Neural, Parallel, and Scientific Computations 22 (2014) 461-484

SKELETONIZATION OF BINARY IMAGE IN IMMUNECOMPUTING PARADIGM AND ITS ANALYSIS

SRIMANTA PAL

Electronics and Communication Sciences Unit, Indian Statistical Institute
203 B T Road, Kolkata 700 108, India
srimanta@isical.ac.in, srimanta59@gmail.com

ABSTRACT. This paper presents a template matching thinning process for skeletonization of a binary edged image and it is viewed as an analogous process of negative selection method in immunecomputing paradigm. This immunocomputing based skeletonization process is also viewed as a Markov process in probability theory. The mathematical formulation of this process helps to study the performance of template matching thinning algorithm for sketelonization in image processing. The objective of this analysis is to measure the time requirement by the algorithm and also determine the bound on the number of iterations required for the convergence of the process of sketelonization.

AMS (MOS) Subject Classification. 94A08, 62H35, 01-08, 68T45, 68U10, 78A70, 92-08, 92B20, 92C10, 06E25.

1. Introduction

Skeletonization is an important task to perform during low-level segmentation in image processing/computer vision. The main objective of a skeletonization process is to reduce the insignificant features by elimination of redundant information. The process of skeletonization is popularly termed as thinning in image processing. The objective of this work is to reduce the storage requirement of image data.

Since 1992, Haralick [20] encouraged to the community of image analysis/machine vision to design a model for the determination of the performance of image analysis algorithms. This is an awful state of affairs for the engineers whose job is to design and build image analysis or machine vision systems. Study of performance analysis of thinning algorithms [7, 20, 21, 22, 26, 27, 28, 34, 35, 36, 41, 50] have started from last few decades. The experimental analysis of thinning algorithms have been studied by Chen & Hsu [7] and Heydorn & Weidner [21]. Jang & Chin [27] studied formally the behavior of thinning algorithms using mathematical morphology. Pal & Bhattacharyya [36] studied the average behavior of template matching thinning algorithms using a probabilistic urn model.

Received June 15, 2014

Different kinds of thinning algorithms have been proposed in the literature of image processing such as sequential, parallel [1, 4, 6, 8, 9, 11, 17, 18, 19, 25, 26, 49, 50] or neural [12, 13, 14, 29, 42, 43] algorithms with one-pass [6, 7, 11, 26, 37, 50] or multi-pass [8, 18, 24, 25] on binary [13, 16, 17, 46], gray level or colour images [2, 48]. These thinning algorithms can be classified as template matching and non-template matching thinning algorithms. The main criteria for any skeletonization algorithms are (i) connectivity and shape preservation as well as (ii) one-pixel width. Also thinning algorithm can be designed based on the type of image used such as binary image, gray level image, colour image, 3D image, range image, etc. In this paper, only the template matching thinning algorithms are considered for binary edged image and these are to be studied in immunecomputing paradigm.

This article is organized as follows. Section 2 briefly describes the *negative selection* method as a novel computational technique of *artificial immune systems* (AIS). Section 3 contains the template matching thinning process. Section 4 deals with the model for thinning process in immunocomputing paradigm. Section 5 discuss the analysis of the immunocomputing based thinning algorithm using Markov process. The conclusion and results are discussed in Section 6.

2. **Artificial Immune System**

The biologically inspired computing techniques such as neurocomputing, evolutionary computing, DNA computing, etc. are growing rapidly. On the other hand, there is a rapid increase of comprehension of the behaves of immune system (IS). The knowledge about the IS functioning has disclosed several of its main operative mechanism, *negative* and/or *positive selection* is one of them [5].

The main task of an immune system is to perform the searching task in the body of living being for malfunctioning cells (e.g., cells for cancer and tumor) from their own bodies, and foreign disease causing elements (e.g., viruses and bacteria). Every element that can be recognized by the immune system is called an *antigen* (Ag). The cells that originally belong to our body and are harmless to its functioning are termed as *self* (or *self antigens*), while the disease causing elements are named as *nonself* (or *nonself antigens*). So the immune system has to be capable of distinguishing between *self* and *nonself*. This process is called *self/nonself* discrimination. Basically this process performed as pattern recognition tasks.

In perspective of *artificial immune systems* first determine the set of pattern to be protected and name it as the *self-set* (**P**). Based upon the *negative* or *positive selection* algorithm, generate a set of *detector* (**M**) that will be responsible to identify all the elements that do not belong (or belong) to the *self-set*, i.e., the *nonself* (or *self*) elements. The negative nature will allowed to mature only those detectors that can identify elements not belonging to *self-set* justifies the name *negative selection*. The

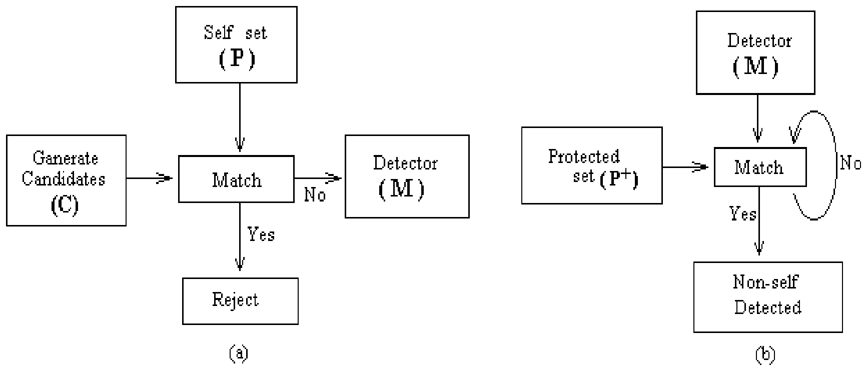

FIGURE 1. Schematic diagram for the functioning of negative selection algorithm: (a) Production of detectors, (b) Monitoring for the presence of undesired (*nonself*) pattern.

process of negative selection is shown in Fig. 1 and its working principle is described in Algorithm 1.

Algorithm 1: Negative selection method of artificial immunecomputing.

```
Input:    Set P and C
Output:   Set M
Step 1.   [Generation] Generate random candidate elements (C) which are stored
          afterwards to form detectors.
Step 2.   [Matching] Compare (match) each element in C with the elements in P.
          If an element of P is recognized by an element of C (match occurred),
          then discard this element of C else store this element of C in the detector
          set M.
Step 3.   [Loop] Repeat Step 2 until no matching takes place.
Step 4.   [Termination] Stop.
```

The newly formed detector set (**M**) in used in monitoring the system for the presence of *self*/*nonself* patterns. This set to be monitored ($\mathbf{P}^+$) might be composed of the set (**P**) plus other new patterns ($\mathbf{P}^+ \subseteq \mathbf{P}$), or it can be completely novel set ($\mathbf{P}^+ = \mathbf{P}$).

3. Skeletonization Process

In this section we consider only template matching thinning techniques for the skeletonization of binary edged image.

Consider a window of a binary edged image as shown in Fig. 2. The neighbors of the element P_1 are P_2, P_3, P_4, P_5, P_6, P_7, P_8, and P_9 where each P_i (for $i = 1, 2, \ldots, 9$) is a Boolean variable with values 0 (i.e., false) or 1 (i.e., true) and NOT P_i is denoted by $\overline{P}_i$. They represent a 3×3 window (inner) with 8-neighbors of P_1. A 5×5 window (whole) with 24-neighbors of P_1 is also shown in Fig. 2.

In the process of thinning of binary images, the new value given to a point at the nth iteration depends on its own value as well as those of its neighbors at the $(n-1)$th iteration.

P_{99}	P_{9j}	P_{2j}	P_{3j}	P_{33}
P_{9i}	P_9	P_2	P_3	P_{3i}
P_{8i}	P_8	P_1	P_4	P_{4i}
P_{7i}	P_7	P_6	P_5	P_{5i}
P_{77}	P_{7j}	P_{6j}	P_{5j}	P_{55}

FIGURE 2. Sliding window or templets of size 5×5 in which P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8, and P_9 forms a 3×3 window.

