Efficient Formation of Auxiliary Markov Chains through Determining Rules for Equivalent States

Donald E.K. Martin
Department of Statistics, North Carolina State University, 5116 SAS Hall, Raleigh, NC 27695

Abstract

When using an auxiliary Markov chain (AMC) to compute sampling distributions, the computation complexity is directly related to the number of Markov chain states. For certain complex pattern statistics, minimal deterministic finite automata (DFA) have been used to facilitate efficient computation by reducing the number of AMC states to a minimal set sufficient for computing the distribution. However, it can be the case that forming a DFA before applying a minimization algorithm is computationally expensive. Examples include situations where statistic updates for overlapping pattern occurrences are different than for non-overlapping occurrences. To deal with these situations, we give a characterization of equivalent states so that extra ones (and their offspring) are deleted during the process of state space formation. The method is illustrated on computing the distribution of coverage of spaced seeds. The computational speed-ups are highlighted.

Key Words: Active proper suffix, auxiliary Markov chain, computational efficiency, minimal deterministic finite automaton, overlapping pattern occurrences, spaced seed coverage

1. Introduction

A method that has proven fruitful for computing distributions associated with patterns and more general statistics is to associate the statistic's distribution with an auxiliary Markov chain (AMC), and then use properties of the Markov chain to compute the desired distributions. Fu and Koutras (1994) forwarded this approach with their finite Markov chain imbedding (FMCI) method, and it has been used by many researchers since then (see, e.g., Koutras and Alexandrou 1995; Ebneshahrashoob et al. 2005; Aston and Martin 2007; Martin and Aston 2008). However, forming a Markov chain requires keeping sufficient information so that the Markov property holds, and thus for certain complex patterns, the state space of the AMC can be prohibitively large.

In recent years, several authors (e.g. Nuel 2008; Lladser et al. 2008; Marshall and Rahmann 2008; Ribeca and Raineri 2008; Martin and Aston 2013) have used the states of a minimal deterministic finite automaton (DFA) as the AMC state space. In that approach, to obtain a minimal DFA, an original AMC is first obtained. This can be a problem, since the original DFA can be prohibitively large, causing excesses in terms of computation time and storage. In this paper, we bypass this problem by using clues from DFA minimization algorithms to delete states in the process of forming the original AMC.

The organization of the paper is as follows. The next section discusses background information on computing a distribution through an AMC, and how minimizing an associated DFA can be helpful in this regard. Section 3 has a characterization of equivalent

states that allows states to be eliminated during the process of forming the state space. In Section 4, this characterization is applied to setting up the state space of an AMC for computing the distribution of coverage of a spaced seed. The reduction in the size of the state space and computation time is highlighted. The final section is a summary.

2. Auxiliary Markov Chains and Minimal DFA

Consider an m th-order Markovian sequence $\mathbf{X} = X_1,\ldots,X_n$ with observed values $\mathbf{x} = x_1,\ldots,x_n$, where each x_i lies in a state space Σ. The problem under consideration is computing the distribution of a statistic Z of a collection of patterns in $\mathbf{X}$.

2.1 Computing Distributions of Pattern Statistics through Auxiliary Markov Chains

A *pattern* (also called a *word* or *string*) is a sequence of symbols from a *state space* (or *alphabet*) Σ. For notation purposes, the length of a pattern u is denoted by $|u|$ ($|B|$ is also used to denote the number of elements in a set B, with the meaning hopefully being clear from the context). The concatenation of pattern v to the right of pattern u is denoted by $u \cdot v$. The *suffix* and *prefix* of length $d \le |u|$ of pattern $u = u_1,\ldots,u_{|u|}$ are respectively given by $(u)_d \equiv u_{|u|-d+1},\ldots,u_{|u|}$ and ${}_d(u) \equiv u_1,\ldots,u_d$. If $d < |u|$, $(u)_d$ and ${}_d(u)$ are respectively a *proper suffix* and *proper prefix*.

If one can form an auxiliary Markov chain $\{A_t\}$ with state space Q such that for z in the range Υ_Z of statistic Z, $P(Z=z) = P(A_n \in Q_z)$ (Q_z is a partition of the states of Q), then basic properties of Markov chains may be used to compute the distribution of Z. In an efficient approach that does not duplicate pattern prefixes for the various values of the statistic, Aston and Martin (2007) used probability matrices Ψ_t, $t = m, m+1,\ldots,n$ to hold probabilities for the AMC lying in the various states of its state space, where the subscript t indicates the time point. The matrices each have rows that correspond to the possible values of the statistic, and columns that hold probabilities of the AMC lying in its various states. (See also Koutras and Alexandrou 1995, where probability vectors are used for each of the possible values of the statistic of interest.) In the approach of Aston and Martin (2007), to update Ψ_t from time t to $t+1$, after multiplying Ψ_t by the transition probability matrix Ω for the AMC states, probabilities for transitions into states where the statistic's count is incremented are moved to the appropriate row. Obviously, it is important for computation purposes that the size of the AMC state space is as small as possible. Minimal DFA can be helpful in this regard.

