, c_{m-1}^{(k)})$. In other words,*

$$e_i^{(k)} = \left(\sum_\sigma p_\sigma \cdot e_{\delta(i,\sigma)}^{(k)}\right) + \sum_{l=1}^{k}\binom{k}{l}w_i^{l,k-l}. \quad (0 \le i < m) \quad (19)$$

*Proof:* Consider any $i$ $(0 \le i < m)$. Denote $q_{\sigma,t} = \Pr(\text{It takes } t \text{ comparsions to reach } s_m \text{ from } s_{\delta(i,\sigma)})$.

By the First-Step-Analysis,

$$e_i^{(k)} = \sum_\sigma p_\sigma \sum_{t\ge 0} q_{\sigma,t}(t + \ell(i,\sigma))^k$$
$$= \sum_\sigma p_\sigma \sum_{t\ge 0} q_{\sigma,t}\sum_{l=0}^{k}\binom{k}{l}t^{k-l}(\ell(i,\sigma))^l$$
$$= \sum_\sigma p_\sigma \sum_{l=0}^{k}\binom{k}{l}(\ell(i,\sigma))^l \sum_{t\ge 0} q_{\sigma,t}t^{k-l}$$

$$= \sum_{\sigma} p_\sigma \sum_{l=0}^{k} \binom{k}{l} (\ell(i,\sigma))^l e_{\delta(i,\sigma)}^{(k-l)}$$
$$= \sum_{\sigma} p_\sigma \left( e_{\delta(i,\sigma)}^{(k)} + \sum_{l=1}^{k} \binom{k}{l} (\ell(i,\sigma))^l e_{\delta(i,\sigma)}^{(k-l)} \right)$$
$$= \left( \sum_{\sigma} p_\sigma e_{\delta(i,\sigma)}^{(k)} \right) + \sum_{l=1}^{k} \binom{k}{l} \sum_{\sigma} p_\sigma (\ell(i,\sigma))^l e_{\delta(i,\sigma)}^{(k-l)}$$
$$= \left( \sum_{\sigma} p_\sigma e_{\delta(i,\sigma)}^{(k)} \right) + \sum_{l=1}^{k} \binom{k}{l} w_i^{l,k-l}.$$

□

It remains to show how to compute the constant $c_i^{(k)}$ efficiently. This reduces to showing the computation of $w_i^{a,b}$.

We need the following formula of $w_i^{a,b} (i > 0)$.

$$w_i^{a,b} = \sum_{l=0}^{a} \binom{a}{l} w_{\pi_i}^{l,b} + p_{A_{i+1}} (e_{i+1}^{(b)} - (\ell_i + 1)^a e_{\delta_i}^{(b)}). \quad (20)$$

*Proof:*

$$\begin{aligned} w_i^{a,b} &= \sum_{\sigma} p_\sigma (\ell(i,\sigma))^a e_{\delta(i,\sigma)}^{(b)} \\ &= \sum_{\sigma \neq A_{i+1}} p_\sigma (\ell(i,\sigma))^a e_{\delta(i,\sigma)}^{(b)} + p_{A_{i+1}} e_{i+1}^{(b)} \\ &= \sum_{\sigma \neq A_{i+1}} p_\sigma (\ell(\pi_i,\sigma) + 1)^a e_{\delta(\pi_i,\sigma)}^{(b)} + p_{A_{i+1}} e_{i+1}^{(b)} \\ &= \sum_{\sigma} p_\sigma (\ell(\pi_i,\sigma) + 1)^a e_{\delta(\pi_i,\sigma)}^{(b)} \\ &\quad - p_{A_{i+1}} (\ell_i + 1)^a e_{\delta_i}^{b} + p_{A_{i+1}} e_{i+1}^{(b)} \\ &= \sum_{\sigma} p_\sigma \sum_{l=0}^{a} \binom{a}{l} \ell(\pi_i,\sigma)^l e_{\delta(\pi_i,\sigma)}^{(b)} \\ &\quad + p_{A_{i+1}} \left( e_{i+1}^{(b)} - (\ell_i + 1)^a e_{\delta_i}^{b} \right) \\ &= \sum_{l=0}^{a} \binom{a}{l} \sum_{\sigma} p_\sigma \ell(\pi_i,\sigma)^l e_{\delta(\pi_i,\sigma)}^{(b)} \\ &\quad + p_{A_{i+1}} \left( e_{i+1}^{(b)} - (\ell_i + 1)^a e_{\delta_i}^{b} \right) \\ &= \sum_{l=0}^{a} \binom{a}{l} w_{\pi_i}^{l,b} + p_{A_{i+1}} \left( e_{i+1}^{(b)} - (\ell_i + 1)^a e_{\delta_i}^{b} \right). \end{aligned}$$

□

We also need the following formula of $w_0^{a,b}$.

$$w_0^{a,b} = (1 - p_{A_1}) e_0^{(b)} + p_{A_1} e_1^{(b)}. \quad (21)$$

*Proof:*

$$\begin{aligned} w_0^{a,b} &= \sum_{\sigma} p_\sigma (\ell(0,\sigma))^a e_{\delta(0,\sigma)}^{(b)} = \sum_{\sigma} p_\sigma e_{\delta(0,\sigma)}^{(b)} \\ &= \sum_{\sigma \neq A_1} p_\sigma e_{\delta(0,\sigma)}^{(b)} + p_{A_1} e_{\delta(0,A_1)}^{(b)} \\ &= \sum_{\sigma \neq A_1} p_\sigma e_0^{(b)} + p_{A_1} e_1^{(b)} = (1 - p_{A_1}) e_0^{(b)} + p_{A_1} e_1^{(b)}. \end{aligned}$$

□

Our final algorithm is shown in Algorithm 5.