Here we describe a binary thinning algorithm with 5×5 templates. A vertical stroke of width 2 (i.e., a 2-stroke) is guarded by keeping one of its edges. So a point on a west edge is preserved if it is not on a corner and its east neighbor is on an edge, i.e., if ($f(P_4)$ AND P_2 AND P_6) is true (i.e., $f(P_4).P_2.P_6 = 1$ or simply $P_2P_6f(P_4) = 1$). Here the Boolean function f checks whether there is no more than one $'01'$ pattern and at least one $'00'$ and $'11'$ pattern of the sequence of 8-neighbors of its argument and $'AND'$ (i.e., $'.'$ or $''$) is the Boolean $'AND'$ operation in Boolean algebra. Mathematically, the Boolean function f is defined as
$f(P_1) = P_1.t_{00}.t_{11}.t_{01s}$ or $f(P_1) = P_1t_{00}t_{11}t_{01s}$
where

$$t_{00} = \begin{cases} 1, & \text{if } '00' \text{ pattern found in } P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9, P_2 \\ 0, & \text{otherwise} \end{cases}$$

i.e., $t_{00} = \overline{P}_2(\overline{P}_9 + \overline{P}_3) + \overline{P}_4(\overline{P}_3 + \overline{P}_5) + \overline{P}_6(\overline{P}_5 + \overline{P}_7) + \overline{P}_8(\overline{P}_7 + \overline{P}_9)$

$$t_{11} = \begin{cases} 1, & \text{if } '11' \text{ pattern found in } P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9, P_2 \\ 0, & \text{otherwise} \end{cases}$$

i.e., $t_{11} = P_2(P_9 + P_3) + P_4(P_3 + P_5) + P_6(P_5 + P_7) + P_8(P_7 + P_9)$

$$t_{01s} = \begin{cases} 1, & \text{if more than one } '01' \text{ pattern found in } P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9, P_2 \\ 0, & \text{otherwise} \end{cases}$$

i.e., $t_{01s} = AB + AC + AD + BC + BD + CD$
for $A = \overline{P}_9P_2 + \overline{P}_2P_3$, $B = \overline{P}_3P_4 + \overline{P}_4P_5$, $C = \overline{P}_9P_8 + \overline{P}_8P_7$, and $D = \overline{P}_5P_6 + \overline{P}_6P_7$
Similarly, the north edge of a horizontal 2-stroke is preserved if the expression ($f(P_6)$ AND P_4 AND P_8) is true (i.e., $P_4P_8f(P_6) = 1$).

The removal of each point of a 2×2 square pattern can be prevented by checking the east, south and south-east neighbors. If each of these is on a corner, the expression ($f(P_4)$ AND $f(P_5)$ AND $f(P_6)$) is true, (i.e., $f(P_4)f(P_5)f(P_6) = 1$) and the point is to be preserved.

The preservation of horizontal and vertical straight lines can be done if a Boolean function f_0 is true, where f_0 indicates a matching of the templates shown in Fig. 3.

Mathematically f_0 is defined as

$$\begin{aligned} f_0(P_1) \;=\; & P_1P_2P_4P_6P_8(\overline{P}_3P_5P_7P_9P_{77}P_{8i}P_{9i}P_{5j}P_{6j} + P_3\overline{P}_5P_7P_9P_{99}P_{7i}P_{8i}P_{2j}P_{3j}+ \\ & P_3P_5\overline{P}_7P_9P_{33}P_{5i}P_{9i}P_{2j}P_{9j} + P_3P_5P_7\overline{P}_9P_{55}P_{3i}P_{4i}P_{6j}P_{7j}) \end{aligned}$$

$$\begin{array}{ccccc} x & x & x & x & x \\ 1 & 1 & 1 & 0 & x \\ 1 & 1 & 1 & 1 & x \\ x & 1 & 1 & 1 & x \\ 1 & x & 1 & 1 & x \\ & & \text{(a)} & & \end{array} \quad \begin{array}{ccccc} 1 & x & 1 & 1 & x \\ x & 1 & 1 & 1 & x \\ 1 & 1 & 1 & 1 & x \\ 1 & 1 & 1 & 0 & x \\ x & x & x & x & x \\ & & \text{(b)} & & \end{array} \quad \begin{array}{ccccc} x & 1 & 1 & x & 1 \\ x & 1 & 1 & 1 & x \\ x & 1 & 1 & 1 & 1 \\ x & 0 & 1 & 1 & 1 \\ x & x & x & x & x \\ & & \text{(c)} & & \end{array} \quad \begin{array}{ccccc} x & x & x & x & x \\ x & 0 & 1 & 1 & 1 \\ x & 1 & 1 & 1 & 1 \\ x & 1 & 1 & 1 & x \\ x & 1 & 1 & x & 1 \\ & & \text{(d)} & & \end{array}$$

FIGURE 3. Horizontal and vertical straight line preserving templates, where x = don't care term.

An element P_1 survives after an iteration if the combined conditions represented by the assignment relation

$$\begin{aligned} P_1 \;\leftarrow\; & (P_1 \; AND \; ((\; NOT \; f(P_1)) \; OR \; (P_2 \; AND \; P_6 \; AND \; f(P_4)) \; OR \\ & (P_4 \; AND \; P_8 \; AND \; f(P_6)) \; OR \; (f(P_4) \; AND \; f(P_5) \; AND \; f(P_6))) \\ & \; AND \; (\; NOT \; f_0(P_1))) \end{aligned}$$

i.e., $P_1 \;\leftarrow\; P_1(\overline{f}(P_1) + P_2P_6f(P_4) + P_4P_8f(P_6) + f(P_4)f(P_5)f(P_6))\overline{f}_0(P_1)$

The iterative formula for updation of P_1 is given below.

$P_1^{(t+1)} \leftarrow P_1^{(t)}\left(\overline{f}(P_1^{(t)}) + P_2^{(t)}P_6^{(t)}f(P_4^{(t)}) + P_4^{(t)}P_8^{(t)}f(P_6^{(t)}) + f(P_4^{(t)})f(P_5^{(t)})f(P_6^{(t)})\right)\overline{f}_0(P_1^{(t)})$

The skeletonization process is described in Algorithm 2.

Algorithm 2: (**One-pass template matching binary thinning algorithm**) This thinning process accepts a binary edged image and produces a shape and connectivity preserving one-pixel width binary image as a skeleton. In this algorithm a 5×5 window (Fig. 2) is applied on the given binary image in left-to-right and subsequently top-to-bottom fashion. At every position of the window, it is checked whether the central pixel (P_1) is to be transformed from 1 to 0.

Input: Binary edged image, I_0.

Output: Binary thinned image, I_T.

Step 1: [**Input**] Get a binary edged image I_0.

Step 2: [**Initialization**] Number of iteration, $t = 0$ and $I_t = I_0$

Step 3: [**Get next pixel**] Get a pixel $P_1^{(t)}$ of the binary edged image I_t and its neighbours based on the size of the sliding windows where the input pixel $P_1^{(t)}$ is the central element of the window.

Step 4: [**Existence of edge of the central pixel**] In a binary edged image I_t (from previous iteration), if edge of the central pixel does not exist then do not transform the central pixel from 1 to 0 (i.e., $P_1^{(t)}$ remains unchanged, i.e., $P_1^{(t+1)} \leftarrow P_1^{(t)}$ in I_{t+1}) and repeat from Step 3 by sliding the window; otherwise, goto Step 5.

Step 5: [**Guard the vertical stroke of width 2**] Compute the edge of east neighbor ($P_4^{(t)}$) and if it satisfies the following conditions:
(1) edge of east ($P_4^{(t)}$) neighbor of $P_1^{(t)}$ exists,
(2) value of north ($P_2^{(t)}$) neighbor of $P_1^{(t)}$ is 1, and
(3) value of south ($P_6^{(t)}$) neighbor of $P_1^{(t)}$ is 1
then do not transform the central pixel from 1 to 0 (i.e., $P_1^{(t)}$ remains unchanged, i.e., $P_1^{(t+1)} \leftarrow P_1^{(t)}$ in I_{t+1}) and goto Step 3 after sliding the window; otherwise, goto Step 6.

Step 6: [**Guard the horizontal stroke of width 2**] Compute the edge of the south neighbor ($P_6^{(t)}$) and if it satisfies the following conditions:
(1) edge of south ($P_6^{(t)}$) neighbor exists,
(2) value of west pixel ($P_8^{(t)}$) is 1, and
(3) value of east ($P_4^{(t)}$) pixel is 1
then do not transform the central pixel from 1 to 0 and goto Step 3 after sliding the window; otherwise, goto Step 7.

Step 7: [**Prevention of** 2×2 **square**] Compute the edges as mentioned below.
(1) edge of east ($P_4^{(t)}$) neighbor,
(2) edge of south ($P_5^{(t)}$) neighbor, and
(3) edge of south-east ($P_6^{(t)}$) neighbor
if each of these edges exists then do not transform the central pixel to 0; otherwise, goto Step 3 after sliding the window.

Step 8: [**Loop**] Repeat the process from Step 3 through Step 7 until no further transformation takes place.

Step 9: [**Preservation of Vertical and Horizontal Lines**] If the 5×5 image window matches with the horizontal and vertical line preserving templets then transform the central pixel of the window to 0; otherwise, goto Step 4 after sliding the window by one position.

Step 10: [**Loop**] $t \leftarrow t+1$ Repeat the process from Step 3 through Step 9 until no further transformation takes place.

Step 11: [**Termination**] Stop.

Note: The steps 3-7 can be combined by the recursive rule:

$$P_1^{(t+1)} \leftarrow P_1^{(t)} \left(\overline{f}(P_1^{(t)}) + P_2^{(t)} P_6^{(t)} f(P_4^{(t)}) + P_4^{(t)} P_8^{(t)} f(P_6^{(t)}) + f(P_4^{(t)}) f(P_5^{(t)}) f(P_6^{(t)})\right) \overline{f}_0(P_1^{(t)})$$

3.1. **Stare-Case Elimination and Trimming.** The following rules are used for stare-case elimination and trimming.