2.2 AMC state minimization through DFA

Formally, a DFA D is a 5-tuple $D = (Q, \Sigma, \delta, q_0, F)$, where Q is a set of states, Σ is the alphabet, δ is the transition function for states $q \in Q$, i.e. $\delta : Q \times \Sigma \to Q$, $q_0 \in Q$ is a start state, and $F \subset Q$ is a set of "final" or "accepting" states. For an input sequence $(x_1, x_2,\ldots,x_n)$ ($x_i \in \Sigma$), the DFA begins in state q_0 and transitions according to δ as symbols x_i are fed in. If $q_j \in F$ for some $j \in (1,\ldots,n)$, then the string $x_1,\ldots,x_j$ is "accepted," otherwise it is "rejected." The *language* of a DFA is the set of strings that it

accepts. When a state of F is entered, an action is taken. For example, for a statistic of a collection of patterns, F consists of the patterns, and the statistic is incremented when a state of F is entered.

Useful for our purposes is that adding probabilities to the transitions of a DFA turns it into a Markov chain. In fact, a DFA with states corresponding to pattern prefixes (an Aho-Corasick automaton; see Aho and Corasick 1975) can be modified to obtain an mth-order Markovian AMC by adding to the pattern prefixes any m-tuples that are not prefixes, deleting all strings of length less than m, and then initializing the computation at time m. Using DFA theory helps because we can apply a DFA minimization algorithm to reduce the state space.

DFA states q and $q*$ are *equivalent* if beginning in them and entering an arbitrary string, the result is either a final state in both cases or a non-final state in both cases. A well-known result from computer science that is useful in our framework is that beginning with any DFA, one can find an equivalent DFA that recognizes the same language and has a minimal number of states (see, e.g. Hopcroft 1971; Hopcroft et al. 2001).

Minimizing a DFA is a special case of the multi-function "coarsest partition" problem (Tewari et al. 2002) where, given an initial partition $H_1,\ldots,H_r$ of a set Q and functions $f_1,\ldots f_\gamma$ over the states, one finds the partition $G_1,\ldots,G_s$, $s \geq r$ with the smallest number of equivalence classes s such that (i) each G_i is a subset of some class H_j, and (ii) q and $q*$ in G_h implies that $f_a(q)$ and $f_b(q*)$ are both in some class G_k for all a and b. For minimizing a DFA we have $r = 2$, the initial partition is $(H_1, H_2) = (F, Q \setminus F)$, and the resulting partition is the equivalence classes of the DFA, which are the states of the minimal DFA. The minimal DFA has the smallest number of states for any DFA that recognizes the same language, and is also unique, up to a renaming of the states (see, e.g. Hopcroft et al. 2001, pp. 154-162). Given any AMC, a DFA minimization algorithm can be applied to its states/transition function to obtain a minimal version. If no states are eliminated the AMC states are already in minimal form.

To add probabilities to DFA transitions, on symbol x_i, pattern prefix q transitions to the longest suffix of $q \cdot x_i$ that is in Q (Aho and Corasick 1975). The initial distribution over AMC states is given by assigning probabilities for m-tuples of Σ^m according to the initial distribution of $\mathbf{X}$, with initial probabilities for all other states being zero. Transition probabilities of matrix Ω for transitions of AMC states are given by $\Pr(q_{j-1} \to q_j) = \Pr(\tilde{x}_{j-1} \to \tilde{x}_j)$, where $\tilde{x}_r = (q_r)_m$, $r = j-1, j$. The states of the resulting Markov chain can be minimized using an analog of the Hopcroft (1971) algorithm (with the additional restriction that all states in an equivalence class must have the same m-tuple as their suffix), to give an AMC with a minimal number of states for that particular model order. An example of using DFA minimization to form a minimal AMC state space is given next.

Consider computing the distribution of the number of overlapping occurrences of the chi motif of *H. influenza* $F = W_8 = \{GATGGTGG, GCTGGTGG, GGTGGTGG, GTTGGTGG\}$ (Ledent and Robin 2005) in a first-order Markovian sequence $\mathbf{X}$, with $\Sigma = \{A, C, G, T\}$.