**Algorithm 5** Algorithm for Computing $e^{(1)}, \ldots, e^{(k)}$

```
(c_0^(0), ..., c_{m-1}^(0)) ← (0, ..., 0).
for b = 0 to k − 1 do
    Use c^(b) to obtain e^(b) by applying Lemma 6;
    for i = 0 to m − 1 do
        Compute w_i^{0,b}, ..., w_i^{k,b} according to (21) or (20);
        c_i^(b+1) ← Σ_{l=1}^{b+1} C(b+1, l) w_i^{l,b+1−l}; (by definition)
    end for
end for
Use c^(k) to obtain e^(k) by applying Lemma 6.
```

The correctness of Algorithm 5 follows from the above equations and discussions. It can be easily checked that every notation in the algorithm is computed before used.

We analyze the running time of Algorithm 5 in the following. Clearly, the bottleneck of this algorithm lies in computing the entries in $\{w_i^{a,b} \mid 0 \le i < m, 0 \le a \le k, 0 \le b < k\}$. Each of them costs $O(k)$ time to compute. Therefore, the total running time is $O(k^3 m)$. (The running time is higher for the case of K than the case of L. In computing the $k$-th moment of K, we have to divide $c_i^{(k)}$ into multiple parts and the computation is thus more involved and expensive.) Nevertheless, it is still $O(m)$ time when $k = O(1)$.

### C. ON THE CASE OF MARKOV MODEL

We now move on to the case where the outcomes $\{Z_i\}$ of the repeated experiments are *first-order homogeneous Markov dependent*. Let $p_{\rho,\sigma} = \Pr(Z_i = \sigma \mid Z_{i-1} = \rho)$ be the *transition probability* from outcome $\rho$ to $\sigma$, and $p_\sigma = \Pr(Z_1 = \sigma)$ be the *initial probability* of $\sigma$. A Markov chain searching $A$ under this Markov dependent model is defined as follows.

*Definition 3:* Build $m + n$ states $s_{-(n-1)}, \ldots, s_{-1}, s_0, \ldots, s_m$. The first $n - 1$ states (with negative subscripts) correspond to the $n - 1$ symbols in $\Sigma/\{A_1\}$; each of them is labeled with a different symbol in $\Sigma/\{A_1\}$. The last $m + 1$ states (with nonnegative subscripts) are corresponding to the $m + 1$ prefixes of $A$. State $s_m$ is the unique *absorbing state*. For each transient state $s_v$ and $\sigma \in \Sigma$, there is an edge from $s_v$ to $s_{\delta^1(v,\sigma)}$ where $\delta^1(v,\sigma)$ is defined as follows. See Fig. 2.

For $v \in \Sigma \setminus \{A_1\}$, $\delta^1(v,\sigma) = \begin{cases} \sigma, & \sigma \neq A_1; \\ 1, & \sigma = A_1. \end{cases}$.

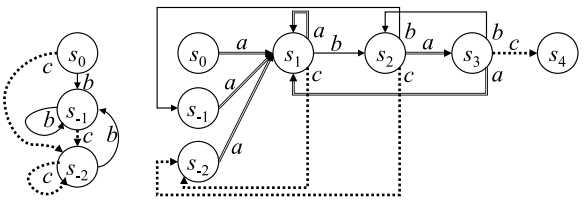


**FIGURE 2.** The Markov chain searching $A$ = "*abac*" under Markov model where $\Sigma$ = {'*a*','*b*','*c*'}. To make the graph clear, we draw those edges between $s_0$, $s_{-1}$, and $s_{-2}$ separately on the left part of the figure.

For $v \in \{0, \ldots, m-1\}$,

$$\delta^1(v,\sigma) = \begin{cases} \delta(v,\sigma), & \delta(v,\sigma) > 0; \\ \tau_\sigma, & \delta(v,\sigma) = 0, \end{cases} \quad (22)$$

where $\tau_\sigma < 0$ is the number so that $s_{\tau_\sigma}$ is labelled with $\sigma$.

The edge from $s_v$ to $s_{\delta^1(v,\sigma)}$ has an *associated probability* that is defined as follows. For $v = 0$, it is $p_\sigma$. For $v \in \{1, \ldots, m-1\}$, it is $p_{A_v,\sigma}$. For $v \in \Sigma \setminus \{A_1\}$, it is $p_{v,\sigma}$.

The constructed markov chain is denoted by $\mathsf{Mc1}(A)$. The following lemma follows from the definition of $\mathsf{Mc1}(A)$.

