IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 16, 2019, accepted July 4, 2019, date of publication July 9, 2019, date of current version July 31, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2927534

A New Method of Creating Minimal-Order Markov Set and Transition States of M/N Sliding Window

KANG ZHAO[1,2], MINGHUA ZHU[1], XUGUANG YANG[2], AND JIAONAN ZHANG[3]

[1]School of Computer Science and Software Engineering, East China Normal University, Shanghai 200062, China
[2]Shanghai IOT Company Ltd., Shanghai 200089, China
[3]Fujian Tianma Science and Technology Group Company Ltd., Fuqing 350308, China

Corresponding author: Minghua Zhu (mhzhu@sei.ecnu.edu.cn)

This work was supported in part by the Science and Technology Commission of Shanghai Municipality under Grant 17511106902, in part by the Shanghai Rising-Star Program under Grant 18QB1403900, and in part by the Fujian Province STS project under Grant 2017T3009.

ABSTRACT M/N sliding window detection is the most commonly used method in secondary decision systems, of which the method seems simple, yet evaluating the probability of the existence of the target after several scans is rather complex. As the minimal-order Markov set is needed in the process of computation, through the analysis of equivalence of states, several conclusions of the minimal-order Markov set for M/N sliding window can be derived, and then the methods for judging whether a state is a smallest equivalent state and hence, converting a state to the smallest equivalent state can be determined. Based on these conclusions, a new method that can directly create the minimal-order Markov set has been proposed, which, compared with the existing method, can greatly improve the computational efficiency of the creation. In this paper, a new method which is easier to implement and has low time complexity for calculating transition states within the minimal-order Markov set is also proposed.

INDEX TERMS M/N sliding window, minimal-order Markov set, decision system.

I. INTRODUCTION

For various reasons, due to the electronic components used in the sensor application field, the result of one single scan is not always 100% accurate when detecting a target. Therefore, it is often necessary to perform multiple consecutive scans before declaring the existence of a target. So, the whole process can be divided into two processes: scan process and acquisition process. We call the ''multiple consecutive scans'' a scan process. In the case where the target exists, each scan has a probability of α detecting the target. α can be a constant, or it can be different each time. We call the subsequent ''declaring the existence of a target'' an acquisition process. Finally, the probability of declaring the existence of a target is taken as the probability of acquisition. The M/N sliding window detection is the most commonly used method in decision systems. Unlike consecutive-k-out-of-n (F system) [1]–[6], M/N sliding window detection is devoted to studying whether a target exists. According to the principle of the M/N sliding window detection, if M or more successful results occur within N scans, the target is considered to ''exist'', and then the system enters the accepting state (sometimes also called ''absorption state'' in some literatures). After entering accepting state (hereafter A state for short), we believe that the system will remain in this state regardless of what kind of scan results that may follow. This method, which belongs to a continuous inspection scheme [7], is simple and practical, and has been widely used in the military field [8], for example, to determine whether a target exists in the secondary decision system for radars and sonars. In general, we are interested in evaluating the acquisition probability of the existence of the target after a certain number of scans. And if we ignore or do not take ''the sliding of window'' into consideration, this would be a very simple binomial distribution [9]. Actually, the process of calculating acquisition probability is a Markov problem [10], [11]. Worsham [12] presents a sequence structure to analyze the problem, which, however, in our opinion, is still essentially a Markov process. Usually, we represent each Markov state with a binary sequence [8], [13]. Each time the new probe value moves in from the right side, '1' for a success result, and '0' for a fail result, and the oldest result exceeding the

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

window is removed from the left side of the window. We will take the decimal h for the Markov state, which corresponds to the binary sequence. Castella [14] conducted a theoretical analysis of the calculation of the acquisition probability of M/N sliding window, but only gives the probability formula under the condition of $1 \leq N \leq 4$, $M \leq N$. Paper in [15] only solved the problem in some cases with N less than 5. There were no algorithms for the minimal-order Markov set creation, state transition and probability on other common situations in [14], [15]. Todd [16] proposed a general algorithm for acquisition probability based on the minimal-order Markov model, but the details haven't been given in the paper. Williams [17] presented a recursive algorithm for the creation of minimal-order Markov set of M/N sliding windows. An iterative method for calculating the acquisition probability was also proposed in [17]. Besides, the author analyzed the equivalence of states and the transition states on the basis of observation only, without detailed theoretical explanation. Abraham [18] used the transition matrix and matrix factorization to derive the formula for calculating the acquisition probability, however the author still followed the Williams's method in the creation of the minimal-order Markov set and the calculation of the transition states.

In general, the evaluation process of the acquisition probability of both [17] and [18] can be divided into the following two steps:

1) Create the minimal-order Markov set; We call this step CMMS process for short; [17], [18] used the exact same CMMS method. Here, we call it the Williams CMMS algorithm.
2) Calculate the acquisition probability. We call this step CAP process for short. The method proposed in the paper [17] is called the Williams CAP algorithm, and the method in the paper [18] is called the Abraham CAP algorithm. We will discuss this in Section III.

Therefore, the prelude to calculating the acquisition probability is to create the minimal-order Markov set.

Williams provided algorithms 5 thru 7 in [17] to complete the entire CMMS step. Here is a brief description of the Williams CMMS algorithm:

1) Add state A to the minimal-order Markov set;
2) Generate state $h = 0$, then calculate the transition states h_1, h_2 of h, and calculate the smallest equivalent states h'_1, h'_2 of these transition states; Add h to the minimal-order Markov set;
3) Determine whether the newly generated h'_1 and h'_2 have already been included in the minimal-order Markov set. Regardless of which transition state, as long as it is not included in the set, step 2) is executed again, and just replace "$h = 0$" in step 2) with "$h = h'_1$" or "$h = h'_2$".

The algorithm for calculating the smallest equivalent state in [17] is a complex recursive algorithm. For more details, please refer to section 5 of the literature [17].

The main reason why Williams CMMS algorithm is not efficient is that the algorithm inevitably needs to calculate the transition states h_1, h_2, and these transition states cover almost the entire original Markov states.

The remainder of this article first introduces the terminologies used in the paper. Two CAP algorithms are reviewed in Section III. Then, based on equivalence analysis of the Markov states, we infer the features of the states in the minimal-order Markov set of M/N sliding window. Thus, the problem of creating the minimal-order Markov set is transformed into a problem of combinatorics, hence a new method arises, which does not need to calculate the intermediate transition states. Therefore, it is more efficient. In addition, new methods for the calculation of transition states have been proposed, which are more concise and efficient. In Section V, we discuss the performance of the algorithm and give a comparison of the run time of the proposed CMMS method and Williams CMMS algorithm. At the end of the paper, we present the acquisition probability distributions of different situations in the form of an attachment.

II. TERMINOLOGY

It is well known that any integer has a corresponding decimal representation and binary representation. The figure below is an example of a state in 4/6 sliding window:

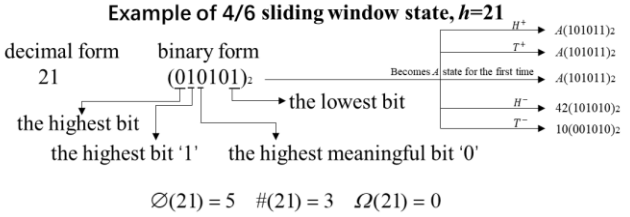

FIGURE 1. Examples of decimal and binary representation.

As shown in the Figure (1), the leftmost (i.e. highest) bit of the binary sequence within the sliding window (N indicates its size) is called "the highest bit". The rightmost (i.e. lowest) bit within the window is "the lowest bit". The '1' closest to the leftmost bit of the binary sequence within the window is "the highest bit '1'". Among all '0's to the right of the highest bit '1', the highest one is called "the highest meaningful bit '0'" of this binary sequence.

Next, we introduce a few symbols used in this paper: $>\overset{N}{>}>$, $<\overset{N,0}{<}<$, $<\overset{N,1}{<}<$, $\emptyset$, #, Ω, H^+, H^-, T^+, T^-.

$>\overset{N}{>}>$ stands for logical right shift within the window, N for the size of the sliding window. $>\overset{N}{>}>$ indicates that after logical right shift, the highest bit within the window is always padded by '0'. E.g.:

$$21 >\overset{6}{>}> 2 = 5$$

$<\overset{N,0}{<}<$ stands for negative left shift within the window, N for the size of the sliding window. '0' indicates that after left shift, the lowest bit within the window is always padded by '0'. E.g.:

$$21 <\overset{6,0}{<}< 3 = 40$$

$\overset{N,1}{<<<}$ stands for positive left shift within the window, N for the size of the sliding window. '1' indicates that after left shift, the lowest bit within the window is always padded by '1'. E.g.:

$$21 \overset{6,1}{<<<} 3 = 47$$

$\emptyset(h)$ represents the number of bits of the state h excluding leading '0's. E.g.:

$$\emptyset(21) = 5$$

$\#(h)$ means the total number of '1's contained by the binary sequence of h, also known as the weight of h; E.g.:

$$\#(21) = 3$$

$\Omega(h)$ is defined as the number of '1's which overflow on the left of the sliding window during an unbroken sequence of successful scans which transform a non-A state h into an A state for the first time; E.g., for 4/6 sliding window:

$$\Omega(21) = 0$$

$h.H^+$ is the next state from state h after a success result; i.e.:

$$h.H^+ = h \overset{N,1}{<<<} 1$$

E.g., for 4/6 sliding window state $h = 21$:

$$21.H^+ = A$$

$h.H^-$ is the next state from state h after a fail result; i.e.:

$$h.H^- = h \overset{N,0}{<<<} 1$$

E.g., for 4/6 sliding window state $h = 21$:

$$21.H^- = 42$$

$h.T^+$ is the smallest equivalent state of $h.H^+$. We will discuss it later in Part D, Section IV; E.g., for 4/6 sliding window state $h = 21$:

$$21.T^+ = A$$

$h.T^-$ is the smallest equivalent state of $h.H^-$. We will discuss it later in Part D, Section IV. E.g., for 4/6 sliding window state $h = 21$:

$$21.T^- = 10$$

III. PROBABILITY

According to the principle of the M/N sliding window, if $\#(h) \geq M$ after several scans, the target is taken as "exist". Its detection model can be represented by the following figure.

In the M/N sliding window detection system, the current state completely determines its next or future state, i.e. the state at time k is determined by its previous state at time $k-1$. Therefore, the process of state transition is a Markov process. Generally speaking, we are more concerned about the probability of the system in a certain state at a certain moment.