(1) North-West bias is eliminated by the following rule:

$$P_1 \leftarrow P_1\overline{P_2\overline{P}_3P_4\overline{P}_7\overline{P_6P_8}P_2\overline{P}_5P_8\overline{P}_9\overline{P_4P_6}}$$

(2) South-East bias is eliminated by the following rule:

$$P_1 \leftarrow P_1\overline{P_4\overline{P}_5P_6\overline{P}_9\overline{P_2P_8}P_3\overline{P}_6P_7\overline{P}_8\overline{P_2P_4}}$$

(3) Trimming can perform by the following rule:

$$\begin{aligned} P_1 \leftarrow\ & P_1(\overline{P}_2\overline{P}_3\overline{P}_4P_6\overline{P}_8\overline{P}_9+\overline{P}_2P_4\overline{P}_6\overline{P}_7\overline{P}_8\overline{P}_9+P_2\overline{P}_4\overline{P}_5\overline{P}_6\overline{P}_7\overline{P}_8+\\ & \overline{P}_2\overline{P}_3\overline{P}_4\overline{P}_5\overline{P}_6\overline{P}_8+\overline{P}_2\overline{P}_4\overline{P}_6\overline{P}_8(\overline{P}_3\overline{P}_9(P_5\bigoplus P_7)+(\overline{P}_5\overline{P}_7(P_3\bigoplus P_9))) \end{aligned}$$

4. The Model

4.1. **Philosophy of the Model.** The *template matching* process over a binary edged image is to choose binary pixels (i.e., 1's) such that the skeleton of the given binary edged image is one-pixel width after preserving the connectivity of the binary image.

The idea of skeletonization using immunecomputing paradigm results from the interesting fact that, when a part of the body is cut by an accident and an operation takes place, the immune system tries to repair the affected part of the body in such a way that it damage a least linear part, i.e., a thin line after curing it.

4.2. **Formulation.** The skeleton of a binary edged image by a template matching thinning process is a skeleton of one-pixel width and also preserve the connectivity.

Let $C = \{X_1, X_2, \ldots, X_n\}$ be a set of n points (i.e., number of 1's) in an edged binary image of size $N = r \times c$ where $r =$ number of rows, $c =$ number of columns of it, $X_i = \{x_i, y_i, \mathbf{w}_i\}$, where $x_i \in \{1, 2, \ldots, r\}$, $y_i \in \{1, 2, \ldots, c\}$, $\mathbf{w}_i = (b_{i_1}, b_{i_2}, \ldots, b_{i_k})$, $b_{i_j} \in \{0, 1\}$, $j = 1, 2, \ldots, k$, $k =$ number of entries in the template used in the thinning process (e.g., $k = 9$ means template size is 3×3) for $i = 1, 2, \ldots, n_0$, and $n_0 =$ number of 1 in the edged image. Assume $P = \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m\}$ is a set of templates used in the thinning process where $m =$ number of different templates used in the thinning process. The problem is to find an one-pixel width skeleton from the given binary edged image such that it preserve the connectivity and the shape of the given binary edged image.

According to the theory of *artificial immune system* (AIS), the self-set P (Fig. 1) is the set of templates (Set T in Fig. 4) used in the thinning process/algorithm. The set P is to be protected. Based upon the *Negative Selection* algorithm, generate a set C from the given/present binary image, i.e., set I in Fig. 4. The structure of elements is described in Fig. 2.

Now depending on the logic for matching we segregate the elements of candidate set C into two sets namely *Accepted set* (A) and *Rejected set* (R). The set A comprises of those points of C which are un-identified by the protected set P (i.e., set

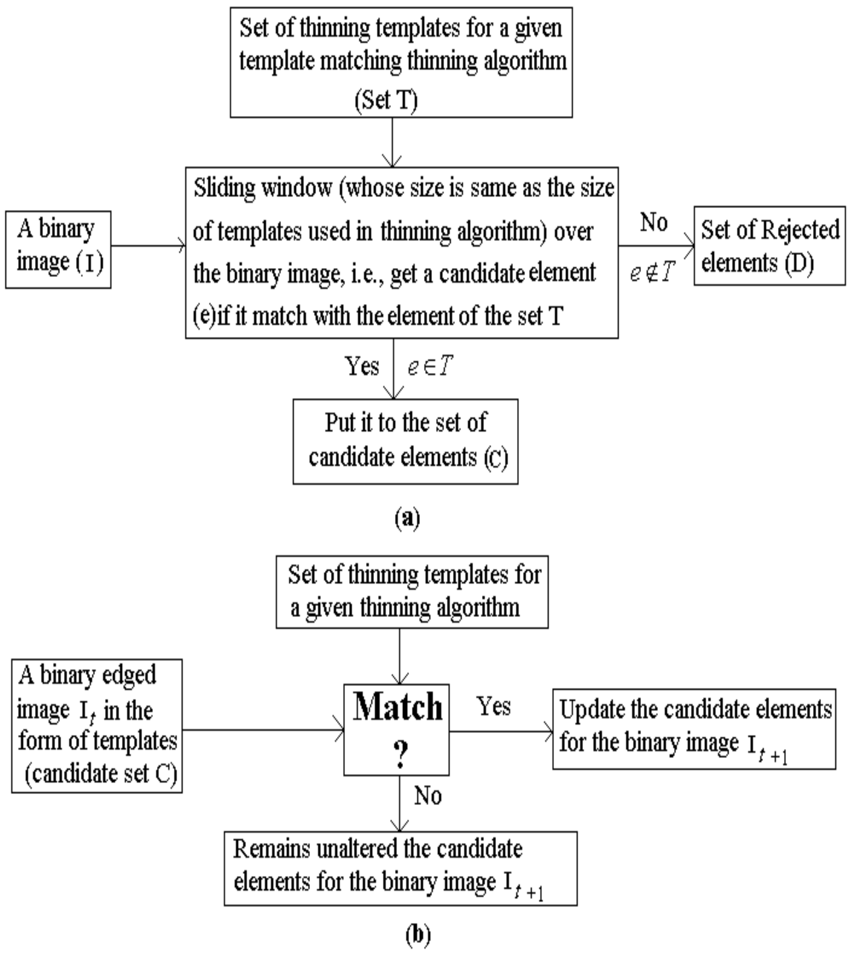
Set of thinning templates for a given template matching thinning algorithm (Set T)
A binary image (I)
Sliding window (whose size is same as the size of templates used in thinning algorithm) over the binary image, i.e., get a candidate element (e)if it match with the element of the set T
No
e ∉ T
Set of Rejected elements (D)
Yes
e ∈ T
Put it to the set of candidate elements (C)
(a)
Set of thinning templates for a given thinning algorithm
A binary edged image I_t in the form of templates (candidate set C)
Match ?
Yes
Update the candidate elements for the binary image I_t+1
No
Remains unaltered the candidate elements for the binary image I_t+1
(b)

FIGURE 4. Schematic diagram for the functioning of template matching thinning algorithm using negative selection method of immunecomputing paradigm: (a) Production of detectors, (b) Monitoring for the presence of undesired (*nonself*) pattern.

of templates, T in Fig. 4), lies inside the intermediate thinned image in C and set R consists of those points which are identified by the protected set P, i.e., pixels are to be deleted for the binary edged image in $C = \{X_1, X_2, \ldots, X_n\}$, as described in Eqn. (4.1).

$$X_i \in \begin{cases} R, & \text{if } \mathbf{w}_i \in P \\ A, & \text{otherwise} \end{cases} \tag{4.1}$$

for $i = 1, 2, \ldots, n$.