In the Aho-Corasick automaton, Q consists of prefixes of the patterns of F (see Figure 1a). The modified DFA with 1-tuples added and the empty string ε deleted is shown in Figure 1b. Final states F and $Q \setminus F$ form the initial state partition. States $W_7 = \{GATGGTG, GCTGGTG, GGTGGTG, GTTGGTG\}$ can enter F on symbol G whereas other states of $Q \setminus F$ cannot, and thus W_7 is split off as a separate class in the minimization process (these states transition the same on other symbols as well, and thus they remain a single class throughout the minimization process). Then states W_6 that enter W_7 on symbol G can be split from those that don't, forming another equivalence class. In the end, states of length longer than two are equivalent if they differ only in their second symbol, and thus four states can be combined in each of the equivalence classes $W_3, \ldots, W_8$ (for length two, states $W_{2a} = GG$ and $W_{2b} = \{GA, GC, GT\}$ are distinguishable because they transition differently on symbols A, C, or G). The final partition for the chi-motif is shown in Figure 1c, with numbered states j representing W_j. Figure 1d shows the result of deleting the final state from Figure 1c, while re-mapping the entering transition. Notice that the number of AMC states has been reduced from 30 to 9.

In the context of computing pattern distributions, equivalent states q and q' must have exactly the same probabilities and statistic updates when concatenating an arbitrary string. For the example above, this would hold for the equivalence classes as defined, as long as the order of Markovian dependence $m \leq 2$.

3. Minimal AMC through active proper suffixes and completion strings

Let W be a collection of patterns, and set $Q = \Sigma^m \cup \tilde{Q}$, where Σ^m are the m-tuples that are needed in the state space due to the Markov assumption, and $\tilde{Q}$ is the set of the proper prefixes of the patterns of W. $\tilde{Q}$ admits the decomposition $\tilde{Q} = \tilde{Q}_m \cup \tilde{Q}_{>m}$, where $\tilde{Q}_m$ are prefixes of patterns of W of length exactly m, and $\tilde{Q}_{>m}$ are the prefixes of length greater than m. The set $\Sigma^m = \hat{Q}_{nosuf} \cup \hat{Q}_{suf} \cup \tilde{Q}_m$, where $\hat{Q}_{suf}$ consists of the m-tuples that have a proper suffix that is a prefix of a pattern of W (though the string itself is not a pattern prefix), and $\hat{Q}_{nosuf}$ is the set of m-tuples that have no suffix that is a pattern prefix.

A *direct occurrence* associated with $q \in \tilde{Q}$ is the occurrence of a pattern $w^{(j)} \in W$ with $w^{(j)} = q \cdot v_j$. Here v_j is the *completion string* for q to reach pattern $w^{(j)}$. The longest proper suffix of a string q that is a proper prefix of a pattern of W is called the *active proper suffix* of q, and is denoted by $aps(q)$. If $aps(q) \in \tilde{Q}_{>m}$, the *failure state* of q (denoted by $fl(q)$) is defined by $fl(q) \equiv aps(q)$. Otherwise, $fl(q) \equiv (q)_m$.