*Lemma 7:* $\mathsf{Mc1}(A)$ *simulates the process of searching pattern A in the Markov dependent experiments* $Z_1 Z_2 Z_3 \cdots$, *and the stopping time of* $\mathsf{Mc1}(A)$ *is the waiting time of A.*

Let $\mathbf{Q}^1$ be obtained from the transition matrix of $\mathsf{Mc1}(A)$ by deleting the last column and last row. Our key result is:

*Lemma 8: Given* $\mathbf{c} = (c_{-(n-1)}, \ldots, c_{-1}, c_0, \ldots, c_{m-1})^\top \in \mathbb{R}^{m+n-1}$, *we can solve* $\mathbf{x} = \mathbf{Q}^1\mathbf{x} + \mathbf{c}$ *in* $O(n^3 + m)$ *time.*

Assume $(x_{-(n-1)}, \ldots, x_{-1}, x_0, \ldots, x_{m-1})^\top$ is the solution to $\mathbf{x} = \mathbf{Q}^1\mathbf{x} + \mathbf{c}$. Denote $y_i = x_1 - x_i$ for any $i \in \{-(n-1), \ldots, -1\} \cup \{1, \ldots, m\}$. Our algorithm for computing the solution $\mathbf{x}$ goes through three steps as follows.

### 1) STEP 1: COMPUTE $y_{-(n-1)}, \ldots, y_{-1}$

The first $n-1$ equations in $\mathbf{x} = \mathbf{Q}^1\mathbf{x} + \mathbf{c}$ are as follows.

$$x_i = Q^1_{i,1}x_1 + \sum_{j<0} Q^1_{i,j}x_j + c_i. \quad (-(n-1) \le i \le -1) \quad (23)$$

Using $y_i = x_1 - x_i$, the above equations imply that

$$y_i = \sum_{j<0} Q^1_{i,j}y_j - c_i. \quad (-(n-1) \le i \le -1) \quad (24)$$

The linear system (24) contains $n-1$ unknowns and $n-1$ equations. By solving (24) using a standard method, which takes less than $O(n^3)$ time, we obtain $y_{-1}, \ldots, y_{-(n-1)}$.

(Note: According to basic theory of Markov chain, (24) is nonsingular if any one of $s_{-(n-1)}, \ldots, s_{-1}$ can reach $s_1$.)

### 2) STEP 2: COMPUTE $y_1, \ldots, y_m$

Applying the last $m-1$ equations in $\mathbf{x} = \mathbf{Q}^1\mathbf{x} + \mathbf{c}$,

$$x_i = \sum_\sigma p_{A_i,\sigma} \cdot x_{\delta^1(i,\sigma)} + c_i. \quad (1 \le i < m) \quad (25)$$

Therefore,

$$y_i = x_1 - x_i = \sum_\sigma p_{A_i,\sigma} \cdot y_{\delta^1(i,\sigma)} - c_i. \quad (1 \le i < m) \quad (26)$$

Denote $z_i = \sum_{\sigma \ne A_{i+1}} p_{A_i,\sigma} \cdot y_{\delta^1(i,\sigma)}$ and we obtain

$$y_i = z_i + p_{A_i,A_{i+1}} \cdot y_{i+1} - c_i. \quad (1 \le i < m) \quad (27)$$

Thus we obtain a linear system on $m-1$ unknowns $y_2, \ldots, y_m$, with $m-1$ equations. (Note 1: Although $y_1$ appears in this system, it is only a dummy variable and equals 0.) (Note 2: $y_{-(n-1)}, \ldots, y_{-1}$ may appear in this system, but they have already been computed in Step 1.) The remaining question is how do we solve (27) efficiently?

As in the previous sections, we need a recursive formula of $z_i$ that facilitates the computation of $z_i$ in $O(1)$ time. This formula is given in (28) below. It is inevitably more involved than (5) as $\mathsf{Mc1}(A)$ is more complicated than $\mathsf{Mc}(A)$. Denote

$$\delta^1_i = \delta^1(\pi_i, A_{i+1}), \quad (1 \le i < m)$$
$$\gamma_\rho = \sum_{\sigma \ne A_1} p_{\rho,\sigma} y_{\tau_\sigma}. \quad (\rho \in \Sigma)$$

For $1 \le i < m$, we claim that $z_i =$

$$\begin{cases} z_{\pi_i} + p_{A_{\pi_i},A_{\pi_i+1}} y_{\pi_i+1} - p_{A_i,A_{i+1}} y_{\delta^1_i}, & \pi_i > 0; \\ \gamma_{A_i} - p_{A_i,A_{i+1}} y_{j_{A_{i+1}}}, & \pi_i = 0 \text{ and } A_{i+1} \ne A_1; \\ \gamma_{A_i}, & \pi_i = 0 \text{ and } A_{i+1} = A_1. \end{cases} \quad (28)$$