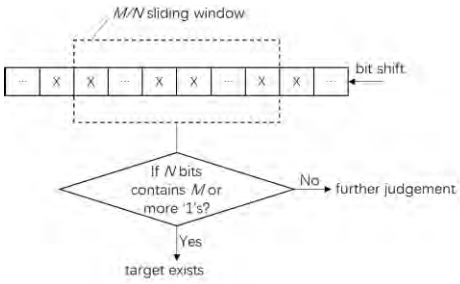

FIGURE 2. Principle of M/N sliding window detection.

A. WILLIAMS CAP ALGORITHM

The probability (indicated by the symbol $p_{h_i}[k]$) that the system will be in state h_i at time k can be expressed by

$$p_{h_i}[k] = \alpha[k]p^+_{h_i}[k-1] + (1-\alpha[k])\,p^-_{h_i}[k-1] \quad (1)$$

Here, $\alpha[k]$ denotes the probability of successful scan at time k. It can be the same each time, or it can be different each time. Symbol $p^+_{h_i}[k-1]$ indicates the probability of a system in certain states at time "k-1", and these states need to meet such condition: the system happens to change to state h_i when the result of the next scan is '1'; $p^-_{h_i}[k-1]$ indicates the probability of a system in certain states at time "k-1", and these states need to meet such condition: the system happens to change to state h_i when the result of the next scan is '0'. Arrange all states in ascending order into $h_0\ h_1 \cdots h_{n_s}$, among which $h_0 = 0$, $h_{n_s} = A$, and n_s stands for the size of the Markov set. $p^+_{h_i}[k-1]$ and $p^-_{h_i}[k-1]$ are defined as polynomial

$$p^+_{h_i}[k-1] = \sum_{j=0}^{n_s} \delta^+_{h_j h_i} p_{h_j}[k-1] \quad (2)$$

$$p^-_{h_i}[k-1] = \sum_{j=0}^{n_s} \delta^-_{h_j h_i} p_{h_j}[k-1] \quad (3)$$

$\delta^+_{h_j h_i}$, $\delta^-_{h_j h_i}$ are indication functions of state transition

$$\delta^+_{h_j h_i} = \begin{cases} 1, & h_j.H^+ = h_i \\ 0, & h_j.H^+ \neq h_i \end{cases} \quad (4)$$

$$\delta^-_{h_j h_i} = \begin{cases} 1, & h_j.H^- = h_i \\ 0, & h_j.H^- \neq h_i \end{cases} \quad (5)$$

From Equations (4) and (5) we can easily draw

$$\delta^+_{Ah_i} = \delta^-_{Ah_i} = 0, \quad h_i \neq A \quad (6)$$

$$\delta^+_{AA} = \delta^-_{AA} = 1 \quad (7)$$

Since none of the non-A states can transit to state A after '0' is detected, Equation (8) holds true all the time.

$$\delta^-_{h_j,A} = 0, \quad h_j \neq A \quad (8)$$

In addition, the initial values are given by equations (9) and (10).

$$p_{h_j}^{+}[-1] = \begin{cases} 1, & h_j = h_o \\ 0, & h_j \neq h_o \end{cases} \tag{9}$$

$$\alpha\ [0] = 1 \tag{10}$$

where h_o refers to the initial state value of the system. Under normal circumstances, when the sliding window system is just started, $h_o = 0$ and $p_0^{+}[-1] = 1$.

Therefore, acquisition probability $p_A[k]$ is

$$p_A[k] = \alpha[k] p_A^{+}[k-1] + (1-\alpha[k])\, p_A^{-}[k-1] \tag{11}$$

$f_K[k]$ is defined as the probability of entering the A state for the first time at time k, which is defined as the stop time, and K represents the stop time random variable.

$$f_K[k] = \alpha[k] \left(\sum_{j=0}^{n_s-1} \delta_{h_j A}^{+} p_{h_j}[k-1] \right) + (1-\alpha[k]) \left(\sum_{j=0}^{n_s-1} \delta_{h_j A}^{-} p_{h_j}[k-1] \right) \tag{12}$$

What's interesting is that, if you modify Equation (7) to $\delta_{AA}^{+} = \delta_{AA}^{-} = 0$, then Equation (11) is converted to the Equation (12). Keeping Equation (7) unchanged, it is actually easy to prove Equation (13), which also meets the ''absorption'' characteristics of entering the A state.

$$p_A[k] = f_K[k] + p_A[k-1] \tag{13}$$

We can see that $f_K[k]$ is a probability mass function (PMF) of the stopping time K.

So, there is (Use the new symbol $F_K[k]$ to denote $p_A[k]$)

$$F_K[k] = p_A[k] = \sum_{l=0}^{k} f_K(l) \tag{14}$$

It can be seen that $F_K[k]$ (i.e., $p_A[k]$) is a cumulative distribution function(CDF) of K, which means the probability of acquisition within k scans.

To make it easier to discuss the problem later, we use the symbol $\rho_A[k]$ to represent $f_K[k]$.

$$\rho_A[k] = f_K[k] = \alpha[k]\rho_A^{+}[k-1] + (1-\alpha[k])\, \rho_A^{-}[k-1] \tag{15}$$

Symbol $\rho_{h_i}^{+}[k-1]$ and $\rho_{h_i}^{-}[k-1]$ are defined as

$$\rho_{h_i}^{+}[k-1] = \sum_{j=0}^{n_s-1} \delta_{h_j h_i}^{+} \rho_{h_j}[k-1] \tag{16}$$

$$\rho_{h_i}^{-}[k-1] = \sum_{j=0}^{n_s-1} \delta_{h_j h_i}^{-} \rho_{h_j}[k-1] \tag{17}$$

Formally speaking, Equation (15) and Equation (11) are similar, but Equation (11) represents CDF and Equation (15) represents PMF. Interestingly, if you modify Equation (7) to $\delta_{AA}^{+} = \delta_{AA}^{-} = 0$, then Equation (11) is transformed into Equation (15).

Because of the nature of Equation (8), Equation (15) can be further simplified to

$$\rho_A[k] = \alpha[k]\rho_A^{+}[k-1] \tag{18}$$

The initial values definition are similar

$$\rho_{h_j}^{+}[-1] = \begin{cases} 1, & h_j = h_o \\ 0, & h_j \neq h_o \end{cases} \tag{19}$$

$$\alpha\ [0] = 1 \tag{20}$$

Now we discuss the impact of the state transition table on probability calculations. It is easy to see from Equations (2) and (3) or Equations (16) and (17) that the process of these cycle accumulation are affected by the n_s, and the larger the value of n_s, the larger the amount of calculation.

The entire states and their transition states of the 4/6 sliding window are given in Table (1).

Obviously, the shaded states in Table (1) are actually the same state, and can be merged into state A. In addition, from the observation we can find that the highest bit (either '0' or '1') of any non-A state has no effect on the state transition, as the highest bit will be moved out of the window when moving to the next state. Thus, state 32 can be merged with state 0, state 33 can be merged with state 1, and so on. Finally, the right half of Table (1) can be completely omitted.

It should be noted that the state merging has no effect on the calculation of probability. Assume that the calculation of $\rho_{h_i}[k]$ contains the item $\rho_{h_a}[k-1]$, and if it is known that h_b is equivalent to h_a, then it is reasonable to replace $\rho_{h_a}[k-1]$ with $\rho_{h_b}[k-1]$.

Table (1) can be further simplified to Table (2) by state merging.

As will be explained in this paper, the states in Table (2) can be further merged to make n_s smaller. Therefore, CMMS is necessary for improving computational efficiency.

B. ABRAHAM CAP ALGORITHM

Abraham [18] uses the state transition matrix to calculate probability. We use the vector $\boldsymbol{p}[k]$ to represent the probability that the system is in various states at time k. $\boldsymbol{H}[k]$ represents the state transition matrix from time k-1 to time k.

$$\boldsymbol{p}^T[k] = \boldsymbol{p}^T[k-1]\boldsymbol{H}[k] \tag{21}$$

Among which,

$$\boldsymbol{H}[k] = \begin{bmatrix} \beta_{h_0 h_0}[k] & \beta_{h_0 h_1}[k] & \cdots & \beta_{h_0 h_{n_s}}[k] \\ \beta_{h_1 h_0}[k] & \beta_{h_1 h_1}[k] & \cdots & \beta_{h_1 h_{n_s}}[k] \\ \vdots & \vdots & \vdots & \vdots \\ \beta_{h_{n_s} h_0}[k] & \beta_{h_{n_s} h_1}[k] & \cdots & \beta_{h_{n_s} h_{n_s}}[k] \end{bmatrix} \tag{22}$$

$$\boldsymbol{p}^T[k] = \begin{bmatrix} p_{h_0}[k] & p_{h_1}[k] & \cdots & p_{h_{n_s}}[k] \end{bmatrix} \tag{23}$$

Similarly, n_s stands for the size of the Markov set. The superscript $^{\mathrm{T}}$ represents the transpose operation. $p_{h_i}[k]$ denotes the probability that the system will be in state

TABLE 1. 4/6 entire states and transition states.

h	Next state		h	Next state	
	H^+	H^-		H^+	H^-
$0(000000)_2$	1	0	$32(100000)_2$	1	0
$1(000001)_2$	3	2	$33(100001)_2$	3	2
$2(000010)_2$	5	4	$34(100010)_2$	5	4
$3(000011)_2$	7	6	$35(100011)_2$	7	6
$4(000100)_2$	9	8	$36(100100)_2$	9	8
$5(000101)_2$	11	10	$37(100101)_2$	11	10
$6(000110)_2$	13	12	$38(100110)_2$	13	12
$7(000111)_2$	A	14	$39(100111)_2$	A	A
$8(001000)_2$	17	16	$40(101000)_2$	17	16
$9(001001)_2$	19	18	$41(101001)_2$	19	18
$10(001010)_2$	21	20	$42(101010)_2$	21	20
$11(001011)_2$	A	22	$43(101011)_2$	A	A
$12(001100)_2$	25	24	$44(101100)_2$	25	24
$13(001101)_2$	A	26	$45(101101)_2$	A	A
$14(001110)_2$	A	28	$46(101110)_2$	A	A
$15(001111)_2$	A	A	$47(101111)_2$	A	A
$16(010000)_2$	33	32	$48(110000)_2$	33	32
$17(010001)_2$	35	34	$49(110001)_2$	35	34
$18(010010)_2$	37	36	$50(110010)_2$	37	36
$19(010011)_2$	A	38	$51(110011)_2$	A	A
$20(010100)_2$	41	40	$52(110100)_2$	41	40
$21(010101)_2$	A	42	$53(110101)_2$	A	A
$22(010110)_2$	A	44	$54(110110)_2$	A	A
$23(010111)_2$	A	A	$55(110111)_2$	A	A
$24(011000)_2$	49	48	$56(111000)_2$	49	48
$25(011001)_2$	A	50	$57(111001)_2$	A	A
$26(011010)_2$	A	52	$58(111010)_2$	A	A
$27(011011)_2$	A	A	$59(111011)_2$	A	A
$28(011100)_2$	A	56	$60(111100)_2$	A	A
$29(011101)_2$	A	A	$61(111101)_2$	A	A
$30(011110)_2$	A	A	$62(111110)_2$	A	A
$31(011111)_2$	A	A	$63(111111)_2$	A	A

h_i at time k. $\beta_{h_j h_i}[k]$ represents the probability that the state transitions from h_j to h_i from time k-1 to time k.

$$\beta_{h_j h_i}[k] = \alpha[k]\delta^+_{h_j h_i} + (1 - \alpha[k])\delta^-_{h_j h_i} \tag{24}$$

$\alpha[k]$ denotes the probability of successful scan at time k. The definitions of $\delta^+_{h_j h_i}$ and $\delta^-_{h_j h_i}$ are the same as Equations (4) and (5).