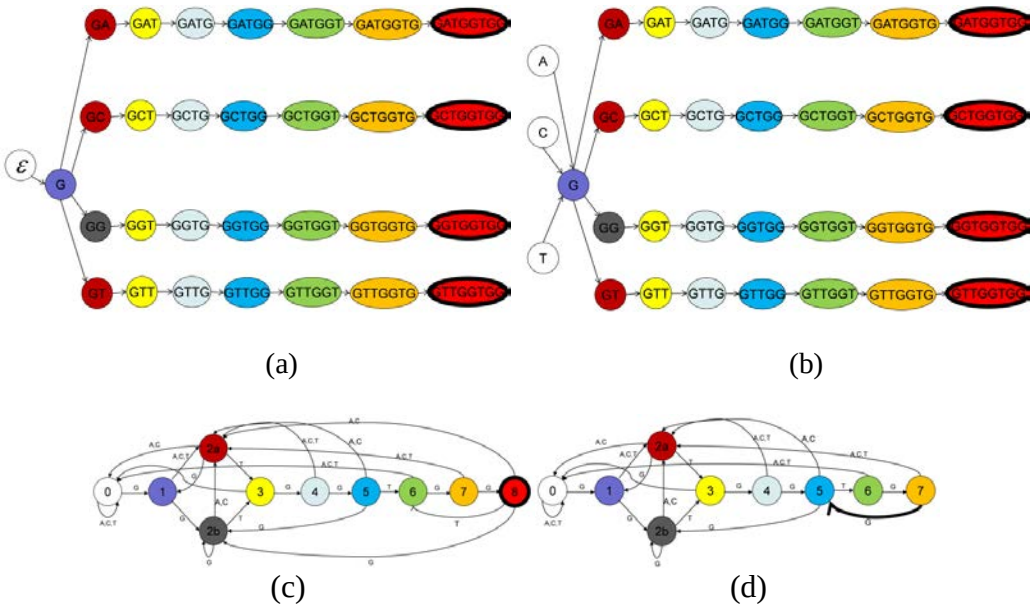

Figure 1. (a) Aho-Corasick DFA for the Chi-motif of *H. influenza*, $\{GATGGTGG, GCTGGTGG, GGTGGTGG, GTTGGTGG\}$; (b) modified version for a Markov chain, with 1-tuples replacing ε; (c) minimized version; (d) minimal version with state representing chi-motif eliminated and its transition (marked with a bold line) re-mapped. In plots (a), (b), and (c), final states are marked with a bold outline. Also note that in plots (a) and (b), some transitions are not shown for clarity).

When the transition of q on symbol x is to state $q \cdot x$, the transition is called a *forward transition.* All other transitions are *failure transitions* that can be executed by first going to $fl(q)$, and then executing the transition on symbol x from there, as with an Aho-Corasick automaton.

To illustrate these concepts, let $\Sigma = \{0,1\}$ and $W = \{1001,101,1101\}$. The set of proper prefixes of patterns is $\tilde{Q} = \{1,10,11,100,110\}$, and if $m = 1$, $\tilde{Q}_1 = \{1\}$, $\tilde{Q}_{>1} = \{10,11,100,110\}$ and $Q = \{0,1,10,11,100,110\}$. For $q_1 = 10 = x_1, x_2$, the set of all completion strings is $\{v_1, v_2\} = \{01,1\}$. The string $x_1, \ldots, x_4 = 1001 = q_1 \cdot v_1$ represents a direct hit of q. With symbol $x = 0$, q_1 transitions to $q_1 \cdot x$, since 100 is a state of Q. On symbol $x = 1$, q_1 transitions to the longest suffix of $q_1 \cdot x = 101$ that is in Q, i.e. to 1. That transition could be executed by first going to $aps(q_1) = 0$, where, on symbol $x = 1$, the transition is to state 1. As 101 is a pattern occurrence, the statistic count would be incremented on the transition. If $m = 2$, $\tilde{Q}_2 = \{10,11\}$, $\hat{Q}_{nosuf} = \{00\}$, $\hat{Q}_{suf} = \{01\}$, and $\tilde{Q}_{>2} = \{100,110\}$, so that $Q = \{00,01,10,11,100,110\}$.

The failure sequence (denoted by fs_q) associated with state $q \in Q$ consists of q, along with its sequence of failure states, i.e.

$$fs_q = \left[fs_q(1), \ldots, fs_q\left(\left|fs_q\right|\right)\right] = \left[q, fl(q), fl\left(fl(q)\right), \ldots, (q)_m \right].$$

(Note that in some cases the AMC state may include more information than pattern prefix q, to facilitate the formation of a Markov chain.) Since $fs_q(j)$ is a suffix of $fs_q(l)$, $l < j \le \left|fs_q\right|$, $fs_q(l) = fs_{q'}(l)$ implies that $fs_q(j) = fs_{q'}(j)$. Setting up a failure sequence may be handled sequentially. If $\left|aps(q)\right| \le m$, then $fs_q = \left(q, (q)_m\right)$. Otherwise, simply place q to the left of the failure sequence of $aps(q)$ to obtain fs_q.

Definition 1. Failure sequences fs_q and $fs_{q'}$ are equivalent (denoted by $fs_q \sim fs_{q'}$) if $\left|fs_q\right| = \left|fs_{q'}\right|$, $(q)_m = (q')_m$, and for $j = 1, \ldots, \left|fs_q\right|$, the corresponding elements of the sequences have the same set of completion strings and the same updates to the statistic's value on all direct occurrences of elements of the failure sequences.

Theorem 1 below indicates that equivalence of states q and q' may be determined through their failure sequences.

Theorem 1. States q and q' are equivalent for the problem of computing the distribution of a pattern statistic if and only if $fs_q \sim fs_{q'}$.

Proof. Let states q and q' have equivalent failure sequences. Then $(q)_m = (q')_m$, and thus probabilities associated with transitions from q and q' after observing an arbitrary string v must be the same, by the Markov property. Since strings $fs_q(j)$ and $fs_{q'}(j)$ have the same set of completion strings, direct occurrences happen for the same strings. The increment to the statistic's value with direct occurrences from elements of the failure sequence are the same, and thus the total increment to the statistic will be the same after concatenating an arbitrary string. The failure sequences contain the only locations where pattern occurrences involving symbols of q and q' can begin, and pattern occurrences that do not contain symbols of q and q' must give the same statistic updates. Thus updates to the statistic after concatenating v are the same, and q and q' are equivalent.