*Proof:* If $\pi_i > 0$, we have

$$\begin{aligned} z_i &= \sum_{\sigma \ne A_{i+1}} p_{A_i,\sigma} \cdot y_{\delta^1(i,\sigma)} = \sum_{\sigma \ne A_{i+1}} p_{A_{\pi_i},\sigma} \cdot y_{\delta^1(\pi_i,\sigma)} \\ &\qquad \text{(because } A_i = A_{\pi_i} \text{ and } \delta^1(i,\sigma) = \delta^1(\pi_i,\sigma)) \\ &= \sum_\sigma p_{A_{\pi_i},\sigma} \cdot y_{\delta^1(\pi_i,\sigma)} - p_{A_{\pi_i},A_{i+1}} \cdot y_{\delta^1_i} \\ &= \sum_{\sigma \ne A_{\pi_i+1}} p_{A_{\pi_i},\sigma} \cdot y_{\delta^1(\pi_i,\sigma)} + p_{A_{\pi_i},A_{\pi_i+1}} \cdot y_{\pi_i+1} \\ &\qquad - p_{A_{\pi_i},A_{i+1}} \cdot y_{\delta^1_i} \\ &= z_{\pi_i} + p_{A_{\pi_i},A_{\pi_i+1}} \cdot y_{\pi_i+1} - p_{A_i,A_{i+1}} \cdot y_{\delta^1_i}. \end{aligned}$$

If $\pi_i = 0$ and $A_{i+1} \ne A_1$, we have

$$\begin{aligned} z_i &= \sum_{\sigma \ne A_{i+1}} p_{A_i,\sigma} \cdot y_{\delta^1(i,\sigma)} \\ &= \sum_{\sigma \ne A_{i+1}, \sigma \ne A_1} p_{A_i,\sigma} \cdot y_{\tau_\sigma} + p_{A_i,A_1} \cdot y_1 \\ &= \sum_{\sigma \ne A_{i+1}, \sigma \ne A_1} p_{A_i,\sigma} \cdot y_{\tau_\sigma} \\ &= \sum_{\sigma \ne A_1} p_{A_i,\sigma} \cdot y_{\tau_\sigma} - p_{A_i,A_{i+1}} \cdot y_{j_{A_{i+1}}} \\ &= \gamma_{A_i} - p_{A_i,A_{i+1}} \cdot y_{j_{A_{i+1}}}. \end{aligned}$$

If $\pi_i = 0$ and $A_{i+1} = A_1$, we have

$$\begin{aligned} z_i &= \sum_{\sigma \ne A_{i+1}} p_{A_i,\sigma} \cdot y_{\delta^1(i,\sigma)} = \sum_{\sigma \ne A_{i+1}} p_{A_i,\sigma} \cdot y_{\tau_\sigma} \\ &= \sum_{\sigma \ne A_1} p_{A_i,\sigma} \cdot y_{\tau_\sigma} = \gamma_{A_i}. \end{aligned}$$

□

### 3) STEP 3: COMPUTE $x_{-(n-1)}, \ldots, x_{-1}, x_1, \ldots, x_m$ AND $x_0$

Because $y_m = x_1 - x_m$ and $x_m = 0$ (dummy variable), we see $x_1 = y_m$, and $x_1$ can thus be computed easily. Next, we compute $x_i$ for $i \notin \{0, 1\}$. Because $y_i = x_1 - x_i$, we get $x_i = x_1 - y_i$. Thus, $x_i$ can be computed from $x_1, y_i$.

When all the $x_i$'s except $x_0$ are known, the last unknown $x_0$ can be computed in $O(m)$ time using the equation $x_0 = \sum_i Q^1_{0,i}x_i + c_0$ which is in linear system $\mathbf{x} = \mathbf{Q}^1\mathbf{x} + \mathbf{c}$.

Next, we prove Lemma 8 by summing up the three steps.

*Proof of Lemma 8:* Array $\gamma$ can be computed in $O(n^2)$ time directly according to its definition. Array $\delta^1_1, \ldots, \delta^1_{m-1}$ can be computed in $O(m)$ time. (Clearly, $\delta^1(i, \sigma)$ can be computed from $\delta(i, \sigma)$ in O(1) time due to (22). Moreover, $\delta^1_i = \delta^1(\pi_i, A_{i+1})$ and $\delta_i = \delta(\pi_i, A_{i+1})$. Therefore $\delta^1_1, \ldots, \delta^1_{m-1}$ can be computed from $\delta_1, \ldots, \delta_{m-1}$ in $O(m)$ time. ) After preprocessing $\pi, \delta^1$, and $\gamma$, we can solve (27) in $O(m)$ time easily based on (28). In addition, Step 1 takes $O(n^3)$ time, whereas Step 3 takes $O(n + m)$ time. Altogether, we solve $\mathbf{x} = \mathbf{Q}^1\mathbf{x} + \mathbf{c}$ in $O(n^3 + n^2 + n + m) = O(n^3 + m)$ time. □

As a corollary of Lemma 8, we have

*Corollary 2:* We can compute the fundamental matrix $(\mathbf{I} - \mathbf{Q}^1)^{-1}$ of $\mathsf{Mc1}(A)$ in $O((n^3 + m)(m + n))$ time.

In the rest part of this subsection, we discuss how to compute the $k$-th moment of the stopping time of $\mathsf{Mc1}(A)$.

The lemma below is almost the same as Lemma 5.