Arrange all states in ascending order into $h_0\ h_1 \cdots h_{n_s}$, among which $h_0 = 0$, $h_{n_s} = A$. So, it is clear that $\beta_{h_{n_s} h_0}[k] = \beta_{h_{n_s} h_1}[k] \cdots = \beta_{h_{n_s} h_{n_s-1}}[k] = 0$, $\beta_{h_{n_s} h_{n_s}}[k] = 1$.

Divide $\boldsymbol{p}^T[k]$ and $\boldsymbol{H}[k]$:

$$\begin{aligned} \boldsymbol{p}_k^T &= \left[p_{h_0}[k]\ p_{h_1}[k]\ \ldots\ \left| p_{h_{n_s}}[k] \right. \right] \\ &= \left[p_c[k]\ p_A[k] \right] \end{aligned} \tag{25}$$

$$\boldsymbol{H}[k] = \begin{bmatrix} \boldsymbol{H}_{cc}[k] & \boldsymbol{p_{cs}[k]} \\ \boldsymbol{O^T} & 1 \end{bmatrix} \tag{26}$$

TABLE 2. 4/6 merged table.

h	Next state	
	H^+	H^-
$0(000000)_2$	1	0
$1(000001)_2$	3	2
$2(000010)_2$	5	4
$3(000011)_2$	7	6
$4(000100)_2$	9	8
$5(000101)_2$	11	10
$6(000110)_2$	13	12
$7(000111)_2$	A	14
$8(001000)_2$	17	16
$9(001001)_2$	19	18
$10(001010)_2$	21	20
$11(001011)_2$	A	22
$12(001100)_2$	25	24
$13(001101)_2$	A	26
$14(001110)_2$	A	28
$16(010000)_2$	1	0
$17(010001)_2$	3	2
$18(010010)_2$	5	4
$19(010011)_2$	A	6
$20(010100)_2$	9	8
$21(010101)_2$	A	10
$22(010110)_2$	A	12
$24(011000)_2$	17	16
$25(011001)_2$	A	18
$26(011010)_2$	A	20
$28(011100)_2$	A	24
A	A	A

where $\boldsymbol{H}_{cc}[k]$ is the transition matrix for the non-A states, and $\boldsymbol{p}_{cs}[k]$ is a vector of the transition probabilities from the non-A states to the A state, and $\boldsymbol{O}$ is a vector of zeros. The probability ($\boldsymbol{p}_c[k]$) of being in one of the non-A states at the kth scan can then be described by the recursion

$$\boldsymbol{p}_c^T[k] = \boldsymbol{p}_c^T[k-1]\boldsymbol{H}_{cc}[k] \tag{27}$$

$\boldsymbol{p}_c[0]$ is defined as a vector of the initial probabilities of being in non-A state. Therefore, the probability ($\rho_A[k]$) of entering the A state for the first time just at time k is

$$\rho_A[k] = \boldsymbol{p}_c^T[k-1]\boldsymbol{p}_{cs}[k] \tag{28}$$

Similarly, if let $\beta_{h_{n_s} h_{n_s}}[k] = 0$, then according to Equation (21) the last item of $\boldsymbol{p}[k]$ (i.e. $p_A[k]$) happens to be $\rho_A[k]$, which is a PMF of the stopping time. Otherwise, if $\beta_{h_{n_s} h_{n_s}}[k] = 1$, then $p_A[k]$ is a CDF of the stopping time.

Suppose the $\alpha[k]$ value does not change with k, then $\boldsymbol{H}_{cc}[k]$ also does not change with k. Thus, the following result can be further obtained by the Equation (28).

$$\begin{aligned} \rho_A[k] &= \boldsymbol{p}_c^T[0]\boldsymbol{H}_{cc}^{k-1}\boldsymbol{p}_{cs} \\ &= \boldsymbol{p}_c^T[0]\boldsymbol{V}\Lambda^{k-1}V^{-1}\boldsymbol{p}_{cs} \end{aligned} \tag{29}$$

While $\mathbf{v}$ and Λ represent eigenvector and eigenvalue matrices of $\boldsymbol{H}_{cc}$, respectively.

When the $\alpha[k]$ value changes with k, Equation (28) can be used to calculate the probability. Obviously, the size of matrix $\boldsymbol{H}_{cc}[k]$ will significantly affect the efficiency of calculation. Otherwise, when the $\alpha[k]$ value does not change with k, the result can be obtained quickly using Equation (29) when the $\boldsymbol{H}_{cc}$ scale is small. However, as the $\boldsymbol{H}_{cc}$ size becomes larger, the calculation of $\mathbf{v}$ and Λ will become more and more difficult. In short, minimizing the size of $\boldsymbol{H}_{cc}$ can improve the efficiency.

IV. ALGORITHM

Since the size of sliding window is N, the entire Markov set has a total of 2^N states. Obviously, as N increases, the set will increase in size exponentially. As mentioned in Section III, in order to improve computational efficiency, it is necessary to create the minimal-order Markov set. After construct the minimal-order Markov set and all transition states, the acquisition probability can be solved.

$0(000000)_2$	$8(001000)_2$	$16(010000)_2$	$24(011000)_2$	$32(100000)_2$	$40(101000)_2$	$48(110000)_2$	$56(111000)_2$
$1(000001)_2$	$9(001001)_2$	$17(010001)_2$	$25(011001)_2$	$33(100001)_2$	$41(101001)_2$	$49(110001)_2$	$57(111001)_2$
$2(000010)_2$	$10(001010)_2$	$18(010010)_2$	$26(011010)_2$	$34(100010)_2$	$42(101010)_2$	$50(110010)_2$	$58(111010)_2$
$3(000011)_2$	$11(001011)_2$	$19(010011)_2$	$27(011011)_2$	$35(100011)_2$	$43(101011)_2$	$51(110011)_2$	$59(111011)_2$
$4(000100)_2$	$12(001100)_2$	$20(010100)_2$	$28(011100)_2$	$36(100100)_2$	$44(101100)_2$	$52(110100)_2$	$60(111100)_2$
$5(000101)_2$	$13(001101)_2$	$21(010101)_2$	$29(011101)_2$	$37(100101)_2$	$45(101101)_2$	$53(110101)_2$	$61(111101)_2$
$6(000110)_2$	$14(001110)_2$	$22(010110)_2$	$30(011110)_2$	$38(100110)_2$	$46(101110)_2$	$54(110110)_2$	$62(111110)_2$
$7(000111)_2$	$15(001111)_2$	$23(010111)_2$	$31(011111)_2$	$39(100111)_2$	$47(101111)_2$	$55(110111)_2$	$63(111111)_2$

FIGURE 3. All original states of 4/6 sliding window.

A. CHARACTERISTICS OF THE MINIMAL-ORDER MARKOV SET

Taking 4/6 sliding window as an example, as shown in Figure (3), we can easily enumerate the entire states. And, by counting the number of bit '1', we can easily distinguish which states indicate that the target is acquired (i.e., A state, which is shaded in Figure (3)), and which are not (i.e., non-A state). The non-A state set and the A state set make up the entire original Markov set.

It is also easy to enumerate such binary sequences (prefixed by decimals) as shown below, when the system is in one of the following states, if another success result '1' is scanned later, it enters A state immediately:

$$7(000111)_2$$
$$11(001011)_2$$
$$19(010011)_2$$
$$13(001101)_2$$
$$21(010101)_2$$
$$25(011001)_2$$
$$14(001110)_2$$
$$22(010110)_2$$
$$26(011010)_2$$
$$28(011100)_2$$

Definition 1 (Adjacent state set): If a non-A state changes to A state immediately after a subsequent result of '1', we call this non-A state an adjacent state (hereafter Ad state for short). The set of all these adjacent states is called adjacent state set, represented by $J_{M,N}$.

Obviously, according to Definition (1), each adjacent state h_J must satisfy

$$\#(h_J) = M - 1, \quad h_J \in J_{M,N} \tag{30}$$

Because the adjacent state will transfer to A state immediately after a subsequent scan of '1', the highest bit of adjacent state h_J is definitely '0', that is,

$$\emptyset(h_J) < N \tag{31}$$

In addition, evidently, any non-A state that changes to A state through state transitions must first become an Ad state before it could become an A state.

Definition 2 (Primary accepting state set): When '1' is scanned, an adjacent state will immediately turn into an A state, which we call primary accepting state (hereafter PA state for short). The set of all primary accepting states is called primary accepting state set, represented by $P_{M,N}$.

It should be noted that once in a PA state, regardless of the subsequent result of '1' or '0', the system is permanently locked, i.e. the system stays in the same very state. According to the Definition (2), a one-to-one relationship exists between Ad state and PA state, and it is easy to list the PA states of 4/6 sliding window:

Ad states		*PA* states
$7(000111)_2$	$\xrightarrow{1}$	$15(001111)_2$
$11(001011)_2$	$\xrightarrow{1}$	$23(010111)_2$
$19(010011)_2$	$\xrightarrow{1}$	$39(100111)_2$
$13(001101)_2$	$\xrightarrow{1}$	$27(011011)_2$
$21(010101)_2$	$\xrightarrow{1}$	$43(101011)_2$
$25(011001)_2$	$\xrightarrow{1}$	$51(110011)_2$
$14(001110)_2$	$\xrightarrow{1}$	$29(011101)_2$
$22(010110)_2$	$\xrightarrow{1}$	$45(101101)_2$
$26(011010)_2$	$\xrightarrow{1}$	$53(110101)_2$
$28(011100)_2$	$\xrightarrow{1}$	$57(111001)_2$

FIGURE 4. A one-to-one relationship exists between an *Ad* state and a *PA* state.