Conversely, let q and q' be equivalent. Then the increments to the statistic and the probabilities associated with observing an arbitrary string v must be the same. Equal probabilities for arbitrary strings implies that $(q)_m = (q')_m$. Now let v be a completion string for q. It must also be a completion string for q' or equivalency is violated, and thus q and q' must have the same set of completion strings. The updates for direct hits of q and q' must be the same, since we can always concatenate symbols such that none of the other elements of fs_q and $fs_{q'}$ are completed, and then the update for direct hits of q and q' are the total update. Now, inductively, let q equivalent to q' imply equal updates for direct occurrences beginning in the corresponding positions of $\left[fs_q(1), \ldots, fs_q(l)\right]$ and

$\left[fs_{q'}(1),\ldots, fs_{q'}(l) \right]$, $l < \left| fs_q \right|$. Then choose a completion string for $fs_q(l+1)$ such that none of the $fs_q(j)$ are completed, $j > l+1$. Then q equivalent to q' implies that the updates when observing the completion strings of $fs_q(l+1)$ and $fs_{q'}(l+1)$ must be the same, whether or not longer string(s) of $fs_q(j)$ and $fs_{q'}(j)$, $j < l+1$ occur first. The result follows inductively. ■

Theorem 1 is key for obtaining a minimal AMC state space in the process of state space formation, as it allows one to show equivalency of states by checking direct hits along their failure sequences. Whereas verifying equivalency can still be a daunting task, in certain situations the theorem can help immensely. This will be illustrated in the next section in the context of computing the distribution of coverage of a spaced seed.

4. Application to coverage of a spaced seed

4.1 Spaced seeds and the coverage distribution

A heuristic method to locate similar segments in sequences is to initially search for relatively short matching (or nearly matching) segments, and then look for alignments around the match with similarity scores that are significantly high. *Seeds* give the shape of the matching segments. Short seeds can occur many times even in sequences with non-similar structure, but searching for long exact matches can result in missing segments whose underlying structure is the same. Thus a trade-off is beneficial.

Spaced seeds (Ma et al. 2002; Keich et al. 2004; Buhler et al. 2005) provide a way to increase the probability of observing at least one matching segment (or *seed hit*) without simultaneously increasing the number of matches that occur at random. A spaced seed is a pattern $S = s_1,\ldots,s_k$ from $\{1,*\}$, with $s_1 = s_k = 1$. A "1" indicates a position where sequence symbols must match, and "*" indicates a position where a match isn't required.

Let $\mathbf{X} = x_1,\ldots,x_n$ be the binary sequence formed by aligning two DNA segments of length n and assigning a value $x_j = 1$ if the j-th position of the segments match, and $x_j = 0$ for a mismatch. Spaced seed S *hits* or *occurs* in $\mathbf{X}$ at position υ if for $j = 1,\ldots,k$, $x_{\upsilon-k+j} = 1$ whenever $s_j = 1$. A "1" at position $\upsilon - k + j$ of $\mathbf{X}$ ($\upsilon \in (k,\ldots,n)$ and $j \in (1,\ldots,k)$) is *covered* by a seed hit if S occurs at position υ and for that occurrence, $s_j = 1$. As an example, for spaced seed $S = 11*1$ and sequence segment

$$\mathbf{X} = \underset{\bullet\bullet}{11}\underset{\bullet}{1}00\underset{\bullet\bullet}{11}0\underset{\bullet\bullet\bullet\bullet}{1111}$$

of length $n = 15$, there are seed hits at sequence positions 4, 11, 14 and 15, and the ten 1's with "$\bullet$" underneath are covered.

One solution to the trade-off between the use of short and long seeds is to use short seeds and require multiple seed hits clustered close together to trigger an alignment. A large value of spaced seed coverage then seems reasonable as a criterion to trigger a full alignment. Benson (1999) used this approach with the number of successes in success runs of length at least k (a spaced seed of length k with $r = 0$ stars) serving as the test statistic, and used a normal approximation to the statistic's distribution. Martin (2006) gave the exact

distribution of that statistic for Markovian sequences of a general order. Benson and Mak (2009) gave a method to compute the distribution of a spaced seed in the i.i.d. case, and Martin and Noé (2015) extended that work to higher-order Markovian sequences, with the efficiency of the method allowing computation for the longer seeds that are used in practice. In that paper, the state space of an auxiliary Markov chain for computation was formed in a sequential manner. The state space of their algorithm was not necessarily the minimal one, but it nonetheless was small and facilitated fast computation. We show how to set up a minimal state space for the coverage problem using Theorem 1, and answer the question raised in the latter reference of whether it makes sense computationally to continue the search for equivalent states beyond the procedure of that paper so that a minimal state space is obtained. First we lay out the basic structure of the algorithm of Martin and Noé (2015) to set up the AMC state space, without going into details on the computation of the distribution using that state space.