For $-(n-1) \le i \le m$ and $k \ge 0$, define

$$e_i^{(k)} = \sum_{t \ge 0} t^k \cdot \Pr(\text{It takes } t \text{ steps to reach } s_m \text{ from } s_i),$$

$$c_i^{(k)} = -\sum_{l=0}^{k-1} \binom{k}{l} (-1)^{k-l} e_i^{(l)}. \quad (\text{note that } l < k)$$

*Lemma 9: For $k \ge 0$, $(e^{(k)}_{-(n-1)}, \ldots, e^{(k)}_{m-1})^\top$ is the solution to $\mathbf{x} = \mathbf{Q}^1\mathbf{x} + \mathbf{c}$ fixing $\mathbf{c} = (c^{(k)}_{-(n-1)}, \ldots, c^{(k)}_{m-1})^\top$. In other words, the following equations holds for every i,*

$$e_i^{(k)} = \left(\sum_\sigma p_\sigma \cdot e^{(k)}_{\delta^1(i,\sigma)}\right) - \sum_{l=0}^{k-1} \binom{k}{l} (-1)^{k-l} e_i^{(l)}. \quad (29)$$

The proof of Lemma 9 is almost the same as the proof of Lemma 5 and is omitted.

According to Lemma 9, we can compute $e^{(1)}, e^{(2)}, \ldots, e^{(k)}$ in sequence. It takes $O((m+n)k)$ time for computing $c^{(k)}$, and $O(n^3 + m)$ time to compute $e^{(k)}$ from $c^{(k)}$. Therefore, the total running time is $O((mk + nk + n^3)k)$.

*Corollary 3:* For the first-order Markov dependent experiments $Z_1Z_2Z_3\cdots$, the $k$-th moment of the waiting time of $A$ can be computed in $O(k^2m + (k^2n + kn^3))$ time.

*Remark 2:* 1. Typically, $m$ will be very large compared to $n, k$. We may assume that $n^3 < m$ and $k = O(1)$. In this case, the running time mentioned above is only $O(m)$.

2. In fact, our method can be generalized to the case of $j$-th order Markov dependent experiments. When $j, k, n$ are regarded as constants, our running time is still $O(m)$.

3. The terms $n^3$ appeared above (which come from the computation of matrix inversion) are not best possible. There exist $o(n^3)$ (say, $O(n^{2.3727})$) time algorithms for computing matrix inversion and multiplication nowadays [31, p. 828].

### D. APPLICATION IN PENNY's GAME PROBLEM

*Penney's game* [1], [19], [20], [39]. Given $t$ strings $A^{(1)}, \ldots, A^{(t)}$ which are *reduced*, i.e., no one is a substring of another. Denote by $\mathsf{L}_1, \ldots, \mathsf{L}_t$ the waiting times of $A^{(1)}, \ldots, A^{(t)}$ respectively in repeated experiments $Z_1, Z_2 \ldots$ (iid or Markov model). Let $\mathsf{L} = \min(\mathsf{L}_1, \ldots, \mathsf{L}_t)$, and $w_j = \Pr(\mathsf{L} = \mathsf{L}_j)$ for $1 \le j \le t$. How can we compute the expected waiting time $\mathbb{E}(\mathsf{L})$ until one of the $t$ strings appear, and the probability $w_1, \ldots w_t$ for each given string to be the first to appear?

A clever method for solving the Penney's problem is given in [19], [20] and is based on the following lemma.

*Lemma 10 [20]: Given a homogeneous first-order Markov chain $C^*$ with all states $s_0, \ldots, s_t$ communicate. Let $\mathsf{L}_{i,j}$ be the first passage time of $s_j$ starting at $s_i$, and $e_{i,j} = \mathbb{E}\,\mathsf{L}_{i,j}$. For any $t$ distinct states $s_{b_1}, \ldots, s_{b_t}$, the following equation holds and its coefficient matrix is nonsingular.*

$$\begin{pmatrix} 0 & 1 & \ldots & 1 \\ 1 & e_{b_1,b_1} & \ldots & e_{b_t,b_1} \\ \vdots & \vdots & & \vdots \\ 1 & e_{b_1,b_t} & \ldots & e_{b_t,b_t} \end{pmatrix} \begin{pmatrix} \mathbb{E}\,\mathsf{L}^* \\ w_1^* \\ \vdots \\ w_t^* \end{pmatrix} = \begin{pmatrix} 1 \\ e_{0,b_1} \\ \vdots \\ e_{0,b_t} \end{pmatrix}, \quad (30)$$

*where $\mathsf{L}^* = \min\{\mathsf{L}_{0,b_1}, \ldots, \mathsf{L}_{0,b_t}\}$ and $w_j^* = \Pr(\mathsf{L} = \mathsf{L}_{0,b_j})$.*

We can apply Lemma 10 as follows to solve the Penney's game. Construct a chain $C^*$ so that each string $A^{(j)}$ corresponds to a state $s_{b_j}$ in $C^*$ and that $(\mathbb{E}\,\mathsf{L}, w_1, \ldots, w_t) = (\mathbb{E}\,\mathsf{L}^*, w_1^*, \ldots, w_t^*)$. The construction is similar to the construction of an automaton for searching a compound pattern, but each transient edge is now associated with a probability (we omit the details). It is unnecessary to construct the chain explicitly. Applying Lemma 10, Penney's problem now reduces to solving (30). Moreover, solving (30) is straightforward, when the coefficients $\{e_{b_j,b_i}\}, \{e_{0,b_i}\}$ are known.