Obviously, the lowest bit of the PA state h_P is always '1', and $\#(h_P) = M$. Therefore, it is easy to conclude that a PA state must be an A state, but an A state is not necessarily a PA state. For example, in the 4/6 sliding window $30(011110)_2$ must be an A state, but not a PA state.

Definition 3 (Equivalent state of M/N sliding window): If $h_1.H^+ = h_2.H^+$, and $h_1.H^- = h_2.H^-$, then these two states h_1 and h_2 can be considered equivalent, i.e. $h_1 \cong h_2$. Here, h_1, h_2 are all non-A states. The symbol $\cong$ is used to represent equivalence.

Definition (3) is self-evident. For example, as depicted in Figure (5), by definition, in the 4/6 sliding window state set, 4 and 36 are equivalent, and 9 and 41 are equivalent, and 8 and 40 are equivalent. According to Definition (3), it is easy to find that all those two states which contain the

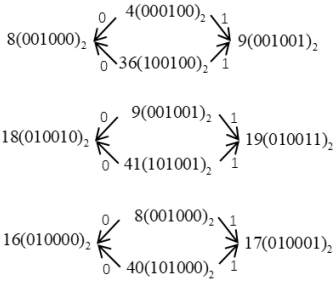

FIGURE 5. Examples of equivalent state in 4/6 sliding window according to definition (3).

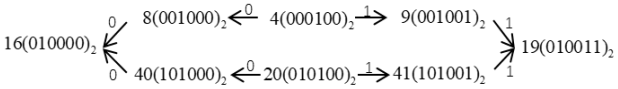

FIGURE 6. Example of equivalent states in 4/6 sliding window.

same bits except the highest bit, are equivalent to each other. This means that for the state with the highest bit being '1', an equivalent state smaller than it is sure to be found. Thus, when determining the smallest equivalent state, those states with the highest bit being '1' can be discarded directly.

Further analysis shows that if two states have the same value H^+ after scanning '1', it is easy to deduce that the transition state H^- after scanning '0' is also have the same value, because it is nothing more than replacing the lowest bit of H^+ with 0. Therefore, when determining whether the two states are equivalent, we only need to consider H^+ or H^-. Thus, we get the following inference:

Conclusion (1): If two non-A states h_1, h_1, have the same transition state H^+ after scanning '1', that is, $h_1.H^+ = h_2.H^+$, then the two states are equivalent, that is, $h_1 \cong h_2$. Similarly, if two non-A states, h_2, h_2 have the same transition state H^- after scanning '0', that is, $h_1.H^- = h_2.H^-$, then the two states are equivalent, that is, $h_1 \cong h_2$. Note that when judging equivalence, determine whether values of $h_1.H^+$ and $h_2.H^+$ are equal, rather than seeing if $h_1.H^+$ and $h_2.H^+$ can all be abstracted as A.

According to the equivalent transitivity, it can be known that, if $h_1 \cong h_2$ and $h_2 \cong h_3$, then $h_1 \cong h_3$. Now, let's have a look at a more complicated situation below, i.e. the equivalence of 4 and 20 in 4/6 sliding window Markov set as shown in the Figure (6).

As can be seen from Figure (5), since 9 and 41 are equivalent, 8 and 40 are equivalent, according to the replaceability of equivalence and Definition (3), observing Figure (6), so 4 and 20 are equivalent too. We can also analyze this by observing Figure (6). According to Conclusion (1), 9 and 41 are equivalent, and 8 and 40 are equivalent, so 4 and 20 are also equivalent. Thus, according to the equivalence transitivity, 4, 20 and 36 are equivalent. Similarly, in the case of Figure (6), for equivalence judgment, we only need to consider the half side of the diagram of the state transition. From the example in Figure (6), we can get such a general conclusion:

Conclusion (2): Given $h_1 \cdot H^+ = h_3$, $h_2 \cdot H^+ = h_4$, if $h_3 \cong h_4$ or $h_3 = h_4$, then $h_1 \cong h_2$. Here h_1, h_2 are all non-A states. Note that when h_3 or h_4 is A state, using $h_3 = h_4$ to judge the equivalence of h_1 and h_2, rather than seeing if h_3 and h_4 can all be abstracted as A.

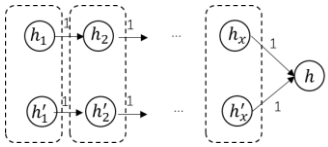

FIGURE 7. The equivalent transition.

For more general cases, as shown in Figure (7), if $\left(h_1 \overset{N,1}{<<<} x\right) = \left(h'_1 \overset{N,1}{<<<} x\right) = h$, according to Conclusion (1), we can infer that $h_x \cong h'_x$, here h_1, h'_1 are non-A states. Then, according to Conclusion (2), likewise, we can infer that $h_{x-1} \cong h'_{x-1}$ and so on and so forth, and finally we can infer that $h_1 \cong h'_1$. Since all non-A states will ultimately transfer to one of anAd states (also PA states) after several times of successful scans, hence a conclusion can be drawn:

Conclusion (3): If two or more non-A states of the M/N sliding window transferred to the same Ad state (also PA state) after successive identical scan of '1', then these states are equivalent.

According to Conclusion (3), consider the specific values r and h_J, where $0 \le r \le M$-1, $h_J \in J_{M,N}$, all non-A states that satisfy the following equation are equivalent to each other:

$$\left(h \overset{N,1}{<<<} r\right) = h_J, \text{ here } 0 \le r \le M-1, \quad h_J \in J_{M,N}$$

We call this collection the $r - J(h_J)$ equivalent state set, represented by $S_{M/N,r-J(h_J)}$.

Similarly, Conclusion (3) can also be explained by the following equation, where R(R indicates the number of left shifts required for h to change to the A state for the first time) and h_P both refer to specific values:

$$\left(h \overset{N,1}{<<<} R\right) = h_P, \text{ here } 1 \le R \le M, \quad h_P \in \boldsymbol{P}_{M,N}$$

Likewise, all non-A states that satisfy the above equation are equivalent to each other. We call such collection the $R - P(h_P)$ equivalent state set, represented by $\boldsymbol{S}_{M/N,R-P(h_P)}$.

Combining the one-to-one relationship between Ad state and PA state, it is easy to know that if $R = r+1$, and $h_p = \left(h_J \overset{N,1}{<<<} 1\right)$, then the set $\boldsymbol{S}_{M/N,r-J(h_J)}$ is the same as the set $\boldsymbol{S}_{M/N,R-P(h_P)}$, i.e. $\boldsymbol{S}_{M/N,r-J(h_J)} = \boldsymbol{S}_{M/N,R-P(h_P)}$.

Taking the example described earlier, all three equivalent states 4, 20, and 36 in 4/6 sliding window belong to $\boldsymbol{S}_{4/6,2-J(19)}$, and $\boldsymbol{S}_{4/6,2-J(19)} = \boldsymbol{S}_{4/6,3-P(39)}$.

There is a special case that for each Ad state no other equivalent states can be found, which means that $\boldsymbol{S}_{M/N,0-J(h_J)}$ (i.e. $\boldsymbol{S}_{M/N,1-P(h_P)}$) contains only one member, i.e. h_J. For example, 13 is an Ad state in the 4/6 sliding window Markov

set, i.e. it belongs to $\boldsymbol{S}_{4/6,0-J(13)}$(i.e. $\boldsymbol{S}_{4/6,1-P(27)}$), and it is the unique member of $\boldsymbol{S}_{4/6,0-J(13)}$.

As all states in $\boldsymbol{S}_{M/N,R-P(h_P)}$ are equivalent to each other, we can choose the smallest one as the representative. As $\forall h \in \boldsymbol{S}_{M/N,R-P(h_P)}$, $\left(h \overset{N,1}{<<<} R\right) = h_P(R, h_P$ are certain values, $h_P \in \boldsymbol{P}_{M,N})$ must be satisfied. Let $h_m = \left(h_P \overset{N}{>>>} R\right)$, then the only difference between h_m and h is the highest R bit(s). As all of the highest R bits of h_m must be '0', the following holds: $h_m \leq h$. In addition, obviously $h_m \in \boldsymbol{S}_{M/N,R-P(h_P)}$. Thus, h_m must be the smallest one in $\boldsymbol{S}_{M/N,R-P(h_P)}$. This can also be easily illustrated by Figure (8).

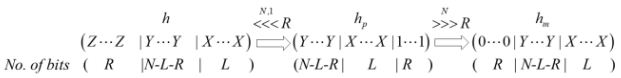

FIGURE 8. The smallest equivalent state.

Hence, this also leads to a method for calculating the smallest equivalent state of a specific $\boldsymbol{S}_{M/N,R-P(h_P)}$. Thus, all those smallest equivalent states from different $\boldsymbol{S}_{M/N,R-P(h_P)}$ make up the smallest equivalent state set, which is represented by a symbol $\boldsymbol{S}^e_{M,N}$.

For example, in 4/6 sliding window system, $23(010111)_2 \in \boldsymbol{P}_{4,6}$, let $h_P = 23(010111)_2$. Since 2, 18, 34, 50 satisfy the following equations, respectively:

$$2(000010)_2 \overset{6,1}{<<<} 3 = 23(010111)_2$$
$$18(010010)_2 \overset{6,1}{<<<} 3 = 23(010111)_2$$
$$34(100010)_2 \overset{6,1}{<<<} 3 = 23(010111)_2$$
$$50(110010)_2 \overset{6,1}{<<<} 3 = 23(010111)_2$$

So, 2, 18, 34 and 50 are equivalent to each other, and they all belong to $\boldsymbol{S}_{4/6,3-P(23)}$. And, the smallest equivalent state h_m in $\boldsymbol{S}_{4/6,3-P(23)}$ can be described as

$$h_m = (h_p \overset{6}{>>>} R) = 23(010111)_2 \overset{6}{>>>} 3 = 2(000010)_2$$

Thus, the smallest equivalent state in $\boldsymbol{S}_{4/6,3-P(23)}$ is 2.