4.2 AMC state space through sequential elimination of equivalent states

Now let W be the set of 2^r possible seed words that are formed by replacing each of the r stars by either 0 or 1. Martin and Noé (2015) defined the AMC states as having a string $q \in Q$ (see Section 3) indicating progress toward a seed hit, as well as a coverage string c that indicates positions of q that were previously covered. This is in contrast to the formulation of Martin and Coleman (2013), who simply extended strings to overlapping seed hits.

An important result of Martin and Noé (2015) is that strings q and q' are equivalent if they have the same length and coverage strings, and equivalent active proper suffixes. Based on that result, a sequential method was used to set up the AMC state space, which we denote by Λ. In the method, first the m-tuples are set up, and then sequentially over lengths $m+1,\ldots,k-1$, proper prefixes of seed words are generated, and equivalent states are combined by discarding the one not already in Λ. The strings of Q are associated with coverage strings that are empty to represent m- tuples and prefixes of non-overlapping occurrences of strings of W.

For seed hits, i.e. when $|q| = k-1$ and the new symbol $x_i = 1$, $q \to q' = aps(q \cdot 1)$, and the new coverage string is $c' = (c_{tem})_{|aps(q\cdot 1)|}$, where the coverage template $c_{tem} \equiv (c_{tem,1},\ldots,c_{tem,k})$ has the symbol "$\bullet$" if $s_j = 1$, and a blank position otherwise. The number of new states η that is added in this stage is recorded.

Now for each of the η new states just defined and input symbol $x_i \in \{0,1\}$ the transitions of q and c are obtained. The string q transitions exactly as when there is no coverage. To update c, if there is no seed hit, then $c \to c'$, where c' is formed by concatenating a zero to the end of c and then taking the suffix of the same length as the new prefix string q'. If there is a seed hit, c' is the suffix of length $|q'|$ of the string of length k that has a "1" in any position where at least one of $c \cdot 0$ or c_{tem} has a "1", and zeroes elsewhere.

If, during any stage of the state generation process, a destination state $\begin{pmatrix} q' \\ c' \end{pmatrix}$ already exists in Λ, the transition is mapped there. Otherwise a new state is generated to receive the

transition. At the beginning of each stage, η is reset to 0 and incremented by one as each new state is generated. The procedure for generating states on state transitions is repeated for the new strings at each stage while $\eta > 0$.

When attempting to enter a state $\begin{pmatrix} q' \\ c' \end{pmatrix}$ into Λ and searching for a possible matching state, if there is a state $\begin{pmatrix} q* \\ c* \end{pmatrix}$ with $q^* = q'$ but $c^* \neq c'$, the algorithm of Martin and Noé (2015) checks to see if $aps(q') \in \hat{Q}_{nosuf}$. In that case, none of the 1's of q' and $q*$ can possibly be involved in an overlapping seed hit that is not a direct one, and thus as long as the updates to coverage on the direct hits of q' and q^* are the same, the two states are combined, i.e. no new state is created. However, it could be that $aps(q') \notin \hat{Q}_{nosuf}$, yet the strings are equivalent and would be combined by a minimization algorithm. The conjecture of Martin and Noé (2015) was that searching for such strings would be more computationally costly than the advantage in possibly finding a smaller state space, and thus a minimal state space was not obtained in all cases. In the present paper, extend the algorithm of Martin and Noé (2015) to obtain a minimal state space using Theorem 1, and compare the computation time for obtaining the coverage distribution for a seed that is used in practice.

4.3 Obtaining a Minimal AMC state space

The main difference between the present paper and the algorithm of Martin and Noé (2015) is that in this work we continue to check for equivalency of strings in the case where $q^* = q'$, $c^* \neq c'$, and $aps(q') \notin \hat{Q}_{nosuf}$, while Martin and Noé (2015) do not. Implied by $q^* = q'$ is that the strings have the same length, active proper suffix, completion strings and failure sequence. Thus Theorem 1 is satisfied if updates on direct hits beginning in the various locations of the failure sequences are equal.

We then check for equality of coverage updates for direct hits from $fs_q(j)$ and $fs_{q^*}(j)$. The check is carried out in a sequential manner for $j = 1,\ldots,\left|fs_q\right|$, beginning with $j = 1$ and continuing if all the previous checks were satisfied. At each stage we note the completion strings, as the completion of a string could imply completion of a longer string, which would render certain positions as covered, and thus could have a bearing on whether coverage updates must be equal. If all the checks are true, the states are equivalent.