After determining sets A and R, we extract information from them which will be used in the maturation of the candidate set C. Extract the position of each element of the set R and set of elements of the binary image $I^{(t)}(x_i, y_i) \leftarrow 0$ when $(x_i, y_i, b_{i_1} b_{i_2} \cdots b_{i_k}) \in R$ where $I^{(0)}$ is the initial $r \times c$ binary edged image for $i = 1, 2, \ldots, n^{(0)}$ and $n^{(0)}$ is the initial number of 1's in the binary image $I^{(0)}$. Set $I^{(t+1)} \leftarrow$

$I^{(t)}$. Reconstruct the candidate set C using the binary image $I^{(t+1)}$. Repeat the above procedure until $R = \phi$. This process is described in Algorithm 3.

```
Algorithm 3: Thinning of binary edged image using immunecomputing paradigm
Input: Binary edged image $I$
Output: Thinned binary edged image $I$
  Step 1. [Initialization] Initialize iteration number $t = 0$,
          generate a candidate set $C(t) = \{X_i | i = 1, 2, \ldots, n^{(0)}\}$ where $n^{(0)}$ is the
          number of 1 in the binary edge image
  Step 2. [Grouping] Split the set $C(t)$ into two sets $A$ and $R$ using match logic
          according to rule in (4.1).
  Step 3. [Modification] Modify the binary image $I^{(t)}$ based on the set $R$.
  Step 4. [Increment] Set $t \leftarrow t + 1$.
  Step 5. [Reconstruction] Reconstruct the candidate set $C(t) = \{X_i | i =
          1, 2, \ldots, n^{(t)}\}$.
  Step 6. [Looping] Repeat from Step 2 to Step 5 until $R = \phi$.
  Step 7. [Termination] Stop.
```

4.3. **Experimental Results of Thinning Process.** The one-pass template matching thinning process describe in Algorithm 2 is applied on a binary edge image as shown in Fig. 5(a). The thinned image is shown in Fig. 5(b) where the elements at $'*'$ are not changed to 0 from 1. For simplicity, both input and output thinned images are incorporated in one figure as shown in Fig. 5(c). Other examples are also shown in Fig. 6.

```
111111111111111      111111111111111      ...............
111111111111111      111111111111111      ...............
111111111111111      111111111111111      ...............
111111111111111      1111********111      ....********...
111111111111111      111*11111111111      ...*...........
111111111111111      111*11111111111      ...*...........
111111111111111      111*11111111111      ...*...........
111111100000000      111*11100000000      ...*...
111111100000000      111*11100000000      ...*...
111111100000000      111*11100000000      ...*...
111111100000000      111*11100000000      ...*...
111111100000000      111*11100000000      ...*...
111111100000000      111111100000000      .......
111111100000000      111111100000000      .......
      (a)                  (b)                  (c)
```

FIGURE 5. Experimental results of thinning process.

```
.........         .........
.........         .........
.........         .........
.........         .........
....*....         ....*....
....*....         ....*....
....*....         ....*....
....*....         ....*....
....*....         ....*....
....*.................*....
....*.................*....
....*.................*....
....*******************....
....*.................*....
....*.................*....
....*.................*....
....*.................*....
....*....         ....*....
....*....         ....*....
....*....         ....*....
....*....         ....*....
....*....         ....*....
....*....         ....*....
.........         .........
.........         .........
.........         .........
.........         .........
              (a)
```

```
  .....
  .....
  ..*..
  ..*..
....*....
.*******.
....*....
  ..*..
  ..*..
  .....
  .....

   (b)
```

```
    .
   ...
  .***.
 .*. .*.
 .*. .*.
..*****..
.*.....*.
.*.   .*.
.*.   .*.
.*.   .*.
...   ...
   (c)
```

FIGURE 6. Other experimental results of thinning process.

5. **Analysis**

In this section we study that the template matching thinning process on a binary image is analogous to the Markov process and also we study the performance of it. Some related terms are defined in the following section.

5.1. **Some Definitions.** **Stochastic process** can be defined [23] as a collection of random variables X_t, $t \in T$, where these random variables are defined on a common probability space and $T \subset (-\infty, \infty)$ is thought of as a time parameter set. The process is called a **continuous parameter process** if T is an interval having positive length and a **discrete parameter process** if T is a subset of the integers. If the random variables X_t all take on values from the fixed set S, then S is called the **state space** of the process.

Many stochastic processes possess the property that, the given present state of a process does not effect on, its past history, i.e., the past states have no influence on the future, mathematically does not affect the conditional probabilities of events defined in terms of the future. Such processes are called **Markov processes**. This property is known as **Markov property** and the systems having this property are called **Markov chains**. Equivalently, we can say that a process X_n having state space **S** is said to be a Markov process if for every choice of non-negative integer n and the numbers $x_0, x_1, \ldots, x_{n+1}$ each in S, the probability

$$P(X_k = x_k | X_0 = x_0, \ldots, X_{k-1} = x_{k-1}) = P(X_k = x_k | X_{k-1} = x_{k-1}) \tag{5.1}$$

Also the conditional probabilities $P(X_t = y | X_0 = x)$ are called the **transition probabilities** of the chain. A Markov process X_t, $t \in T$, is said to be **jump process** if

$$x = \begin{cases} x_0, & \text{for } 0 \le t < \tau_1 \\ x_1, & \text{for } \tau_1 \le t < \tau_2 \\ x_2, & \text{for } \tau_2 \le t < \tau_3 \\ \vdots & \\ x_{k-1}, & \text{for } \tau_{k-1} \le t < \tau_k \\ x_{k-1}, & \text{for } \tau_k \le t < \infty \end{cases} \tag{5.2}$$

where, $T = [0, \infty)$.

5.2. **Pure Death Process.** Let Y_t be the random variable (r.v.) representing a number of deaths occurred at time interval of length t. So Y_t takes values 0,1,2,... etc. The process $\{Y_t, t > 0\}$ is known as **death process** [23] provided the following assumptions hold.

Assumption 1: The conditional probability that during an interval $(t, t+h)$, where h (> 0) is small, a death occurs, given that at the beginning of the interval the system was in state i, is approximately equal to $\mu_i h$, where μ_i is death rate in state i.

Assumption 2: The conditional probability of more than one death during $(t, t+h)$ is negligible. In pure death process there is absolutely no birth. Thus, the pure death process can be treated as jump process.

Define the transition probabilities as

$$p_{i,j}(t) = P(Y_{t+h} = j | Y_h = i); \text{ for } t, h > 0, \tag{5.3}$$

which gives the probability that at time $(t+h)$ the system is in state j (i.e., number of alive at time $t+h$ is j) given that at time h it was in state i. So, we have $p_{ij}(t) = 0$, for $j > i$, since it is pure death process. So, by Assumption 1 we get

$$p_{i,i-1}(h) = P(Y_{t+h} = i-1|Y_t = i) = \mu_i h, \tag{5.4}$$

By Assumption 2 we get

$$p_{j+k,j}(h) = P(Y_{t+h} = j|Y_t = j+k), \tag{5.5}$$

which is negligible, for $k > 1$, i.e., $p_{j+k,j}(h) = 0$, $k > 1$

Suppose that at time $t = 0$ the system is at state i. In order to have state j at time $t + h$ (where h is small positive number), we consider the following three possibilities:

(a): at time t, it is in state $j+1$ and one death occurred during $(t, t+h)$,
(b): at time t, it is in state j and no death occurred during $(t, t+h)$, and
(c): more than one death occurred during $(t, t+h)$.

Thus we have,

$$\begin{aligned} p_{i,j}(t+h) &= p_{i,j+1}(t)p_{j+1,j}(h) + p_{i,j}(t)p_{j,j}(h) + \sum_{k=j+2}^{i} p_{i,k}(t)p_{k,j}(h) \\ &= p_{i,j+1}(t)\mu_{j+1}h + p_{i,j}(t)[1-\mu_j h] + \text{negligible terms} \\ &\qquad [\text{by Assumption 1 and Assumption 2 and the fact, } \sum_{k=0}^{i} p_{j,j-k}(h) = 1] \end{aligned}$$

Now by dividing both sides by h and taking limit on h, i.e., as $h \to 0$, we get

$$p'_{i,j}(t) = p_{i,j+1}(t)\mu_{j+1} - p_{i,j}(t)\mu_j \tag{5.6}$$

Initial conditions are

$$p_{i,j}(0) = 0, \forall\ i \neq j \text{ and } p_{i,i}(0) = 1 \tag{5.7}$$

Now, using Eqns. (5.6) and (5.7) we get

$$p'_{i,i}(t) = -p_{i,i}(t)\mu_i \tag{5.8}$$

By solving the differential equation (5.8) and using initial conditions in (5.7) we get

$$p_{i,i}(t) = e^{-\mu_i t},\ \forall t \geq 0 \tag{5.9}$$

Put $j = i-1$ in Eqn. (5.6), so we get using Eqn. (5.9)

$$p'_{i,i-1}(t) = p_{i,i}(t)\mu_i - p_{i,i-1}(t)\mu_{i-1} = \mu_i e^{-\mu_i t} - p_{i,i-1}(t)\mu_{i-1} \tag{5.10}$$

Lemma 1 : If $f'(t) = \alpha f(t) + g(t)$, for $t \geq 0$, then $f(t) = e^{\alpha t}f(0) + \int_0^t e^{\alpha(t-s)}g(s)ds$.