So far, the definition of $\Omega(h)$ can also be described as follows: the number of '1's which overflow on the left during the operation $\left(h \overset{N,1}{<<<} R\right) = h_P$. According to Definition (2), it is easy to deduce the following relationship among $\#(h)$, $\Omega(h)$ and R

$$R = \Omega(h) + M - \#(h) \tag{32}$$

By continuously performing $\overset{N,1}{<<<}$ operation R times, a non-A state h will definitely become a PA state h_P, and this is also the first time when the system enters A state. As we can imagine, for those equivalent states larger than $h_m (h_m = \left(h_P \overset{N}{>>>} R\right))$, one or more '1's will definitely be overflowing from the left side of the window, which means $\Omega(h) > 0$. But such is not the case for h_m itself, which satisfies $\Omega(h_m) = 0$. Therefore, all of the smallest equivalent states must satisfy the following two conditions at the same time.

TABLE 3. The conditions that a smallest equivalent state must satisfy.

1.	For some particular i ($1 \leqslant i \leqslant M$), the highest i bits of the state are all '0'. There may or may not be '0's immediately following the leftmost i '0's;
2.	For the same i, by continuously performing $\overset{N,1}{<<<}$ operation i times, the system can turn into a PA state, which means that the (N-i) lowest bits in the original state contain (M-i) '1's.

Table (3) tells us the fact that a smallest equivalent state h must satisfy $\Omega(h) = 0$.

Instead, we can also prove the following fact:

If a non-A state satisfies $\Omega(h) = 0$, then it must be a smallest equivalent state. Proof is as follows:

According to Equation (32), $\Omega(h) = 0$ leads to $R = M - \#(h)$. It also means that by performing an $\left(h \overset{N,1}{<<<} R\right)$ operation without losing any '1's, h will transfer to PA state, i.e. $\left(h \overset{N,1}{<<<} R\right) = h_p$. Thus, we can also infer that $R \leq N - \emptyset(h)$. Since the leftmost $N - \emptyset(h)$ bits of h are all '0', naturally its highest R bits are all '0'. Therefore, h must be exactly the smallest equivalent state in $S_{M/N,R-P(h_P)}$. The proof is done.

We can also easily prove the fact that $\Omega(h) = 0$ and $N - M \geq \emptyset(h) - \#(h)$ are completely equivalent. The proof is as follows:

If $\Omega(h) = 0$, then $R = M - \#(h)$, and $R \leq N - \emptyset(h)$. Thus, $N - M \geq \emptyset(h) - \#(h)$.

Instead, if $N - M \geq \emptyset(h) - \#(h)$, i.e. $N - \emptyset(h) \geq M - \#(h)$, which means that the number of leftmost '0's is greater than or equal to the number of new '1's required for h to transit to A state, i.e. $\Omega(h) = 0$. The proof is done.

So far, we can get the following conclusion:

Conclusion (4): The necessary and sufficient condition for the non-A state h to be a smallest equivalent state is

$$N - M \geq \emptyset(h) - \#(h) \tag{33}$$

Equation (33) can also be equivalently described as

$$\Omega(h) = 0 \tag{34}$$

which means that the non-A state h can transfer to A state after $M - \#(h)$ successive scans of '1' without losing any '1's.

Note that the right-hand side of Equation (33) can be interpreted as:

$$\emptyset(h) - \#(h)$$

= Number of bits excluding leading '0's – Number of 1's
= Number of '0's to the right of the highest bit '1'.

Therefore, Equation (33) can be expounded more straightforwardly like this: if the number of '0's to the right of the

highest bit '1' is less than or equal to (N-M), then the state is a smallest equivalent state, otherwise, it can be represented by a smallest equivalent state. Therefore, we can draw the following conclusion:

Conclusion (5): The way to calculate the smallest equivalent state of a non-A state h is to count the number of '0's from the rightmost. When (N-M+1) '0's are counted, zero all the remaining uncounted bits. The newly generated binary sequence is the smallest equivalent state for the original state h.

Below, we illustrate Conclusion (5) with an example using a big state value h, i.e., calculating the smallest equivalent state of $h = 47876$ of 15/20 sliding window.

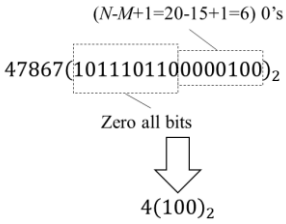

FIGURE 9. Equivalent states for $h = 47876$ in sliding window 15/20.

So, the smallest equivalent state of $h = 47867\ (1011101100000100)_2$ in the 15/20 sliding window is $h = 4\ (100)_2$.

B. SIZE OF THE MINIMAL-ORDER MARKOV SET

It is easy to get the size of the space $S^e_{M,N}$ according to Table (3)

$$Sizeof(S^e_{M,N}) = \sum_{i=1}^{M} \binom{N-i}{M-i} \tag{35}$$

Let $e = M - i$, Equation (35) is converted to

$$Sizeof(S^e_{M,N}) = \sum_{e=0}^{M-1} \binom{N-M+e}{e} = \binom{N}{M-1} \tag{36}$$

All the smallest equivalent states plus the A state (i.e. abstracted into one state) constitute the minimal Markov set. Let $\boldsymbol{S}^*_{M,N}$ be the minimal-order Markov set.

So, the size of $\boldsymbol{S}^*_{M,N}$ is

$$Sizeof(S^*_{M,N}) = \binom{N}{M-1} + 1 \tag{37}$$

When M and N are determined, the (R, h_p) value pair can uniquely determine a $\boldsymbol{S}_{M/N,R-P(h_P)}$, which means that the ($R$, h_p) value pair can uniquely determine a smallest equivalent state h_m, in which $h_m = \left(h_P >\overset{N}{>}> R\right)$, $h_p \in \boldsymbol{P}_{M,N}$. Therefore, the total number of smallest equivalent states, i.e. the size of $\boldsymbol{S}^e_{M,N}$, is equal to the total number of (R, h_p) value pairs. Since $h_p = \left(h_m <\overset{N,1}{<}< R\right)$, this means that there will always be one or more consecutive '1's on the rightmost side of h_p. Suppose that the number of consecutive '1's on the rightmost side of h_p is λ. We will prove that

$$\left(h_P >\overset{N}{>}> 1\right), \left(h_P >\overset{N}{>}> 2\right), \ldots, \left(h_P >\overset{N}{>}> R\right), \ldots, \left(h_P >\overset{N}{>}> \lambda\right),$$

i.e.,

$$\left(h_P >\overset{N}{>}> 1\right), \left(h_P >\overset{N}{>}> 2\right), \ldots, h_m, \ldots, \left(h_P >\overset{N}{>}> \lambda\right)$$

are definitely smallest equivalent states too. We use the symbol $\left(h_P >\overset{N}{>}> i\right)$ to represent each value in the sequence. Here we discuss in two situations. The first case is $i < R$, i.e., items on the left side of h_m in the sequence. In this case. '1' is continuously moved into the right side of h_m (i.e. $\left(h_P >\overset{N}{>}> R\right)$). We can find that after each insertion of '1', $\emptyset(h)$ increases by one, and $\#(h)$ increases by one too. Thus, the quantity $\emptyset(h) - \#(h)$ is remains the same like h_m's. The other case is $i > R$, i.e., items on the right side of h_m in the sequence. In this case, '1' continuously moves out of the window from the right side of h_m. Thus, we can find that after each overflow of '1', $\emptyset(h)$ decreases by one, and $\#(h)$ decreases by one too. Thus, the quantity $\emptyset(h) - \#(h)$ is also remains the same like h_m's. As we have mentioned that h_m is a smallest equivalent state, according to Conclusion (4), $\left(h_P >\overset{N}{>}> 1\right)$, $\left(h_P >\overset{N}{>}> 2\right)$,..., $\left(h_P >\overset{N}{>}> \lambda\right)$ are smallest equivalent states. Thus, when $\boldsymbol{P}_{M,N}$ is known, as depicted in Figure (11), we can get the size of $\boldsymbol{S}^e_{M,N}$ by summing all the number of consecutive '1' to the right of the lowest bit '0' in each PA state.

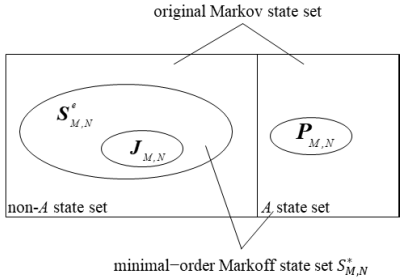

FIGURE 10. A diagram of the relationship between different sets. $\boldsymbol{J}_{M,N}$ represents Ad state set. $\boldsymbol{P}_{M,N}$ represents PA state set. $\boldsymbol{S}^e_{M,N}$ represents the smallest equivalent state set. $\boldsymbol{S}^*_{M,N}$ represents the minimal-order Markov set.

C. CREATING THE MINIMAL-ORDER MARKOV SET

In fact, Table (3) has offered an approach to listing all the smallest equivalent states: List all binary sequences that satisfy (N-i) bits containing (M-i) '1's, here $1 \leq i \leq M$, and the (N-i) bits must be the rightmost bits, and the remaining

PA states	bit '1' count
$(00\mathbf{1}111)_2$	4
$(01\mathbf{0}111)_2$	3
$(100\mathbf{1}11)_2$	3
$(0110\mathbf{1}1)_2$	2
$(1010\mathbf{1}1)_2$	2
$(1100\mathbf{1}1)_2$	2
$(01110\mathbf{1})_2$	1
$(10110\mathbf{1})_2$	1
$(11010\mathbf{1})_2$	1
$(11100\mathbf{1})_2$	1
sum:	20

FIGURE 11. The size of the minimal-order markov set (except *A* state) of 4/6 sliding window. The '0' in bold type represents the lowest bit '0'.

leftmost bits must be '0'. Let $j = M\text{-}i$, thus this description is completely equivalent to the following:

List all binary sequences that satisfy (N-M + j) bits containing j '1's, here $0 \le j \le M$-1. Here, the (N-M + j) bits must be the rightmost bits, and the remaining leftmost bits must be '0'.

Thus, its process can be simplified as Table (4).

TABLE 4. Creating the minimal-order markov set.

1.	Add A state to the set;
2.	Create all of the possible states h_j which satisfy (N-M+j) bits that contain exactly j '1's, here $0 \le j \le M$ -1, and the (N-M+j) bits must be the rightmost bits, and the remaining leftmost bits must be '0'. Add these states to the set.

Thus, Table (4) is essentially a combinatorics problem. There are various algorithms [19]–[21]. Basically, the time complexity of these algorithms is related to the binomial coefficient $O\left(\binom{N}{M-1}\right)$.