We use as an example the seed 1*111*11 with $m = 1$, where the algorithm of Martin and Noé (2015) obtained a state space with 38 states, whereas the minimal state space has 35. State $\underset{\cdot\cdot\cdot\ \cdot\cdot}{1111111}$ is equivalent to $\underset{\cdot\cdot\cdot\ \cdot\cdot}{1111111}$, and state $\underset{\cdot\ \ \cdot\cdot}{101110}$ is equivalent to $\underset{\cdot\ \ \cdot\cdot}{101110}$, so that $\underset{\cdot\ \ \cdot\cdot}{1011101}$ and $\underset{\cdot\ \ \cdot\cdot}{1011101}$ are also equivalent. However, since the respective active proper suffixes of the strings are not in $\hat{Q}_{nosuf}$, the algorithm of the latter reference did not identify the equivalencies. Here we show equivalency of states $\begin{pmatrix} q' \\ c' \end{pmatrix} = \underset{\cdot\cdot\cdot\ \cdot\cdot}{1111111}$ and $\begin{pmatrix} q* \\ c* \end{pmatrix} = \underset{\cdot\cdot\cdot\ \cdot\cdot}{1111111}$.

The respective failure sequences are

$fs_{q'} = \left(1\dot{1}\dot{1}\dot{1}\dot{1}\dot{1}\dot{1}, 1\dot{1}\dot{1}\dot{1}\dot{1}\dot{1}, 1\dot{1}\dot{1}\dot{1}\dot{1}, 1\dot{1}\dot{1}\dot{1}, 1\dot{1}\dot{1}, 1\dot{1}, 1\right)$ and $fs_{q^*} = \left(1\dot{1}\dot{1}\dot{1}\dot{1}\dot{1}\dot{1}, 1\dot{1}\dot{1}\dot{1}\dot{1}\dot{1}, 1\dot{1}\dot{1}\dot{1}\dot{1}, 1\dot{1}\dot{1}\dot{1}, 1\dot{1}\dot{1}, 1\dot{1}, 1\right)$.

Strings q' and $q*$ are completed with the symbol 1, and the update to coverage is +3 in both cases, leaving all positions covered. This implies that updates must be the same for strings 111111, 1111, 111, and 11, as to complete those strings the first symbol must be 1, so that q and $q*$ hit. States $1\dot{1}\dot{1}\dot{1}\dot{1}$ and $1\dot{1}\dot{1}\dot{1}\dot{1}$ are completed with strings of the form *11 with coverage update +3 for 011, and with 111 implying that q and $q*$ must hit, rendering all positions as covered. For $\dot{1}$ and 1, the completion strings begin with *11, implying that 11111 hits, which covers the last 1 so that updates must be the same. Thus the states are equivalent.

The algorithm was programmed in MATLAB (version R2010b), and applied to computing the coverage distribution of the spaced seed 111*1**1*1**11*111 that was used in Version 1 of the Patternhunter software (Ma et al. 2002). A Dell PC with an Intel Core i7 CPU 873 with 2.93 GHz and 8 GB RAM was used for the computations. The Patternhunter seed is optimal in the sense that it has the highest single hit probability for Bernoulli trials with match probability $p = 0.7$ on alignment length $n = 64$ (Ma et al. 2002). The size of the state space using the algorithm of Martin and Noé (2015) was 4215, whereas the minimal state space that was obtained using the present algorithm is of size 3782. The computation using the current algorithm of took 72.0s (43.8s for setting up the state space and 28.2s to compute the distribution), compared to 65.3s using the algorithm of Martin and Noé (2015) (26.6s/38.7s). Thus, as expected, the new algorithm took more time to set up the state space and less to do the computation. These times may be compared to the over 26 hours that were required to set up a "full" state space of 321,596 strings that are prefixes of patterns extended to overlapping pattern occurrences (this type of state space was used in the initial stage of AMC setup by Martin and Coleman (2011) in computing the distribution of coverage of clumps, with the Hocroft algorithm then applied to minimize the state space). Thus, whereas the present algorithm uses more total computation time than the algorithm of Martin and Noé (2015), surprisingly the difference in times is small. This shows that an extensive search to find a minimal state space for the spaced seeds coverage problem is feasible, and leaves open the possibility that applications of the method of this paper to other settings could yield favorable results.

5. Application to coverage of a spaced seed

This paper deals with computing distributions of statistics using an AMC. The paper focuses on reducing the number of states in the AMC in situations where first forming an AMC state space and then applying a minimization algorithm to reduce its size is not feasible. A situation where this can occur is where overlapping pattern occurrences are reckoned differently than non-overlapping occurrences, rendering a need for using prefixes of patterns extended to overlapping pattern occurrences.