Proof: One can easily prove it by multiplying both sides by $e^{-\alpha s}$ of the given condition and integrating over s from 0 to t, i.e.,

$\int_0^t e^{-\alpha s} f'(s)ds = \alpha \int_0^t f(s)e^{-\alpha s}ds + \int_0^t g(s)e^{-\alpha s}ds$

Using Integration by parts, we get

$e^{-\alpha s}f(s)|_0^t + \alpha \int_0^t e^{-\alpha s}f(s)ds = \alpha \int_0^t f(s)e^{-\alpha s}ds + \int_0^t g(s)e^{-\alpha s}ds$

or, $e^{-\alpha t}f(t) - f(0) = \int_0^t g(s)e^{-\alpha s}ds$

or, $f(t) = e^{\alpha t}f(0) + \int_0^t g(s)e^{\alpha(t-s)}ds$

So using Lemma 1, we get from Eqn. (5.10) as

$p_{i,i-1}(t) = e^{-\mu_{i-1}t}p_{i,i-1}(0) + \mu_i \int_0^t e^{-\mu_{i-1}(t-s)}e^{-\mu_i s}ds$

Using conditions in (5.7) we get

$$p_{i,i-1}(t) = \frac{\mu_i}{\mu_i - \mu_{i-1}}(e^{-\mu_{i-1}t} - e^{-\mu_i t}) \tag{5.11}$$

Now putting $j = i - 2$ in Eqn (5.6), we get

$p'_{i,i-2}(t) = \mu_{i-1}p_{i,i-1}(t) - \mu_{i-2}p_{i,i-2}(t)$

i.e., $p_{i,i-2}(t) = \frac{\mu_i\mu_{i-1}}{\mu_i - \mu_{i-1}} e^{-\mu_{i-2}t}\left[\frac{1 - e^{-(\mu_{i-1}-\mu_{i-2})t}}{\mu_{i-1} - \mu_{i-2}} - \frac{1 - e^{-(\mu_i-\mu_{i-2})t}}{\mu_i - \mu_{i-2}}\right]$ [by Lemma 1]

It is reasonable to assume that $\mu_i \propto i,\ \forall\ i$. So $\mu_i = i\mu$ where μ is the proportionality constant.

Now, $p_{i,i-1}(t) = ie^{-\mu t(i-1)}(1 - e^{-\mu t})$ and $p_{i,i-2}(t) = \begin{pmatrix} i \\ 2 \end{pmatrix} e^{-\mu t(i-2)}(1 - e^{-\mu t})^2$

Thus for any $j \leq i$, we have

$$p_{i,j}(t) = \begin{pmatrix} i \\ j \end{pmatrix} e^{-\mu tj}(1 - e^{-\mu t})^{i-j} \tag{5.12}$$

5.3. **Performance Study.** We attempt to study the performance of template matching thinning algorithms using immunecomputing paradigm. We formulate that the template matching thinning process using negative selection method of immunecomputing paradigm is analogous to a kind of stochastic process (i.e., Markov process) in statistics.

5.3.1. *Template Matching Thinning Process.* The central/decision element of the thinning templates is considered as 1 in all the thinning algorithms and also the objective of a thinning algorithm is to replace 1 by 0 as the desire criteria are fulfilled. Each element of a binary pattern is considered as a central element matches with a given set of thinning templates then this element is converted to 0 from 1. This converted value is to be used in the subsequent iteration, and this process is continued, till no conversion from 1 to 0 takes place in an iteration.

$$I_0 \xRightarrow{A} I_1 \xRightarrow{A} I_2 \xRightarrow{A} \cdots \xRightarrow{A} I_{k-1} \xRightarrow{A} I_k \xRightarrow{A} \cdots$$

FIGURE 7. Thinning of a binary edged image I_0 by a template matching algorithm A in k iterations

5.3.2. *Thinning Process using AIC.* Suppose A is a parallel template matching thinning algorithm and I_0 is the given binary edged image. Now we apply the Algorithm 2 on the binary image I_0 and an output I_1 is obtained. Again the Algorithm 2 is applied in I_1 repeatedly, until two consecutive outputs are equal. We get different modified images, say, $I_1, I_2, \ldots, I_k$ after 1st, 2nd, ..., kth iterations respectively and then it will stop after kth iterations, if the number of 1's (considering binary image) in I_{k-1} is same as that in I_k. The intermediate thinned image I_{r+1} depends only on the immediate past thinned image I_r, not on the previous intermediate output images $I_0, I_1, \ldots, I_{r-1}$. This process of thinning of I_0 using Algorithm 2 is shown in Fig. 7.

Suppose X_r is a random variable denoting the number of 1's in I_r, $r = 0, 1, \ldots, k$, then we can say that

$$P(X_k = x_k | X_0 = x_0, \ldots, X_{k-1} = x_{k-1}) = P(X_k = x_k | X_{k-1} = x_{k-1}) \tag{5.13}$$

where, $x_1, x_2, \ldots, x_k$ are non-negative integers. Therefore template matching thinning process on binary edged image I_0 by the Algorithm 2 (Fig. 7) is a *Markov process.* Since its present state depends only on the previous state of the binary image. The process of thinning for a given binary edged image can also be treated as *pure jump process*, as defined in Eqn. (5.2). This gives a number of 1 in the binary images at different time point t. In thinning process of binary edged image, 0 can never be changed to 1. But 1 may change to 0 or remains unchanged. So it is a *Pure Death Process.*

Now, $P(Y_0 = x_0, Y_{\tau_1} = x_1, \ldots, Y_{\tau_{k-1}} = x_{k-1}, Y_{\tau_k} = x_{k-1})$
$= P(Y_0 = x_0)P(Y_{\tau_1} = x_1 | Y_0 = x_0)P(Y_{\tau_2} = x_2 | Y_0 = x_0, Y_{\tau_1} = x_1) \cdots$
$P(Y_{\tau_{k-1}} = x_{k-1} | Y_0 = x_0, \ldots, Y_{\tau_{k-2}} = x_{k-2})P(Y_{\tau_k} = x_{k-1} | Y_0 = x_0, \ldots, Y_{\tau_{k-1}} = x_{k-1})$
$= P(Y_0 = x_0)P(Y_{\tau_1} = x_1 | Y_0 = x_0) \cdots P(Y_{\tau_{k-1}} = x_{k-1} | Y_{\tau_{k-2}} = x_{k-2})$
$P(Y_{\tau_k} = x_{k-1} | Y_{\tau_{k-1}} = x_{k-1})$ [By Markov Property]

$$= \pi(0) \binom{x_0}{x_1} e^{-\mu x_1 \tau_1} [1 - e^{-\mu \tau_1}]^{(x_0 - x_1)} \binom{x_1}{x_2} e^{-\mu x_2 (\tau_2 - \tau_1)} [1 - e^{-\mu(\tau_2 - \tau_1)}]^{(x_1 - x_2)} \cdots$$
$$\binom{x_{k-2}}{x_{k-1}} e^{-\mu x_{k-1}(\tau_{k-1} - \tau_{k-2})} [1 - e^{-\mu(\tau_{k-1} - \tau_{k-2})}]^{(x_{k-2} - x_{k-1})} e^{-\mu x_{k-1}(\tau_k - \tau_{k-1})}$$

where, $\pi(0) = P(Y_0 = x_0)$ and by using the result in Eqn. (5.12)

$$= \pi(0)\left[\prod_{i=1}^{k-1}\begin{pmatrix} x_{i-1} \\ x_i \end{pmatrix}\right] e^{-\mu\left[\sum_{i=1}^{k-1} x_i(\tau_i - \tau_{i-1}) + x_{k-1}(\tau_k - \tau_{k-1})\right]} \times$$

$\left[\prod_{i=1}^{k-1}[1 - e^{-\mu(\tau_i - \tau_{i-1})}]^{(x_{i-1}-x_i)}\right]$ [with $\tau_0 = 0$]

Therefore

$P(Y_0 = x_0, Y_{\tau_1} = x_1, \ldots, Y_{\tau_{k-1}} = x_{k-1}, Y_{\tau_k} = x_{k-1})$

$$= \pi(0)\left[\prod_{i=1}^{k-1}\begin{pmatrix} x_{i-1} \\ x_i \end{pmatrix}\right] e^{-\mu d\left(\sum_{i=1}^{k-1} x_i + x_{k-1}\right)} \left(1 - e^{-\mu d}\right)^{(x_0 - x_{k-1})} \tag{5.14}$$

[Assuming that $\tau_i - \tau_{i-1} = d$, $\forall\ i$]

Let $w = u \times v$ be the number of elements in the templates used in a template matching binary thinning algorithm. So 2^w is the number of all possible patterns of size w.

Let m be the number of templates used in the template matching thinning algorithm. Now we shall compute the probabilities for the following events.

(1): 0 is changed to 1 is an impossible event in a thinning algorithm. So the probability of the event is 0.

(2): 0 is changed to 0 is a certain event in a thinning algorithm. The probability of the event is 1.

(3): 1 is changed to 0 is an event whose probability depends upon the nature of thinning algorithm applied on a binary edge images. Note that a thinning algorithm does not return a null binary image from a non-null binary image (in which the number of 1 is not equal to zero). So the probability must lies between 0 and 1.

(4): 1 is changed to 1 is an event whose probability depends upon the nature of the thinning algorithm applied on a binary edge images.

Now we define the probability as:

P(0 is changed to 1) $= p_{01} = 0$.

P(0 is changed to 0) $= p_{00} = 1$.