On the other hand, *our algorithms in II-B can be applied to compute the coefficients of* (30). Denote by $\mathsf{Mc}_i^*$ the Markov chain searching $A^{(i)}$ ($\mathsf{Mc}_i^* = \mathsf{Mc}(A^{(i)})$ for iid model and $\mathsf{Mc}_i^* = \mathsf{Mc1}(A^{(i)})$ for Markov model). First, for each $i$, compute the expected stopping time of $\mathsf{Mc}_i^*$ starting from each state (this can be done by our algorithms). Then, $e_{b_j,b_i}$ equals the expected stopping time of $\mathsf{Mc}_i^*$ starting from $s_h$, where $h$ is the largest element such that $A^{(i)}(1, h)$ is a suffix of $A^{(j)}$ (holds for iid model; Markov model is similar).

*Remark 3:* 1. For the compound pattern case, the second moment of the waiting time $\mathsf{L}$ is also studied. Combining the martingale approach proposed in [19], [20] with a technique called *gambling teams*, [21] and [23] showed how to compute the second moment under iid and Markov models. Higher moments are not studied to the best of our knowledge.

2. Conway gave a simple formula for the 2-players Penney's game. References [16], [40], [41] studied strategies about the game.

## V. CONCLUSION

To sum up, the $k$-th moment of waiting time of a single pattern $A$ in random experiments and some closely related statistics (such as the $k$-th moment of the number of comparisons spent by the KMP algorithm) can be computed efficiently via the Markov embedding approach.

Our results are based on a fast solver of the linear system (1) related to the Markov chain corresponding to $A$.

Our technique extends to the Markov model, and our algorithm computes the $k$-th moment starting from every state, not only from the initial state.

Applications of our algorithm include the computation of the coefficients of (30), which leads to the solution of the Penney's game, as shown in subsection IV-D. The Penney's game has found applications in networks [42]. Other applications are mainly in bioinformatics; see some details in [12], [23].

## APPENDIX A
## OMITTED PROOFS

Claim: *The number of comparisons* $\mathsf{K}$ *equals the total length of edges traveled by the Markov chain.*

*Proof:* (In this proof, we need some preliminaries in I-B.)

It reduces to proving that the length $\ell(i,\sigma)$ of the edge from $s_i$ $(i < m)$ to $s_{\delta(i,\sigma)}$ is defined as the number of comparisons consumed by Algorithm 1 with input $s_i$ and $\sigma$.

Recall the following well-known property of $\pi$:

$$\{j \in \{0,\ldots,i\} \mid A(1,j) \text{ is a prefix of } A(1,i)\} = \{i, \pi(i), \pi(\pi(i)), \ldots, 0\} := \{j_0, j_1, \ldots, j_r\} := J.$$

Algorithm 1 compares $A[j_0+1], A[j_1+1], \ldots$ in sequence with $\sigma$, until it finds $A[j_h+1] = \sigma$ or find that no such $j_h$ exists. For each $j$ in $J$ that is at least $\delta(i,\sigma)-1$, it consumes one comparison. On the other hand, the edge length $\ell(i,\sigma)$ is also the number of $j$'s in $J$ that is at least $\delta(i,\sigma)-1$. □

The proof of (17) is attached here.

$$\sum_{h=l}^{k}\binom{k}{h}\binom{h}{l}(-1)^h = \binom{k}{l}\sum_{h=l}^{k}\binom{k-l}{h-l}(-1)^h = \binom{k}{l}\sum_{h=0}^{k-l}\binom{k-l}{h}(-1)^h = \binom{k}{l}(1-1)^{k-l} = 0.$$

## APPENDIX B
## MORE ABOUT FUNDAMENTAL MATRIX

*Theorem 1 [29], [38]: Let* $\mathbf{N}_{dg}$ *denote the diagonal matrix with the same diagonal as* $\mathbf{N}$*, and* $\mathbf{N}_{sq}$ *denote the Hadamard product of* $\mathbf{N}$ *with itself (i.e. each entry of* $\mathbf{N}$ *is squared). Let* $\mathbf{1}$ *denote the length-m column vector whose entries are all* 1.

Expected number of visits to each state.

*Entry* $N_{i,j}$ *is the expected number of visits to state* $s_j$ *(before being absorbed) if it is started in state* $s_i$.

Variance of number of visits to each state.

*Let* $\mathbf{V} = (V_{i,j})$*, where* $V_{i,j}$ *denotes the variance of the number of visits to state* $s_j$ *(before being absorbed) starting from state* $s_i$*. We have* $\mathbf{V} = \mathbf{N}(2\,\mathbf{N}_{dg} - \mathbf{I}) - \mathbf{N}_{sq}$.

Expected number of visited states.

*Let* $\mu_i$ *be the expected number of visited states (before being absorbed) starting from state* $s_i$*. Then* $\mu = \mathbf{N}\,\mathbf{N}_{dg}\,\mathbf{1}$.

Transient probabilities.

