ON THE LIMIT CYCLE STRUCTURE OF THRESHOLD BOOLEAN NETWORKS OVER COMPLETE GRAPHS

GEORGE VOUTSADAKIS*
School of Mathematics and Computer Science,
Lake Superior State University, 650 W. Easterday Avenue,
Sault Sainte Marie, MI 49783, USA

Received 29 November 2002
Revised 25 August 2003
Accepted 7 April 2004

In previous work, the limit structure of positive and negative finite threshold boolean networks without inputs (TBNs) over the complete digraph $K_n$ was analyzed and an algorithm was presented for computing this structure in polynomial time. Those results are generalized in this paper to cover the case of arbitrary TBNs over $K_n$. Although the limit structure is now more complicated, containing, not only fixed-points and cycles of length 2, but possibly also cycles of arbitrary length, a simple algorithm is still available for its determination in polynomial time. Finally, the algorithm is generalized to cover the case of symmetric finite boolean networks over $K_n$.

Keywords: Finite Boolean Networks; Random Boolean Networks; Finite Automata Networks; Neural Networks; Threshold Boolean Networks; State Space; Fixed Points; Limit Cycles; Algorithm for Limit Cycles.

1991 AMS Subject Classification: Primary: 93A05, 92B20, Secondary: 68-04.

1. Introduction

Finite Threshold Boolean Networks without inputs (TBNs) are special cases of the Random Boolean Networks (RBNs) (also called Kauffman nets) that were introduced in Ref. 15. See also Ref. 18 for a more recent exposition. An RBN is a system of $N$ automata with two possible states of a Boolean variable. Each of the automata is connected randomly with exactly $K$ neighbors. The state of each automaton is updated by means of a Boolean function, also randomly chosen among all Boolean functions with $K$ arguments. Once the choice of the neighborhoods and the functions have been made, they remain fixed. TBNs are the special cases of RBNs where all randomness has been removed and the functions are taken to be Boolean threshold functions. RBNs have been extensively studied with respect to many of their properties. For instance, Refs. 4 and 5 study some statistical properties of their behavior. Kauffman, in Refs. 16 and 17, observed that RBNs may exhibit either an orderly or a chaotic behavior depending on the value of the parameter $K$, determining the connectivity of the network. The critical value of that parameter was numerically discovered by Kauffman and, later, analytically determined in Refs. 7 and 8. Other aspects of the behavior of RBNs

*This research was supported by a grant from the U.S. Department of Defense.
that have been studied in the literature include the quantification of the mutual information they contain at the order-disorder phase transition,\textsuperscript{19} control of the chaotic phase in RBNs,\textsuperscript{20,21} and evolution of their topology with time.\textsuperscript{9} Many other variants of RBNs have also been considered,\textsuperscript{12,13} including versions where external inputs are also allowed.\textsuperscript{1} Boolean networks are also special cases of Finite Automata Networks,\textsuperscript{11–9} in which local automata may each have any number of states.

A \textbf{Finite Boolean Network} (FBN) \( N = \langle G, \{f_i\}_{i \in V} \rangle \) (see also Ref. 14) consists of a digraph \( G = (V, E) \) together with a collection \( \{f_i\}_{i \in V} \) of functions \( f_i : \{0, 1\}^V \rightarrow \{0, 1\}, i \in V \), such that \( f_i \) only depends on those \( j \), such that \( \langle j, i \rangle \in E \). The \( f_i \)'s are called the \textbf{local update functions} of the FBN. The \textbf{global update function} \( f : \{0, 1\}^V \rightarrow \{0, 1\}^V \) of the FBN \( N \) is the function given by

\[
f(x)_i = f_i(x), \quad \text{for all } x \in \{0, 1\}^V, \quad i \in V.
\]

\( N \) is said to be a \textbf{symmetric FBN} if \( f_i \) is a \textbf{symmetric function} in those \( j \)'s with \( \langle j, i \rangle \in E \), for all \( i \in V \), i.e., if \( f_i \) is invariant under permutations of the inputs or, equivalently, if it depends only on the number of 1's among its arguments.