P(1 is changed to 0) $= p_{10}$

where

$p_{10} = \dfrac{\text{Total number of possible windows (i.e., template) for matching}}{\text{All possible windows (i.e., template)}} = \dfrac{m}{2^w}$, and

$m =$ number of templates used in the thinning algorithm

P(1 is changed to 1) $= p_{11} = 1 - p_{10}$

These probabilities must satisfy the criteria

$$p_{01} = 0, p_{00} = 1, 0 < p_{10} < 1, 0 < p_{11} < 1 \tag{5.15}$$

Initially, number of 1's in the input binary edged image is X_0. After the first iteration the number of 1's in the output binary image is $x_1 = x_0 \times p_{11}$.
Similarly $x_2 = x_1 \times p_{11} = x_0 \times p_{11}^2$. Hence,

$$x_i = x_0 \times p_{11}^i,\ \forall\ i = 1, 2, \ldots, k-1 \tag{5.16}$$

where $k =$ number of iterations applied on the input binary edged image.

So, the probability
$P(Y_0 = x_0, Y_{\tau_1} = x_1, \ldots, Y_{\tau_{k-1}} = x_{k-1}, Y_{\tau_k} = x_{k-1})$

$$= \pi(0) \prod_{i=1}^{k-1} \left(\begin{array}{c} x_{i-1} \\ x_i \end{array} \right) e^{-\mu d x_0 \left[p_{11} \dfrac{1-p_{11}^{k-1}}{1-p_{11}} + p_{11}^{k-1} \right]} \left(1 - e^{-\mu d}\right)^{x_0\left(1-p_{11}^{k-1}\right)} \leq 1 \tag{5.17}$$

[as it is a probability]
where $d =$ time taken for one iteration, i.e., state transition time.

Lemma 2: If $N = n_1 + n_2 + \cdots + n_k$ for any non-negative integer N, where, $n_1, n_2, \ldots, n_k$ are also non-negative integer, then $N! \geq n_1! n_2! \cdots n_k!$

Proof: Trivial.

Now, by Lemma 2 we can write

$$\prod_{i=1}^{k-1} \left(\begin{array}{c} x_{i-1} \\ x_i \end{array} \right) = \frac{x_0!}{(x_0 - x_1)!(x_1 - x_2)! \cdots (x_{k-2} - x_{k-1})! x_{k-1}!} \geq 1, \tag{5.18}$$

Hence combining the results in Eqns. (5.17) and (5.18) we get,

$$\pi(0) e^{-\mu d x_0 \left[p_{11} \dfrac{1-p_{11}^{k-1}}{1-p_{11}} + p_{11}^{k-1} \right]} \left(1 - e^{-\mu d}\right)^{x_0(1-p_{11}^{k-1})} \leq 1$$

i.e., $\log \pi(0) - \mu d x_0 p_{11} \dfrac{1-p_{11}^{k-1}}{1-p_{11}} - \mu d x_0 p_{11}^{k-1} + x_0(1 - p_{11}^{k-1}) \log(1 - e^{-\mu d}) \leq 0$

i.e., $x_0 p_{11}^{k-1} \left[\mu d \dfrac{2p_{11} - 1}{1 - p_{11}} - \log(1 - e^{-\mu d}) \right]$

$$\leq \mu d x_0 \frac{p_{11}}{1 - p_{11}} - x_0 \log(1 - e^{-\mu d}) - \log \pi(0) \tag{5.19}$$

Here, μd is the average number of deletion during d. So, $\mu d > 0$.
Thus $\log(1 - e^{-\mu d}) < 0$
Case 1: $0.5 < p_{11} < 1$.
So, $1 - p_{11} > 0$ and $2p_{11} - 1 > 0$. Therefore, $\dfrac{2p_{11} - 1}{1 - p_{11}} > 0$.

Hence, $\mu d \dfrac{2p_{11} - 1}{1 - p_{11}} - \log(1 - e^{-\mu d}) > 0$, provide $p_{11} > 0.5$.
Again, μd is the average number of deletion (i.e., from 1 to 0) during time d.
So $\mu d > 0$.

Using Eqn. (5.19) we get

$$p_{11}^{k-1} \leq \frac{\mu d x_0 \frac{p_{11}}{1-p_{11}} - x_0 \log(1-e^{-\mu d}) - \log \pi(0)}{x_0[\mu d \frac{2p_{11}-1}{1-p_{11}} - \log(1-e^{-\mu d})]} \quad = 1 + \frac{\mu d x_0 - \log \pi(0)}{x_0[\mu d \frac{2p_{11}-1}{1-p_{11}} - \log(1-e^{-\mu d})]}$$

$= C$ (say)

As $0 < \pi(0) < 1$, so, $\mu d x_0 - \log \pi(0) > 0$.
Thus, $C > 0$

So, $(k-1)\log p_{11} < \log C$ or, $k > 1 + \dfrac{\log C}{\log p_{11}}$, i.e., $k \geq 1$ [as $p_{11} < 1$ and so $\log p_{11} < 0$]

Case 2 : $0 < p_{11} \leq 0.5$.

Assume $\mu d \dfrac{2p_{11}-1}{1-p_{11}} - \log(1-e^{-\mu d}) < 0$,

i.e., $-2 + \dfrac{1}{1-p_{11}} - \dfrac{\log(1-e^{-\mu d})}{\mu d} < 0$ [as $\mu d > 0$]

i.e., $p_{11} < 1 - \dfrac{1}{2 + \frac{\log(1-e^{-\mu d})}{\mu d}}$.

Again, $0 < 2 + \dfrac{\log(1-e^{-\mu d})}{\mu d} < 2$ as $\log(1-e^{-\mu d}) < 0$.

So, $\dfrac{1}{2 + \frac{\log(1-e^{-\mu d})}{\mu d}} > 0.5$ or, $1 - \dfrac{1}{2 + \frac{\log(1-e^{-\mu d})}{\mu d}} < 0.5$

Thus, $0 < p_{11} < 1 - \dfrac{1}{2 + \frac{\log(1-e^{-\mu d})}{\mu d}} < 0.5$, which is true.

So, $\mu d \dfrac{2p_{11}-1}{1-p_{11}} - \log(1-e^{-\mu d}) < 0$,

Thus $k < 1 + \dfrac{\log C}{\log p_{11}}$, provided $p_{11} < 0.5$ (proceeding as *Case 1*).

But in practice, $p_{11} \leq 0.5$. So, this relation provides lower bound of the number of iteration required to converge the process of thinning of binary image. We compute an upper bound of k.

Suppose, $P(Y_t = y | Y_0 = x) = e^{-\lambda} \dfrac{\lambda^{x-y}}{(x-y)!}$, which is *Poisson distribution* with λ as the parameter denoting average deletion of 1's in each iteration.

Now,

$$\begin{aligned}
1 &\geq P(Y_0 = x_0, Y_{\tau_1} = x_1, \ldots, Y_{\tau_{k-1}} = x_{k-1}, Y_{\tau_k} = x_{k-1}) \\
&= P(Y_0 = x_0) P(Y_{\tau_1} = x_1 | Y_0 = x_0) \ldots P(Y_{\tau_{k-1}} = x_{k-1} | Y_0 = \\
&\quad x_0), \ldots, P(Y_{\tau_{k-2}} = x_{k-2}) P(Y_{\tau_k} = x_{k-1} | Y_0 = x_0), \ldots, P(Y_{\tau_{k-1}} = x_{k-1}) \\
&= P(Y_0 = x_0) P(Y_{\tau_1} = x_1 | Y_0 = x_0) \cdots P(Y_{\tau_{k-1}} = x_{k-1} | Y_{\tau_{k-2}} = \\
&\quad x_{k-2}) P(Y_{\tau_k} = x_{k-1} | Y_{\tau_{k-1}} = x_{k-1}) \text{ [by Markov property]} \\
&= \pi(0) e^{-\lambda} \frac{\lambda^{x_0 - x_1}}{(x_0 - x_1)!} e^{-\lambda} \frac{\lambda^{x_1 - x_2}}{(x_1 - x_2)!} \cdots e^{-\lambda} \frac{\lambda^{x_{k-2} - x_{k-1}}}{(x_{k-2} - x_{k-1})!} \\
&= \pi(0) e^{-k\lambda} \frac{\lambda^{x_0 - x_{k-1}}}{(x_0 - x_1)!(x_1 - x_2)! \cdots (x_{k-2} - x_{k-1})!} \\
&\geq \pi(0) e^{-k\lambda} \frac{\lambda^{x_0 - x_{k-1}}}{(x_0 - x_{k-1})!} \text{ [by Lemma 2]}
\end{aligned}$$

Therefore

$$1 \geq \pi(0) e^{-k\lambda} \frac{\lambda^{x_0 - x_{k-1}}}{(x_0 - x_{k-1})^{x_0 - x_{k-1}}} \text{ [as } \frac{1}{i!} \geq \frac{1}{i^i}] \tag{5.20}$$