*Let* $\mathbf{H} = (H_{i,j})$*, where* $H_{i,j}$ *denotes the probability of visiting* $s_j$ *starting from state* $s_i$*. Then,* $\mathbf{H} = (\mathbf{N} - \mathbf{I})\,\mathbf{N}_{dg}^{-1}$.

## ACKNOWLEDGMENT

The authors would like to thank Prof. Andrew Yao, Zhiyi Huang, and Yifei Jin for joining some discussions and for unselfish help. This article was presented at the 27th International Symposium on Algorithms and Computation (ISAAC'16) [DOI = 10.4230/LIPIcs.ISAAC.2016.39].

## REFERENCES

[1] W. Penney, "Problem 95: Penney-ante," *J. Recreational Math.*, vol. 2, no. 4, p. 241, Oct. 1969.
[2] A. D. Solov'ev, "A combinatorial identity and its application to the problem concerning the first occurrence of a rare event," *Theory Probab. Appl.*, vol. 11, no. 2, pp. 276–282, Jan. 1966.
[3] R. L. Graham, D. E. Knuth, and O. Patashnik, *Concrete Mathematics: A Foundation for Computer Science*, 2nd ed. Boston, MA, USA: Addison-Wesley, 1994.
[4] M. Lothaire, *Application Combinatorics Words*. Cambridge, U.K.: Cambridge Univ. Press, 2005.
[5] S. I. Hakak, A. Kamsin, P. Shivakumara, G. A. Gilkar, W. Z. Khan, and M. Imran, "Exact string matching algorithms: Survey, issues, and future research directions," *IEEE Access*, vol. 7, pp. 69614–69637, 2019.
[6] D. Barbará and T. Imieliński, "Sleepers and workaholics: Caching strategies in mobile environments (extended version)," *VLDB J.*, vol. 4, no. 4, pp. 567–602, Oct. 1995.
[7] J. C. Fu, "Reliability of consecutive-k-out-of-n: F systems with (k-1)-step Markov dependence," *IEEE Trans. Rel.*, vol. R-35, no. 5, pp. 602–606, Dec. 1986.
[8] G. Reinert, S. Schbath, and M. S. Waterman, "Probabilistic and statistical properties of words: An overview," *J. Comput. Biol.*, vol. 7, nos. 1–2, pp. 1–46, Feb. 2000.
[9] J. I. Naus and K.-N. Sheng, "Matching among multiple random sequences," *Bull. Math. Biol.*, vol. 59, no. 3, pp. 483–496, May 1997.
[10] M. Régnier, "A unified approach to word occurrence probabilities," *Discrete Appl. Math.*, vol. 104, nos. 1–3, pp. 259–280, Aug. 2000.
[11] S. Karlin and V. Brendel, "Chance and statistical significance in protein and DNA sequence analysis," *Science*, vol. 257, no. 5066, pp. 39–49, Jul. 1992.
[12] M. E. Lladser, M. D. Betterton, and R. Knight, "Multiple pattern matching: A Markov chain approach," *J. Math. Biol.*, vol. 56, nos. 1–2, pp. 51–92, Nov. 2007.
[13] S. Borah, D. Bhattacharjee, K. Vijay, K. Singh, and B. Rai, "Multithreaded kmp: A proposed dna sequencing algorithm," *Int. J. Adv. Sci., Eng. Technol.*, vol. 3, pp. 36–39, Oct. 2015.
[14] L. Dudas, "Improved pattern matching to find DNA patterns," in *Proc. IEEE Int. Conf. Autom., Qual. Test., Robot.*, vol. 2, May 2006, pp. 345–349.
[15] P. Neamatollahi, M. Hadi, and M. Naghibzadeh, "Simple and efficient pattern matching algorithms for biological sequences," *IEEE Access*, vol. 8, pp. 23838–23846, 2020.
[16] L. J. Guibas and A. M. Odlyzko, "String overlaps, pattern matching, and nontransitive games," *J. Combinat. Theory A*, vol. 30, no. 2, pp. 183–208, 1981.
[17] S. Breen, M. S. Waterman, and N. Zhang, "Renewal theory for several patterns," *J. Appl. Probab.*, vol. 22, no. 1, pp. 228–234, Mar. 1985.
[18] W. Feller, *An Introduction to Probability Theory and its Applications*, vol. 1. Hoboken, NJ, USA: Wiley, 1968.
[19] S.-Y.-R. Li, "A martingale approach to the study of occurrence of sequence patterns in repeated experiments," *Ann. Probab.*, vol. 8, no. 6, pp. 1171–1176, Dec. 1980.
[20] H. U. Gerber and S.-Y.-R. Li, "The occurrence of sequence patterns in repeated experiments and hitting times in a Markov chain," *Stochastic Processes Appl.*, vol. 11, no. 1, pp. 101–108, Mar. 1981.
[21] V. Pozdnyakov, J. Glaz, M. Kulldorff, and J. M. Steele, "A martingale approach to scan statistics," *Ann. Inst. Stat. Math.*, vol. 57, no. 1, pp. 21–37, Mar. 2005.