A result is given that characterizes equivalent states, and facilitates elimination of any extra states during the process of state space formation. The method was applied to setting up a minimal state space for computing the distribution of coverage of a spaced seed. It was shown that a minimal state space may be obtained in computation time that is slightly more

than the algorithm of Martin and Noé (2015). In future work, we would like to apply the algorithm to the computation of other pattern distributions.

Acknowledgements

D.E.K. Martin was supported in this research by the National Science Foundation under Grant DMS-1107084.

References

Aho, A.V. and Corasick, M.J. (1975). Efficient string matching: An aid to bibliographic search. *Communications of the ACM, 18(6)*, 333-340.

Aston, J.A.D. and Martin, D.E.K. (2007). Distributions associated with general runs and patterns in hidden Markov models. *Annals of Applied Statistics*, *1(2*), 585-611.

Benson, G. (1999). Tandem repeats finder: a program to analyze DNA sequences. *Nucleic Acids Research*, *27*, 573-580.

Benson, G. and Mak, D.Y.F. (2009). Exact distribution of a spaced seed statistic for DNA homology detection. *String Processing and Information Retrieval, Lecture Notes in Computer Science, 5280*, 282-293.

Buhler, J., Keich, U., Sun, Y. (2005). Designing seeds for similarity search in genomic DNA. *Journal of Computer and System Sciences*, *70*, 342-363.

Ebneshahrashoob, M., Gao, T., Wu, M. (2005). An efficient algorithm for exact distribution of discrete scan statistic. *Methodology and Computing in Applied Probability*, *7*, 459-471.

Fu, J.C. and Koutras, M.V. (1994). Distribution theory of runs: a Markov chain approach. *Journal of the American Statistical Association*, *89*, 1050–1058.

Hopcroft, J.E. (1971). An *n* log *n* algorithm for minimizing states in a finite automaton. In: Z. Kohavi and A. Paz (Ed.) *Theory of Machines and Computations*, (pp. 189-196). New York: Academic Press.

Hopcroft, J.E., Motwani, R., and Ullman, J.D. (2001). *Introduction to automata theory, languages, and computation.* New York: Addison-Wesley.

Keich, U., Li, M., Ma, B., Tromp, J. (2004). On spaced seeds for similarity search. *Discrete applied mathematics*, *138(3)*, 253-263.

Koutras, M.V. and Alexandrou, V.A. (1995). Runs, scans and urn models: A unified Markov chain approach. *Annals of the Institute of Statistical Mathematics*, *47*, 743-766.

Lladser, M., Betterton, M.D., Knight, R. (2008). Multiple pattern matching: A Markov chain approach. *Journal of Mathematical Biology*, *56(1-2*), 51-92.

Ledent, S. and Robin, S. (2005). Checking homogeneity of motifs' distribution in heterogenous sequences. *Journal of Computational Biology*, *12*, 672-685.

Ma, B., Tromp, J., Li, M. (2002). Patternhunter-faster and more sensitive homology search. *Bioinformatics*, *18(3)*, 440-445.

Marshall, T. and Rahmann, S. (2008). Probabilistic arithmetic automata and their application to pattern matching statistics. *Lecture Notes In Computer Science*; Vol. 5029, *Proceedings of the 19th Annual Symposium on Combinatorial Pattern Matching* (pp. 95-106).

Martin, D.E.K. (2006). The exact joint distribution of the sum of heads and apparent size statistics of a "tandem repeats finder" algorithm. *Bulletin of Mathematical Biology*, *68*, 2353-2364.

Martin, D.E.K. and Aston, J.A.D. (2008). Waiting time distribution of generalized later patterns. *Computational Statistics and Data Analysis*, *52*, 4879-4890.

Martin D.E.K. and Aston, J.A.D. (2013). Distributions of statistics of hidden state sequences through the sum-product algorithm. *Methodology and Computing in Applied Probability*, *15(4)*, 897-918.

Martin D.E.K. and Coleman, D.A. (2011). Distributions of clump statistics for a collection of words. *Journal of Applied Probability*, *48*, 1049-1059.

Martin D.E.K and L. Noe (2015). Faster exact probabilities for statistics of overlapping pattern occurrences. (In press, *Annals of the Institute of Statistical Mathematics*, doi 10.1007/210463-015-0540-y).

Nuel, G. (2008). Pattern Markov chains: Optimal Markov chain embedding through deterministic finite automata. *Journal of Applied Probability, 45(1)*, 226-243.

Ribeca, P. and Raineri, E. (2008). Faster exact Markovian probability functions for motif occurrences: a DFA-only approach. *Bioinformatics 24(24)*, 2839-2848.

Tewari, A., Srivastava, U., Gupta, P. (2002). A parallel DFA minimization algorithm. *Lecture Notes in Computer Science, Vol. 2552*, (pp. 34-40).