Taking the 4/6 sliding window as an example, according to Table (4), the minimal-order Markov set can be listed as follows:

				28(11100)	
				26(11010)	
				25(11001)	
				22(10110)	
			12(1100)	21(10101)	
			10(1010)	19(10011)	
			9(1001)	14(01110)	
		4(100)	6(0110)	13(01101)	
		2(010)	5(0101)	11(01011)	
	0(00)	1(001)	3(0011)	7(00111)	A
(N-M+j) bits	2	3	4	5	
Contain j '1's	0	1	2	3	

FIGURE 12. Creation of 4/6 minimal-order markov set.

D. TRANSMISSION STATES T$^+$ AND T$^-$

Now, we will discuss methods for calculating the transition states H^+ and H^- within the minimal-order Markov set. It can also be expressed as calculating the smallest equivalent states of $h.H^+$ and $h.H^-$ when $h \in S^*_{M,N}$ is known. We use symbol $h.T^+$ to represent the smallest equivalent state of $h.H^+$, and use symbol $h.T^-$ to represent the smallest equivalent state of $h.H^-$. Obviously, if h is the A state, then $h.H^+$ and $h.H^-$ are always A.

As mentioned in Table (3), the highest i bits of the smallest equivalent state must be '0', where $i \ge 1$. We can get the following conclusion:

Conclusion (6): In the minimal-order Markov set of the M/N sliding window, for any state except A state, the highest bit of them is '0'.

Thus, $\mathrm{H}^+h.H^+$ is easy to deduce, that if the weight of h ($h \in \boldsymbol{S}^e_{M,N}$) is M-1, when another '1' is inserted into the right side of the window, the system will enter A state, that is, $h.H^+ = A$; if h contains less than $M-1$ '1's, due to Conclusion (6), $h.H^+ = 2h{+}1$. Meanwhile, because satisfying Equation (33), $2h{+}1$ also belongs $\boldsymbol{S}^*_{M,N}$. Therefore, the calculation of $h.T^+$ can be represented by

$$h.T^+ = \begin{cases} A & ,\#(h) = M-1 \\ 2h+1 & ,\#(h) < M-1 \end{cases} \tag{38}$$

Since $h.H^-\left(h \in \boldsymbol{S}^e_{M,N}\right)$ is obtained by 1-bit left shift with a new rightmost '0', then according to Conclusion (6), we can get the followings

$$\emptyset(h.H^-) = \emptyset(h) + 1 \tag{39}$$

$$\#(h.H^-) = \#(h) \tag{40}$$

There are two situations:

In the first case, if $N - M > \emptyset(h) - \#(h)$, then $N - M \ge (\emptyset(h) + 1) - \#(h)$, which leads to $N - M \ge \emptyset(h.H^-) - \#(h.H^-)\mathrm{N} - \emptyset\left(\mathrm{h.H}^-\right) + \#(\mathrm{h.H}^-) \ge \mathrm{M}$. Thus, according to Equation (33), $h.H^- = 2\,h$ is also a member of $\boldsymbol{S}^*_{M,N}$.

In the second case, if $N - M = \emptyset(h) - \#(h)$, then $N - M + 1 = (\emptyset(h)+1) - \#(h)$, which leads to $N - M + 1 = \emptyset\left(h.H^-\right) - \#\left(h.H^-\right)$. So, $h.H^- = 2\,h$ is not member of $S^*_{M,N}$. In such rare situations, $2h$ happens to contain (N-M+1) '0's to the right of the highest bit '1', and this means that the highest meaningful bit '0' (see Figure (1)) happens to be the (N-M+1)th counted '0'. By observation, it is easy to know that there must be one or more consecutive '1's to the left side of the highest meaningful bit '0'. Following the steps in Conclusion (5), so long as we zero all '1's to the left of the highest meaningful bit '0' in $2h$, the newly-established sequence is the smallest equivalent state of $h.H^-$. Equally, we can also zero all '1's to the left of the highest meaningful bit '0' in h, then multiply the modified h by 2, the newly-established sequence is the smallest equivalent state of $h.H^-$.

$$h.T^- = \begin{cases} 2h & ,N - M > \emptyset(h) - \#(h) \\ 2\omega(h) & ,N - M = \emptyset(h) - \#(h) \end{cases} \tag{41}$$

The $\omega(h)$ function is defined as follows:

$\omega(h)$:= Zero all '1's to the left of the highest meaningful bit '0' in h.

An example of calculation of $\omega(h)$ is given as follows.

For example, suppose state $h = 26$ ($26 \in \boldsymbol{S}^*_{4,6}$) and its corresponding binary sequence is $(011010)_2$. As $N - M = 6 - 4 = 2$, $\emptyset(h) - \#(h) = 5 - 3 = 2$, when calculating $h.T^-$,

TABLE 5. Calculate $\omega(h)$.

```
k ⇐ ∅(h)
loop
   w ⇐ h mod 2^(k−1)
   if ( w == h || w == 0 )
   {
      return w
   }
   h ⇐ w
   k ⇐ k − 1
end loop
```

TABLE 6. Transition states within minimal-order markov set of 4/6.

h	Next state	
	T^+	T^-
0	1	0
1	3	2
2	5	4
3	7	6
4	9	0
5	11	10
6	13	12
7	A	14
9	19	2
10	21	4
11	A	22
12	25	0
13	A	26
14	A	28
19	A	6
21	A	10
22	A	12
25	A	2
26	A	4
28	A	0
A	A	A

zero all '1's to the left of the highest meaningful bit '0' in h, leaving behind $(000010)_2$, which is then multiplied by 2, thus we get $26.T^- = 4$.

Table (6) gives an example of T^+ and T^- for 4/6 sliding window.

As we have seen formula (38) and (41) are easier to implement than Williams's method.

Now let's list classification and transition state for all states under the 4/6 sliding window.

V. DISCUSSION

According to Table (4) and formulas (38) and (41), we can get a new CMMS algorithm, as shown in Table (8) below:

The biggest difference between the proposed CMMS algorithm and the Williams CMMS algorithm is that the proposed

TABLE 7. All states in 4/6 sliding window.

h	h_m [a]	$J_{4,6}$ [b]	$P_{4,6}$ [c]	$S^e_{4,6}$ [d]	Next state	
					T^+	T^-
$0(000000)_2$	0			Y	1	0
$1(000001)_2$	1			Y	3	2
$2(000010)_2$	2			Y	5	4
$3(000011)_2$	3			Y	7	6
$4(000100)_2$	4			Y	9	0
$5(000101)_2$	5			Y	11	10
$6(000110)_2$	6			Y	13	12
$7(000111)_2$	7	Y[e]		Y	A	14
$8(001000)_2$	0				1	0
$9(001001)_2$	9			Y	19	2
$10(001010)_2$	10			Y	21	4
$11(001011)_2$	11	Y		Y	A	22
$12(001100)_2$	12			Y	25	0
$13(001101)_2$	13	Y		Y	A	26
$14(001110)_2$	14	Y		Y	A	28
$15(001111)_2$	A		Y		A	A
$16(010000)_2$	0				1	0
$17(010001)_2$	1				3	2
$18(010010)_2$	2				5	4
$19(010011)_2$	19	Y		Y	A	6
$20(010100)_2$	4				9	0
$21(010101)_2$	21	Y		Y	A	10
$22(010110)_2$	22	Y		Y	A	12
$23(010111)_2$	A		Y		A	A
$24(011000)_2$	0				1	0
$25(011001)_2$	25	Y		Y	A	2
$26(011010)_2$	26	Y		Y	A	4
$27(011011)_2$	A		Y		A	A
$28(011100)_2$	28	Y		Y	A	0
$29(011101)_2$	A		Y		A	A
$30(011110)_2$	A				A	A
$31(011111)_2$	A				A	A
$32(100000)_2$	0				1	0
$33(100001)_2$	1				3	2
$34(100010)_2$	2				5	4
$35(100011)_2$	3				7	6
$36(100100)_2$	4				9	0
$37(100101)_2$	5				11	10
$38(100110)_2$	6				13	12
$39(100111)_2$	A		Y		A	A
$40(101000)_2$	0				1	0
$41(101001)_2$	9				19	2
$42(101010)_2$	10				21	4
$43(101011)_2$	A		Y		A	A
$44(101100)_2$	12				25	0
$45(101101)_2$	A		Y		A	A
$46(101110)_2$	A				A	A
$47(101111)_2$	A				A	A

TABLE 7. ***(Continued.)*** **All states in 4/6 sliding window.**

48$(110000)_2$	0				1	0
49$(110001)_2$	1				3	2
50$(110010)_2$	2				5	4
51$(110011)_2$	*A*		Y		*A*	*A*
52$(110100)_2$	4				9	0
53$(110101)_2$	*A*		Y		*A*	*A*
54$(110110)_2$	*A*				*A*	*A*
55$(110111)_2$	*A*					
56$(111000)_2$	0				1	0
57$(111001)_2$	*A*		Y		*A*	*A*
58$(111010)_2$	*A*				*A*	*A*
59$(111011)_2$	*A*				*A*	*A*
60$(111100)_2$	*A*				*A*	*A*
61$(111101)_2$	*A*				*A*	*A*
62$(111110)_2$	*A*				*A*	*A*
63$(111111)_2$	*A*				*A*	*A*

The shaded states are *A* states, and the others are non-*A* states.
[a] The smallest equivalent state of *h*.
[b] *Ad* state set of 4/6 sliding window.
[c] *PA* state set of 4/6 sliding window.
[d] The smallest equivalent state set of 4/6 sliding window.
[e] 'Y' indicates that the state is contained in the set.

TABLE 8. **New CMMS algorithm.**

1. Add *A* state to the set;
2. Create all of the possible states h_j which satisfy (*N*-*M*+*j*) bits that contain exactly *j* '1's, here $0 \le j \le M$ -1, and the (*N*-*M*+*j*) bits must be the rightmost bits, and the remaining leftmost bits must be '0'. Add these states to the set.
3. According to formula (38) and (41), calculate T^+ and T^- of each state in the set.

method does not need to calculate H^+ and H^-, but to directly calculate T^+ and T^-. While the Williams CMMS algorithm always calculates H^+ and H^- first, then calculates the T^+ and T^- of them.