[22] V. Pozdnyakov and M. Kulldorff, "Waiting times for patterns and a method of gambling teams," *Amer. Math. Monthly*, vol. 113, no. 2, pp. 134–143, Feb. 2006.
[23] V. Pozdnyakov, "On occurrence of patterns in Markov chains: Method of gambling teams," *Statist. Probab. Lett.*, vol. 78, no. 16, pp. 2762–2767, Nov. 2008.
[24] R. J. Gava and D. Salotti, "Stopping probabilities for patterns in Markov chains," *J. Appl. Probab.*, vol. 51, no. 1, pp. 287–292, Mar. 2014.
[25] J. C. Fu and Y. M. Chang, "On probability generating functions for waiting time distributions of compound patterns in a sequence of multistate trials," *J. Appl. Probab.*, vol. 39, no. 1, pp. 70–80, Mar. 2002.
[26] J. C. Fu and M. V. Koutras, "Distribution theory of runs: A Markov chain approach," *J. Amer. Stat. Assoc.*, vol. 89, no. 427, pp. 1050–1058, Sep. 1994.
[27] N. Balakrishnan and M. V. Koutras, *Runs Scans with Application*. Hoboken, NJ, USA: Wiley, 2002.
[28] J. C. Fu and W. Y. W. Lou, *Distribution Theory of Runs and Patterns and its Applications*. Singapore: World Scientific, 2003.
[29] M. Iosifescu, *Finite Markov Processes Their Application* (Dover Books on Mathematics). New York, NY, USA: Dover, 2014.
[30] D. E. Knuth, J. H. Morris, Jr., and V. R. Pratt, "Fast pattern matching in strings," *SIAM J. Comput.*, vol. 6, no. 2, pp. 323–350, Jul. 1977.
[31] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA, USA: MIT Press, 2009.
[32] K. Jin, "Computing the pattern waiting time: A revisit of the intuitive approach," in *Proc. ISAAC*, vol. 64, S.-H. Hong, Ed. Wadern, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016, pp. 1–12.
[33] I. Fudos, E. Pitoura, and W. Szpankowski, "On pattern occurrences in a random text," *Inf. Process. Lett.*, vol. 57, no. 6, pp. 307–312, Mar. 1996.
[34] O. Chrysaphinou and S. Papastavridis, "The occurrence of sequence patterns in repeated dependent experiments," *Theory Probab. Appl.*, vol. 35, no. 1, pp. 145–152, Jan. 1991.
[35] J. A. D. Aston and D. E. K. Martin, "Waiting time distributions of competing patterns in higher-order Markovian sequences," *J. Appl. Probab.*, vol. 42, no. 4, pp. 977–988, Dec. 2005.
[36] M. Régnier and W. Szpankowski, "On pattern frequency occurrences in a Markovian sequence," *Algorithmica*, vol. 22, no. 4, pp. 631–649, Dec. 1998.
[37] M. L. Fredman, J. Komlós, and E. Szemerédi, "Storing a sparse table with 0(1) worst case access time," *J. ACM*, vol. 31, no. 3, pp. 538–544, Jun. 1984.
[38] C. M. Grinstead and J. L. Snell, *Introduction to Probability*. Providence, RI, USA: AMS, 2003.
[39] Anonymous. (2016). *Penney's Game*. [Online]. Available: https://en.wikipedia.org/wiki/Penney%27s_game
[40] M. Gardner, "On the paradoxical situations that arise from nontransitive relations," *Sci. Amer.*, vol. 231, no. 4, pp. 120–125, 1974.
[41] J. A. Csirik, "Optimal strategy for the first player in the penney ante game," *Combinatorics, Probab. Comput.*, vol. 1, no. 4, pp. 311–321, Dec. 1992.
[42] A. Baldi, G. Cencetti, D. Fanelli, and F. Bagnoli, "Intransitiveness in games and random walks," in *Internet Science*, S. El Yacoubi, F. Bagnoli, and G. Pacini, Eds. Cham, Switzerland: Springer, 2019, pp. 204–216.

**DANNA ZHANG** was born in Hengyang, Hunan, China, in 1998. She received the B.S. degree in internet of things from Dalian Maritime University, Dalian, China. She is currently pursuing the M.S. degree in theoretical computer science with Sun Yat-sen University, Shenzhen, Guangdong, China, under the supervision of Prof. Jin. Her main research interests include algorithm design and computational geometry.

**KAI JIN** was born in Changsha, Hunan, China, in 1986. He received the B.S. and Ph.D. degrees in computer science and technology from Tsinghua University, Beijing, China, in 2008 and 2016, respectively.

From 2016 to 2018, he held a postdoctoral position with the Department of Computer Science, The University of Hong Kong, and the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, from 2018 to 2020. Since 2020, he joined the School of Intelligent Systems Engineering, Sun Yat-sen University, Shenzhen, Guangdong, China, as an Associated Professor and a member of Hundred Talents Program. His research interests include algorithm design, combinatorics, game theory, and computational geometry. See his publications on personal website https://cscjjk.github.io.

Dr. Jin was a Program Committee Member of the 20th Conference in Autonomous Agents and Multiagent System (AAMAS'21). He is a recipient of the National Natural Science Foundation of China (Grant 62002394) for the term 2021–2023.