The \textbf{state space} \( S(N) \) of the FBN \( N \) is the digraph with set of vertices \( \{0, 1\}^V \) and edges all pairs \( \langle x, y \rangle \in (\{0, 1\}^V)^2 \), such that \( y = f(x) \). A point \( x \) is said to be a \textbf{fixed-point} if \( x = f(x) \) and a sequence of points \( x_1, \ldots, x_m \) is said to form a \textbf{limit cycle} of length \( m \) if, for all \( 1 \leq i \leq m - 1 \), \( x_{i+1} = f(x_i) \) and \( x_1 = f(x_m) \). Thus fixed points are limit cycles of length 1. All points in limit cycles are collectively termed \textbf{limit points}.

The focus in this paper will be on a special class of FBNs. This class is a subclass of neural or threshold networks\textsuperscript{14} and it was introduced in Ref. 22 (under a different name; see below) as an alternative platform to the sequential dynamical systems,\textsuperscript{2,3} for modelling and analytically studying properties of computer simulations.

A \textbf{Finite Threshold Boolean Network without inputs} (TBN) (introduced in Ref. 22 under the name Threshold Agent Network) is an FBN \( A = \langle G, \{f_i\}_{i \in V} \rangle \), whose functions \( f_i \) are integer threshold functions, i.e., \( f_i, i \in V \), is determined by an integer \( t_i \), in the following way, for all \( x \in \{0, 1\}^V \),

\[
f_i(x) = \begin{cases} 1, & \text{if } \left| \{ j : \langle j, i \rangle \in E \text{ and } x_j = 1 \} \right| \geq t_i \\ 0, & \text{otherwise} \end{cases}.
\]

if \( t_i \geq 0 \), and

\[
f_i(x) = \begin{cases} 0, & \text{if } \left| \{ j : \langle j, i \rangle \in E \text{ and } x_j = 1 \} \right| \geq -t_i \\ 1, & \text{otherwise} \end{cases}.
\]

if \( t_i < 0 \).

Note that negative thresholds are also allowed which correspond to inhibitory rather than exciting behavior.

Since the \( f_i \)'s are completely determined by the thresholds \( t_i \), the TBN \( A \) is most often denoted by \( A = \langle G, t \rangle \), where \( t = \{ t_i : i \in V \} \) is the sequence of integer thresholds. A TBN \( A \) is said to be \textbf{positive} if, for all \( i \in V \), \( 0 \leq t_i \leq |V| \) and it is said to be \textbf{negative} if, for all \( i \in V \), \( -|V| \leq t_i \leq -1 \).

In Ref. 23 the limit cycle structure of positive and negative TBNs over the complete digraph \( K_n \) was determined and a polynomial algorithm was provided for computing it. More specifically, it was shown that positive TBNs over \( K_n \) have only fixed-points, i.e., no limit cycles of length greater than 1, and a formula for the number of these fixed-points was given. In the case of negative TBNs, it was shown that they only have limit cycles of lengths 1 and 2 and formulas were also given for computing their number in polynomial time. These results are interesting because they provide polynomial time prediction tools for the number of limit cycles, whereas the brute force approach of computing the entire state space obviously needs exponential time. Thus, even though the entire state space computation works relatively fast in an up-to-date personal computer for up to approximately 20+ vertex TBNs, from then on, it is hopeless to compute the limit cycle structure without a polynomial time prediction tool.

In Sec. 2, it is shown that the limit cycle structure of arbitrary TBNs over the complete digraph \( K_n \) is more complicated than the ones of positive and negative TBNs. Namely, for each \( k \), a TBN is constructed over \( K_{2k+2} \) whose state space contains a
On the Limit Cycle Structure of Threshold Boolean Networks

2. TBNs over $K_n$ with Arbitrarily Large Limit Cycles

It was shown in Ref. 23 that positive TBNs over $K_n$ have only fixed-points and that negative TBNs over $K_n$ have only fixed-points and cycles of length 2, but no limit cycles of length 3 or greater. In this section, given a positive integer $k \geq 4$, a TBN is constructed over $K_{2k+2}$ that has a limit cycle of length $k$. In particular, this shows that TBNs over $K_n$ may have limit cycles of arbitrarily large length.

Let $k \geq 4$ be a positive integer. Let $A$ be the TBN over $K_{2k+2}$ which has the following sequence of thresholds

$$t = \langle -2k, -2k+1, \ldots, -2k+1, \underbrace{k+1, k+2, \ldots, 2k-1, 2k, 2k}_{k-1} \rangle.$$

It is not difficult to check that the state space of this TBN contains the following limit cycle of length $k$

\[
\begin{array}{c}
\text{11} \ldots \text{100} \ldots \text{0} \\
\downarrow \\
\text{11} \ldots \text{1} \text{00} \ldots \text{0} \\
\downarrow \\
\vdots \\
\downarrow \\
\text{1} \text{1} \ldots \text{1} \text{00} \\
\downarrow \\
\text{1} \text{00} \ldots \text{01} \text{1} \ldots \text{1} \text{00} \\
\downarrow \\
\text{1} \text{1} \ldots \text{1} \text{00} \\
\downarrow \\
\text{1} \text{1} \ldots \text{1} \text{00} \ldots \text{0}
\end{array}
\]

which proves the assertion.