In the time complexity analysis, we consider the calculation of T^+ (H^+) and T^- (H^-) as a calculation unit, and we call this calculation unit an T_c unit. The Williams CMMS algorithm needs to calculate the H^+ and H^- of each smallest equivalent state first, and the number of H^+ and H^- is almost proportional to the size of the original Markov state set, i.e. 2^N. Therefore, the calculation number of T_c unit is also proportional to the size of the original Markov state set. Thus, the time complexity of Williams CMMS algorithm is $O(2^N)$. It is easy to know that the calculation number of T_c unit of the proposed CMMS algorithm is proportional to the size of minimal-order Markov set. Therefore, the time complexity of the proposed CMMS algorithm is $O\left(\binom{N}{M-1}\right)$. The following diagram is a comparison of the time complexity between the two.

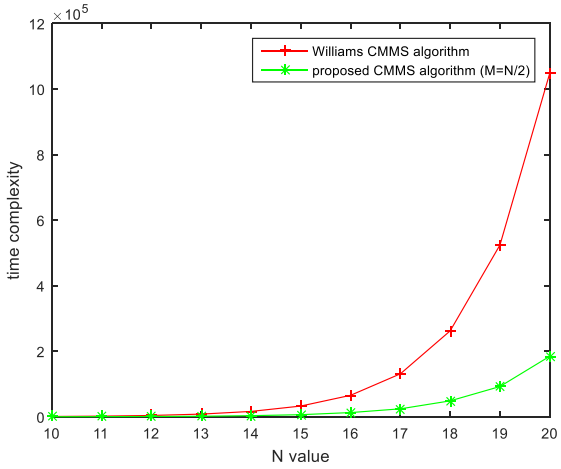

FIGURE 13. **A comparison of time complexity of CMMS algorithm.**

TABLE 9. **Run time of CMMS.**

M/*N*	time consumed (s)	
	Williams CMMS algorithm	proposed CMMS algorithm
5/10	0.194	0.083
5/11	0.268	0.152
5/12	0.424	0.184
5/13	0.722	0.268
5/14	1.262	0.347
5/15	2.323	0.482
5/16	4.846	0.712
5/17	9.559	0.949
5/18	20.099	1.181
5/19	41.262	1.443
5/20	88.093	1.843

Figure (13) is a theoretical comparison. Table (9) gives the run time in the experimental environment. The experimental environment is as follows:

Computer: DELL PRECISION TOWER 5810
System: Windows 10 Enterprise Edition
CPU: Intel Xeon CPU E5-1620
RAM: 32G
Language: MATLAB script

Figure (14) is a graphical representation of the data in Table (9). It gives a comparison of time consumption under two different methods. Figure (14) and Figure (13) are very similar, indicating that the actual conclusions are consistent with the theoretical analysis. Obviously, the method proposed in this paper has less time consumption. When $M/N = 5/20$, the new method only takes 1/48 of that of the Williams CMMS algorithm. It is obvious to see that the proposed CMMS algorithm is more efficient than Williams CMMS algorithm.

From equations (38) and (41), the time complexity of calculating $h.T^+$ is $O(1)$, and the time complexity of calculating $h.T^-$ is $O(1)$ or $O(N)$. In [17], the recursive method is used to calculate the smallest equivalent transition states, and its time

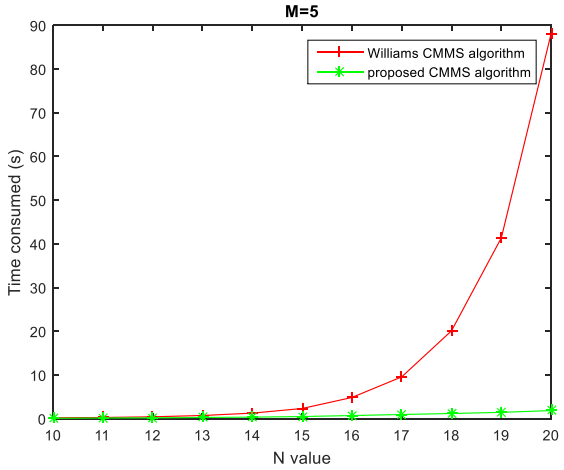

FIGURE 14. A comparison of time consumption of CMMS algorithm.

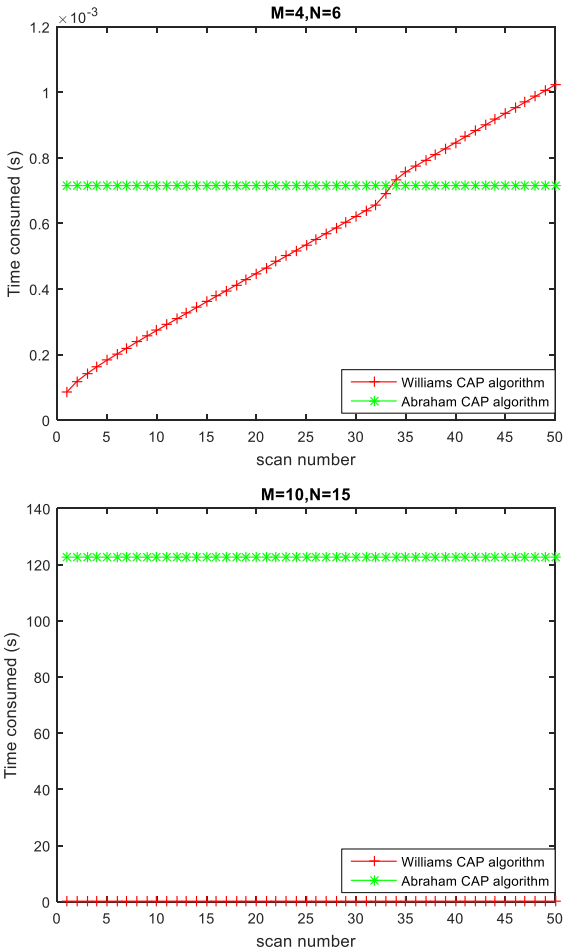

FIGURE 15. Comparisons of time consumption of CAP algorithm. $\alpha = 0.8$.

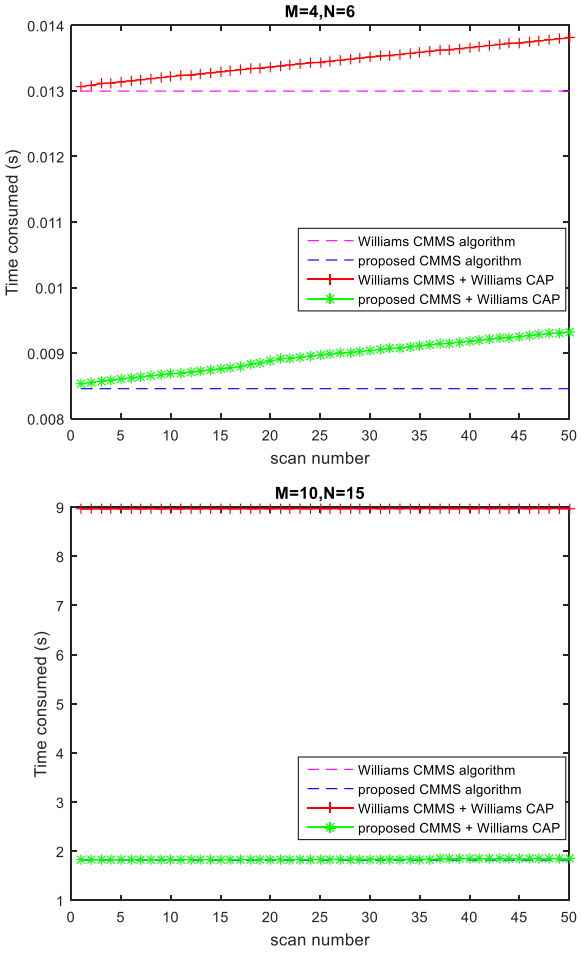

FIGURE 16. Comparisons of time consumption of the whole algorithm. $\alpha = 0.8$.

complexity is $O(N)$. Therefore, from the perspective of time complexity, the proposed method for calculating the smallest equivalent states in this paper is more efficient.

Two pictures in Figure (15) give the time consumption comparison between Williams CAP algorithm and Abraham CAP algorithm. Here, we do not consider the use of historical data, which means that Williams CAP algorithm calculates the probability each time from the first scan, and also means that the eigenvalues and eigenvectors of the transition matrix in Abraham CAP algorithm are calculated each time, regardless of whether the scan number (i.e. x-axis value in the figure) change or not. The first figure in Figure (15) is a comparison of the two methods applied to a test when the size of the minimal-order Markov set is small, which also means that the transition matrix is a small matrix. The second figure in Figure (15) is a comparison of the two methods applied to a test when the size of the minimal-order Markov set is large, which also means that the transition matrix is a large matrix. Since the acquisition probability in the Abraham CAP algorithm is directly solved by the formula, the running time under different scan number is the same when the N and M values are determined, which can be seen from both figures in Figure (15). As can be seen from the first figure in Figure (15), the run time of Williams CAP algorithm increases as the number of scans increases. This is because calculating acquisition probability at time k needs to first calculate the acquisition probability at time $k-1$. The Abraham CAP algorithm needs to calculate the eigenvalues and eigenvectors of the transition matrix (please refer to

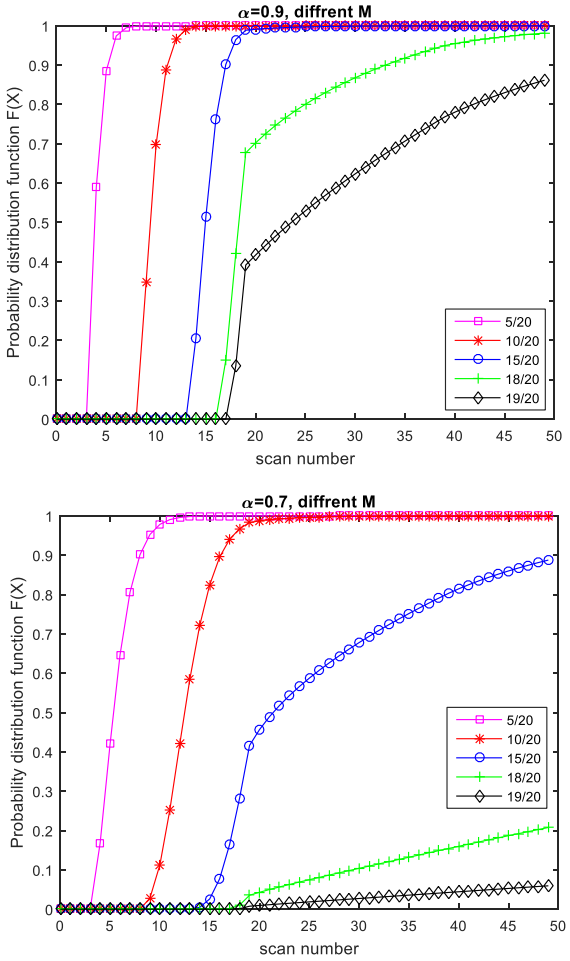

FIGURE 17. The acquisition probability distributions. The value of *N* and α are fixed and the value of *M* changes.