Now, $x_0 > x_{k-1}$ [trivial] or $0 < x_0 - x_{k-1} < x_0$ then

$$\frac{1}{(x_0 - x_{k-1})^{x_0 - x_{k-1}}} > \frac{1}{x_0{}^{x_0}} \tag{5.21}$$

Combining the results in Eqns. (5.20) and (5.21), we get

$$\pi(0) e^{-k\lambda} \frac{\lambda^{x_0 - x_{k-1}}}{(x_0)^{x_0}} < 1. \tag{5.22}$$

Here, λ is estimated unbiased by the mean of the random variable denoting the number of deletion in each iteration.

$$\hat{\lambda} = \frac{1}{k}[(x_0 - x_1) + (x_1 - x_2) + \cdots + (x_{k-2} - x_{k-1})] = \frac{x_0 - x_{n-1}}{k}$$

$$x_0 - x_{k-1} = k\hat{\lambda} \tag{5.23}$$

Combining the results in Eqns. (5.22) and (5.23) we get

$\pi(0) e^{-k\hat{\lambda}} \dfrac{\hat{\lambda}^{k\hat{\lambda}}}{(x_0)^{x_0}} < 1$, i.e., $\dfrac{\pi(0)}{(x_0)^{x_0}} \left(\dfrac{\hat{\lambda}}{e}\right)^{k\hat{\lambda}} < 1$, i.e., $\left(\dfrac{\hat{\lambda}}{e}\right)^{-k\hat{\lambda}} > \dfrac{\pi(0)}{(x_0)^{x_0}}$

i.e., $k\hat{\lambda} \log \left(\dfrac{e}{\hat{\lambda}}\right) > \log \pi(0) - x_0 \log x_0$

Therefore

$$k < \frac{1}{\hat{\lambda}} \left[\frac{\log \pi(0) - x_0 \log x_0}{1 - \log \hat{\lambda}}\right] \tag{5.24}$$

Assume $B = \dfrac{1}{\hat{\lambda}} \left[\dfrac{\log \pi(0) - x_0 \log x_0}{1 - \log \hat{\lambda}}\right]$, $\hat{\lambda} > e$, i.e., $1 - \log \hat{\lambda} < 0$

Since $0 < \pi(0) < 1$, i.e., $\log \pi(0) < 0$, again x_0 is a positive integer and $x_0 \geq 1$. So $x_0 \log x_0 > 0$ (Note that $x_0 \log x_0 = 0$ only when $x_0 = 0$ when the input binary edged image itself is a thinned image) and thus $\log \pi(0) - x_0 \log x_0 < 0$ as $x_0 > 1$.

Therefore $B > 0$. So, Eqn. (5.24) gives an upper bound for k.

6. **Results and Conclusion**

The performance of an image processing algorithm is measured in terms of time and space requirement. These are the functions of the size of the input image. We have designed a model for the study of average case complexity that measures the performance of template matching thinning algorithms using immunocomputing paradigm for the binary images. In this model, we have computed the bounds on the number of iterations required to converge a thinning process. In this context we have conducted several experiments with normally distributed binary images [38] for the following binary template matching thinning algorithms designed by: (1) Zhang & Suen [49], (2) Chin *et. al.* [11], (3) Hall [19], (4) Holt *et. al.* [25], and (5) Pal & Bhattacharyya [37]. After applying on a normally distributed binary images [38] we measured the number of iterations and time requirements using workstation. The results from the experiment and by computation from the proposed model are shown in Tables 1-10 in which θ, σ and n are the initial angle, standard deviation and number of points (i.e., 1's in the generated binary image) respectively. These are the parameters of the algorithm [38] for normally distributed binary image generation. N is the number of iterations required, B is the bound on number of iterations obtained from the model and $\hat{\lambda}$ is the average number of deletions in each iteration and T is the required time in μsec. The ratio T/B indicates the implementation complexity of the algorithms. The order of implementation complexity of these thinning algorithms [49, 11, 19, 25, 37] are shown in Tables 1-10. Table 1 to Table 6 are corresponding to image of size 32×32 and the rests (Tables 7-10) are corresponding to 64×64.

Table 1 : $\theta = 0$, $\sigma = 0.356$, $n = 456$, $x_0 = 610$

Algorithm	N	$\hat{\lambda}$	B	Time T (μsec)	Ratio (T/B)
Zhang & Suen [49]	5	37.60	39.542	2260	57.15
Chin [11]	11	42.72	34.272	3430	100.08
Hall [19]	7	66.86	18.242	1790	98.12
Holt [25]	8	56.37	22.85	5820	254.70
Pal & Bhattacharyya [37]	9	57.78	22.116	9470	428.19

Table 2 : $\theta = 36$, $\sigma = 0.467$, $n = 456$, $x_0 = 616$

Algorithm	N	$\hat{\lambda}$	B	Time T (μsec)	Ratio (T/B)
Zhang & Suen [49]	4	55.75	23.456	2010	85.69
Chin [11]	10	48.50	28.267	2990	105.77
Hall [19]	6	80.33	14.522	1570	108.11
Holt [25]	7	67.00	18.398	5200	282.64
Pal & Bhattacharyya [37]	10	58.30	22.103	8590	388.68

Table 3 : $\theta = 47$, $\sigma = 0.567$, $n = 456$, $x_0 = 694$

Algorithm	N	$\hat{\lambda}$	B	Time T (μsec)	Ratio (T/B)
Zhang & Suen [49]	6	41.83	39.646	2960	74.66
Chin [11]	13	43.96	37.365	4000	107.05
Hall [19]	9	62.89	22.95	2300	100.21
Holt [25]	10	55.40	27.148	7570	278.84
Pal & Bhattacharyya [37]	8	80.00	16.757	8850	528.13

Table 4 : $\theta = 9$, $\sigma = 0.245$, $n = 357$, $x_0 = 440$

Algorithm	N	$\hat{\lambda}$	B	Time T (μsec)	Ratio (T/B)
Zhang & Suen [49]	4	31.00	35.419	1190	33.59
Chin [11]	6	51.67	17.562	1630	93.03
Hall [19]	3	105.00	6.965	770	110.55
Holt [25]	4	47.75	10.786	2540	235.49
Pal & Bhattacharyya [37]	4	86.00	8.995	3240	360.20

Table 5 : $\theta = 47$, $\sigma = 0.677$, $n = 478$, $x_0 = 742$

Algorithm	N	$\hat{\lambda}$	B	Time T (μsec)	Ratio (T/B)
Zhang & Suen [49]	7	39.57	46.213	3600	77.90
Chin [11]	16	37.75	49.311	5380	109.10
Hall [19]	10	61.40	25.587	2760	170.86
Holt [25]	11	53.64	30.618	9290	303.41
Pal & Bhattacharyya [37]	9	75.78	19.421	10540	542.71

Table 6 : $\theta = 28$, $\sigma = 0.367$, $n = 389$, $x_0 = 605$

Algorithm	N	$\hat{\lambda}$	B	Time T (μsec)	Ratio (T/B)
Zhang & Suen [49]	5	38.00	38.599	1950	50.50
Chin [11]	17	26.59	63.805	5050	79.15
Hall [19]	7	65.86	18.43	1770	96.04
Holt [25]	7	62.28	19.833	5430	273.78
Pal & Bhattacharyya [37]	8	66.12	18.332	7920	432.03

Table 7 : $\theta = 0$, $\sigma = 0.478$, $n = 698$, $x_0 = 1717$

Algorithm	N	$\hat{\lambda}$	B	Time T (μsec)	Ratio (T/B)
Zhang & Suen [49]	6	83.67	44.579	6330	141.99
Chin [11]	9	136.78	23.848	8520	357.26
Hall [19]	6	209.67	14.028	4620	329.34
Holt [25]	7	172.00	17.917	14590	814.31
Pal & Bhattacharyya [37]	7	177.86	17.188	16900	983.24

Table 8 : $\theta = 15$, $\sigma = 0.699$, $n = 798$, $x_0 = 1636$

Algorithm	N	$\hat{\lambda}$	B	Time T (μsec)	Ratio (T/B)
Zhang & Suen [49]	9	68.44	54.795	9140	166.80
Chin [11]	17	78.65	45.718	13980	305.76
Hall [19]	11	122.91	25.827	7590	293.87
Holt [25]	12	109.25	29.983	21940	731.74
Pal & Bhattacharyya [37]	11	120.27	26.544	23560	887.58

Table 9 : $\theta = 30$, $\sigma = 0.955$, $n = 1276$, $x_0 = 2930$

Algorithm	N	$\hat{\lambda}$	B	Time T (μsec)	Ratio (T/B)
Zhang & Suen [49]	11	96.10	68.249	18040	264.32
Chin [11]	31	78.387	88.731	36090	406.73
Hall [19]	18	134.50	44.557	16510	370.53
Holt [25]	18	130.89	46.107	52140	1130.84
Pal & Bhattacharyya [37]	19	133.21	45.099	63150	1400.22

Table 10 : $\theta = 45$, $\sigma = 0.966$, $n = 910$, $x_0 = 2144$

Algorithm	N	$\hat{\lambda}$	B	Time T (μsec)	Ratio (T/B)
Zhang & Suen [49]	11	75.36	65.65	13420	204.41
Chin [11]	26	68.46	74.421	25280	339.69
Hall [19]	12	149.33	27.476	10010	364.31
Holt [25]	13	133.46	31.631	31630	999.96
Pal & Bhattacharyya [37]	15	116.87	37.398	39190	1047.91

REFERENCES

[1] C. Arcelli, L. Cordella and S. Levialdi, Parallel thinning of binary pictures, *Electron Lett.*, 11(1975), 148–149.