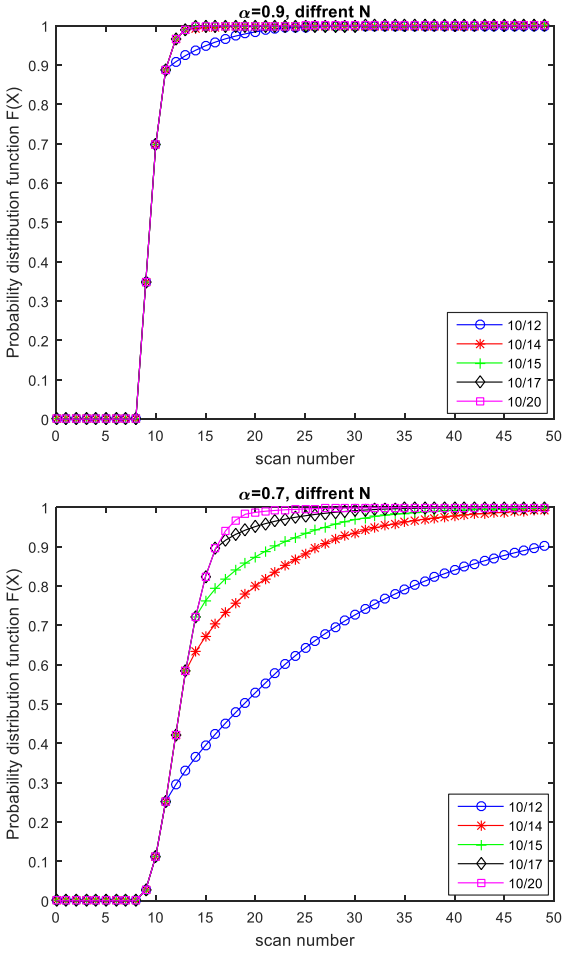

FIGURE 18. The acquisition probability distributions. The value of *M* and α are fixed and the value of *N* changes.

Equation (29)) in the process of CAP. The calculation of the eigenvalues and eigenvectors is time consuming in the case of a large matrix. Therefore, it can be seen from the second figure in Figure (15) that the Abraham CAP algorithm runs significantly less efficiently than the Williams CAP algorithm when the transition matrix becomes larger (i.e. the size of the minimal-order Markov set becomes larger).

Figure (16) shows that the time consumed by CMMS algorithm takes a larger proportion that of the whole algorithm (i.e. CMMS + CAP). The graph of CMMS line in the second figure of Figure (16) almost overlaps that of the corresponding CMMS + CAP. These two graphs can well illustrate that the proposed CMMS algorithm can obviously improve the performance of the whole acquisition probability algorithm. Of course, when M and N remain unchanged, the CMMS algorithm needs to be executed only once.

The relevant MATLAB script, including the proposed CMMS algorithm and transition states calculation, can be downloaded from https://github.com/zk47/mofn.git.

VI. CONCLUSION

In this paper, through theoretical analysis, we have drawn some conclusions on the minimal-order Markov set for M/N sliding window. Using Conclusion (4), one can quickly determine whether a state is a smallest equivalent state. Using Conclusion (5), a non-A state can be quickly converted to its smallest equivalent state. In light of these conclusions, we propose a new algorithm for CMMS, i.e., Table (8).

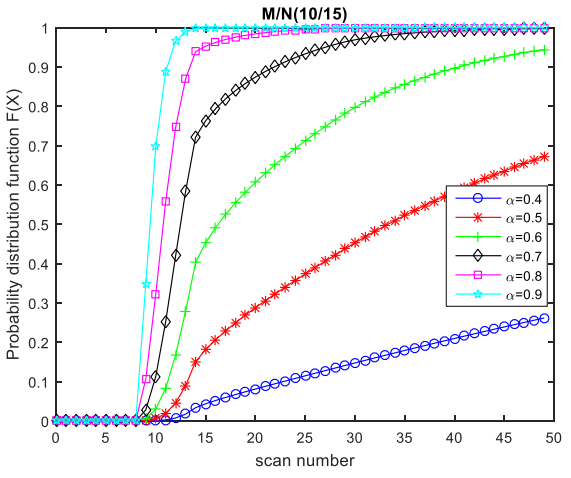

FIGURE 19. The acquisition probability distributions. The value of *M and N are* fixed and the value of α changes.

Comparing this algorithm with Williams approach, we can see that the proposed algorithm greatly reduces the time complexity, which sharply improves the computational efficiency of CMMS process. We also propose formulas (38) and (41) for calculating the transition states T^+ and T^-, which are easier to implement and more efficient.

APPENDIX

The results of some acquisition probabilities are given in Figures (17), (18), and (19). We assume that each time the probability of successful scan is a constant α.

REFERENCES

[1] R. C. Bollinger and A. A. Salvia, "Consecutive-*k*-out-of-*n*: F networks," *IEEE Trans. Rel.*, vol. R-31, no. 1, pp. 53–56, Apr. 1982.
[2] C. Derman, G. J. Lieberman, and S. M. Ross, "On the consecutive-*k*-of-*n*: F System," *IEEE Trans. Rel.*, vol. R-31, no. 1, pp. 57–63, Apr. 1982.
[3] R. C. Bollinger, "Direct computation for consecutive-*k*-out-of-*n*: F systems," *IEEE Trans. Rel.*, vol. R-31, no. 5, pp. 444–446, Dec. 1982.
[4] C. Jun, "Reliability of a large consecutive-k-out-of-r-from-n: F system with unequal component-reliability," *IEEE Trans. Rel.*, vol. 43, no. 1, pp. 107–111, Mar. 1994.
[5] W.-C. Yeh, "A simple algorithm for evaluating the *k*-out-of-*n* network reliability," *Rel. Eng. Syst. Saf.*, vol. 83, no. 1, pp. 93–101, Jan. 2004.
[6] G. Levitin and Y. Dai, "*k*-out-of-*n* sliding window systems," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 42, no. 3, pp. 707–714, May 2012.
[7] A. R. Kamat, "Continuous inspection schemes," *Biometrika*, vol. 41, nos. 1–2, pp. 100–115, Apr. 1954.
[8] G. Dillard, "A moving-window detector for binary integration," *IEEE Trans. Inf. Theory*, vol. IT-13, no. 1, pp. 2–6, Jan. 1967.
[9] J. Harrington, "An analysis of the detection of repeated signals in noise by binary integration," *IRE Trans. Inf. Theory*, vol. 1, no. 1, pp. 1–9, Mar. 1955.
[10] R. Sittler, "Systems analysis of discrete Markov processes," *IRE Trans. Circuit Theory*, vol. 3, no. 4, pp. 257–266, Dec. 1956.
[11] G. Sponsler, "First-order Markov process representation of binary radar data sequences," *IRE Trans. Inf. Theory*, vol. 3, no. 1, pp. 56–64, Mar. 1957.
[12] R. Worsham, "The probabilities of track initiation and loss using a sliding window for track acquisition," in *Proc. IEEE Radar Conf.*, May 2010, pp. 1270–1275.
[13] J. B. Nelson, "Minimal-order models for false-alarm calculations on sliding windows," *IEEE Trans. Aerosp. Electron. Syst.*, vol. AES-14, no. 2, pp. 351–363, Mar. 1978.
[14] F. R. Castella, "Sliding window detection probabilities," *IEEE Trans. Aerosp. Electron. Syst.*, vol. AES-12, no. 6, pp. 815–819, Nov. 1976.
[15] M. P. Fewell, J. M. Thredgold, and D. J. Kershaw, "Benefits of sharing detections for networked track initiation in anti-submarine warfare," DSTO, Edinburgh, Australia, Tech. Rep. DSTO-TR-2086, Jan. 2008.
[16] P. H. Todd, "Direct minimal-order Markov model for sliding-window detection probabilities," *IEE Proc. F Commun., Radar Signal Process.*, vol. 128, no. 3, pp. 152–154, Jun. 1981.
[17] P. Williams, "Evaluating the state probabilities of m out of n sliding window detectors," DSTO, Pyrmont, New South Wales, Australia, Tech. Rep. DSTO-TN-0132, Feb. 1998.
[18] D. A. Abraham, "Analysis and design of sliding m-of-n detectors," DSTO, Ellicott, MD, USA, Tech. Rep. 2011-01, Oct. 10, 2011.
[19] R. A. Brualdi, "Generating combinations," in *Introductory Combinatorics*, 5th ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2009, pp. 98–109.
[20] G. Ehrlich, "Loopless algorithms for generating permutations, combinations, and other combinatorial configurations," *J. ACM*, vol. 20, no. 3, pp. 500–513, Jul. 1973.
[21] F. Ruskey and A. Williams, "The coolest way to generate combinations," *Discrete Math.*, vol. 309, no. 17, pp. 5305–5320, Sep. 2009.

KANG ZHAO received the B.S. degree in communication engineering from Southwest Jiaotong University, Chengdu, China, in 2003, and the M.S. degree in communication engineering from the University of Electronic Science and Technology of China. He is currently pursuing the Ph.D. degree in software engineering with the East China Normal University, Shanghai, China.

From 2006 to 2015, he was a Research Assistant with the Chinese Academy of Sciences, Shanghai, China. He is currently with Shanghai IOT Company Ltd. His research interests include the development of the Internet of Things, indoor high-precision positioning techniques, and embedded systems.

MINGHUA ZHU received the Ph.D. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2002. He is currently a Professor with the School of Computer Science and Software Engineering, East China Normal University, Shanghai, China. His research interests include the Internet of Things, embedded systems, and artificial intelligence.

XUGUANG YANG received the B.S. degree in electronic information engineering from Jilin University, China, in 2004, and the M.S. and Ph.D. degrees from the Chinese Academy of Sciences. He is currently with Shanghai IOT Company Ltd. His research interest includes the development of indoor high-precision positioning products. The positioning products are widely used in warehousing, logistics, judiciary, robotics, UAV, and other fields.

JIAONAN ZHANG received the B.S. degree in marine biology from Xiamen University, Xiamen, China, in 1999. He is a Senior Engineer, and is currently the Technical Director of the Fujian Tianma Science and Technology Group Company Ltd. His research interests include production management informatization and so on.