[2] C. Arcelli and G. Ramella, Finding grey-skeletons by iterated pixel removal, *Image Vision Computation*, 13(1995), 159–167.

[3] C. Arcelli and G. Sanniti di Baja, A width-independent fast thinning algorithm, *IEEE PAMI*, 7(1985), 463–474.

[4] N. G. Bourbaks, A parallel-symmetric thinning algorithm, *Pattern Recog.*, 22(1989), 387–396.

[5] L. N. de Castro and J. Timmis. *Artificial Immune System: A New Computational intelligence Approach*, Springer, 2002.

[6] C. S. Chen and W. H. Tsai, A new fast one-pass thinning algorithm and its parallel hardware implementation, *Pattern Recog. Lett.*, 11(1990), 471–477.

[7] Y. S. Chen and W. H. Hsu, A comparison of some one-pass parallel thinnings. *Pattern Recog. Lett.*, 11(1990), 35–41.

[8] Y. S. Chen and W. H. Hsu, A systematic approach for designing 2-subcycle and pseudo-subcycle parallel thinning algorithms, *Pattern Recog.*, 22(1989), 267–282.

[9] Y. S. Chen and W. H. Hsu, A modified fast parallel algorithm for thinning digital patterns, *Pattern Recog. Lett.*, 7(1988), 99–106.

[10] Y. S. Chen, and Y.-T. Yu, Thinning approach for noisy digital patterns *Pattern Recog.*, 29(1996), 1847–1862

[11] R. T. Chin, H.-K. Wan, D. L. Stover and R. D. Iverson, A one-pass thinning algorithm and its parallel implementation, *Comput. Vision, Graphics and Image Processing*, 40(1987), 30–40.

[12] A. Datta, S. Pal and S. Chakraborti, Image thinning by neural networks, *Neural Comput. & Applic.*, 11(2002), 122–128.

[13] A. Datta and S. K. Parui, A robust parallel thinning algorithm for binary images, *Pattern Recog.*, 27(1994), 1181–1192.

[14] A. Datta, S. K. Parui and B. B. Chaudhuri, Skeletonization by a topology-adaptive self-organizing neural network, *Pattern Recog.*, 34(2001), 617–629.

[15] A. R. Dill, M. D. Levine, and P. B. Noble, Multiple resolution skeletons *IEEE Trans. on PAMI*, 9(1987), 495–504.

[16] K. C. Fan, D. F. Chen and M. G. Wen, Skeletonization of binary images with nonuniform width via block decomposition and contour vector matching. *Pattern Recog.*, 31(1998), 823–838.

[17] A. Favre and Hj. Keller, Parallel syntactic thinning by recoding of binary pictures, *Computer Vision, Graphics, and Image Processing*, 23(1983), 99–112.

[18] Z. Guo and R. W. Hall, Parallel thinning with two-subiteration algorithms, *Commun. ACM*, 32(1989), 359–373.

[19] R. W. Hall, Fast parallel thinning algorithms: parallel speed and connectivity preservation, *Commun. ACM*, 32(1989), 124–131.

[20] R. M. Haralick, Performance characterization in image analysis: thinning, a case in point, *Pattern Recog. Letters*, 13(1992), 5–12.

[21] S. Heydorn and P. Weidner, Optimization and performance analysis of thinning algorithms on parallel computers, *Parallel Comput.*, 17(1991), 17–27.

[22] C. J. Hilditch, Comparison of thinning algorithms on a parallel prcessor, *Image Vision Comput.*, 1 (1983), 115–132.

[23] P.G. Hoel, S.C. Port and C.J. Stone, Introduction to Stochastics Process, *Houghton Mifflin Company, USA*, 1972.

[24] C. M. Holt and A. Stewart, A parallel thinning algorithm with fine grain subtasking, *Parallel Comput.*, 10 (1989), 329–334.

[25] C. M. Holt, A. Stewart, M. Clint and R. H. Perrott, An improved parallel thinning algorithm, *Commun. ACM*, 30 (1987), 156–160.

[26] B. K. Jang and R. T. Chin, One-pass parallel thinning: analysis, properties and quantitative evaluation , *IEEE-PAMI*, 14(1992), 1129–1140.

[27] B. K. Jang and R. T. Chin, Analysis of thinning algorithms using mathematical morphology, *IEEE-PAMI*, 12(1990), 541–551.

[28] L. Ji and J. Piper, Fast homotopy – preserving skeletons using mathematical morphology, *IEEE-PAMI*, 14(1992), 653–664.

[29] Z. Kalmar, Z. Marczell, C. Szepesvari and A. Lorincz, Parallel and robust skeletonization built on self-organizing elements, *Neural Networks*, 12 (1999), 163–173.

[30] P. C. K. Kwok, A thinning algorithm by conture generation, *Commun. ACM*, 31 (1988), 1314–1324.

[31] L. Lam, S.-W. Lee and C. Y. Suen, Thinning methodologies - a comprehensive survey, *IEEE-PAMI*, 14(1992), 869–885.

[32] H.E. Lu and P.S.P. Wang, A comment on a fast parallel algorithm for thinning digital patterns, *Commun. ACM*, 29 (1986), 239–242.

[33] P. A. Maragos and R. W. Schafer, Morphological skeleton representation and coding of binary images, *IEEE Trans Acoust., Speech Signal Processing* ASSP, 34 (1986), 1228–1244.

[34] F. Meyer, Skeletons in digital spaces in image analysis and mathematical morphology: Theoretical Advances, J. Serra, Ed. New York, Academic, 1988.

[35] C. Neusius, J. Olszewski and D. Scheerer, An efficient distributed thinning algorithm, *Parallel Comput.*, 18(1992), 47–55.

[36] S. Pal and P. Bhattacharyya, Analysis of template matching thinning algorithms, *Pattern Recog.*, 25(1992), 497–505.

[37] S. Pal and P. Bhattacharyya, A shape preserving one-pass thinning algorithm and its behavioral analysis, *Advances in Modelling* & *Analysis, B*, 29(1994), 37–64.

[38] S. Pal, R. Bhattacharyya, S. Bhattacharyya and A Ray, A statistically distributed random binary image generator, *Indian J. Pure Appl. Math.*, 25(1994), 247–265.

[39] T. Pavlidis, A thinning algorithm for discrete images, *Comput. Graphics and Image Processing*, 13(1980), 142–157.

[40] C. Pudney, Distance-ordered homotopic thinning: a skeletonization algorithm for 3D digital images, *Computer Vision and Image Understanding*, 72(1998), 404–413.

[41] A. Rosenfeld, A Characterization of parallel thinning algorithms. *Info. Contr.*, 29(1975), 286–291.

[42] J. Shen and S. Castan, Image thinning by neural networks, *Artificial Neural Networks*, ed T. Kohonen, K. Makisara, O. Simula, J. Kangas, Elsevier Science Pub, North-Holland, 1991, 841–846.

[43] R. Singh, V. Cherkassky and N. Papanikolopoulos, Self-organizing maps for the skeletonization of sparse shapes, *IEEE Transactions on Neural Networks*, 11(2000), 241–248

[44] R. Stefanelli and A. Rosenfeld, Some parallel thinning algorithms for digitial pictures, *J. ACM*, 18(1971), 155–264.

[45] R. Syski, Random Process,Vol 98, *Marcel Dekker Inc.*, New York and Basel, 2ed, 283–294, 1989.

[46] H. Tamura and K. Mori, A parallel thinning algorithm for binary pictures and its connectivity, 1974 National Convention Record of IECEJ No 1539, p. 1390.

[47] W. Xu and C. Wang, CGT : A fast thinning algorithm implemented on a sequential computer, *IEEE SMC*, 17(1987), 847–851.

[48] S.-S. Yu and W.-H.Tsai, A new thinning algorithm for gray scale images by the relaxation technique, *Pattern Recog.*, 23(1990), 1067–1076.

[49] T. Y. Zhang and C. Y. Suen, A fast parallel algorithm for thinning digital patterns, *Comm. ACM*, 27(1984), 236–239.

[50] R. W. Zhou, C. Quek, and G. S. Ng, A novel single-pass thinning algorithm and an effective set of performance criteria, *Pattern Recog. Letters*, 16(1995), 1267–1275.