For a concrete example of the construction, consider the case $k = 5$ and the TBN over $K_{12}$ with sequence of thresholds

$$t = \langle -10, -9, -9, -9, -9, -9, 6, 7, 8, 9, 10, 10 \rangle.$$

The cycle above is, in this case, the 5-cycle

\[
\begin{array}{c}
\text{111111000000} \\
\downarrow \\
\text{111111000000} \\
\downarrow \\
\text{111111100000} \\
\downarrow \\
\text{111111110000} \\
\downarrow \\
\text{100000111100} \\
\downarrow \\
\text{111111000000}
\end{array}
\]

3. Limit Cycles of TBNs over $K_n$

In this section, a linear time algorithm is provided for computing the number of limit cycles of each length in the state space of a TBN over $K_n$, given the sequence $t = \langle t_i : i \in V \rangle$ of its thresholds.

The key observation that validates this algorithm is that, given the current state of a TBN over $K_n$, the next state is uniquely determined by the number of 1’s in the current state. Thus, in every limit cycle (with length at least 2), no two states may have
June 24, 2004 14:25 00194

reason, even different limit cycles may not contain states with the same number of 1’s, since these may belong to the same connected components of the state space. Finally, noting that the number of 1’s in a state equals the number of vertices with negative thresholds that are greater in absolute value than the number of 1’s in the previous state plus the number of vertices with positive thresholds that are less than or equal to the number of 1’s in the previous state, the following algorithm computes the number of limit cycles of each length in the state space of a TBN over $K_n$.

In this algorithm three array structures $N$, $S$ and $Next$ will be used. $N[i]$ will contain the number of vertices whose threshold value is equal to $t$, for $t = -n$ to $n$. $S[i]$ will contain, for $t$ negative, the number of vertices whose thresholds are less than or equal to $t$ and, for $t$ non-negative, the number of vertices whose thresholds are non-negative and less than or equal to $t$. Finally, $Next[i]$ denotes the number of 1’s that are contained in the state succeeding a state containing $i$ 1’s. Pseudo-code for the algorithm follows:

Algorithm for Computing the Limit Cycle Structure of a TBN over $K_n$

Input: Sequence of thresholds $t = \langle t_1, t_2, \ldots, t_n \rangle$, with $-n \leq t_i \leq n$, for all $i = 1, 2, \ldots, n$.

// $N[i]$ is set to contain the number of vertices whose threshold value is equal to $t$, for $t = -n$ to $n$. So $N[i]$ is initialized to 0 and then increased by 1 each time a threshold is found whose value is $t$. //
For $i = -n$ to $n$, $N[i] := 0$;
For $i = 1$ to $n$, $N[i] := N[i] + 1$;

// $S[i]$ is set to contain, for $t$ negative, the number of vertices whose thresholds are less than or equal to $t$ and, for $t$ non-negative, the number of vertices whose thresholds are non-negative and less than or equal to $t$. //
$S[0] := N[0]; S[-n] := N[-n]$;
For $i = -n + 1$ to $-1$, $S[i] := S[i - 1] + N[i]$;
For $i = 1$ to $n$, $S[i] := S[i - 1] + N[i]$;
// For $i = 0, 1, \ldots, n$ $Next[i]$ denotes the number of 1’s that are contained in the state succeeding a state containing $i$ 1’s.

Note that this is a well-defined function, since we are dealing with a TBN over $K_n$. //
For $i = 0$ to $n - 1$, $Next[i] := S[i] + S[-i - 1]$;
$Next[n] := S[n]$;

Output: Output the number of limit cycles of each length of the finite dynamical system over $0, \ldots, n$ with dynamics function $Next$.

Based on the observations listed at the beginning of the current section, it is not difficult to check that this algorithm correctly enumerates the number of limit cycles of each length of the TBN with sequence of thresholds $T = \langle t_1, \ldots, t_n \rangle$, with $-n \leq t_i \leq n$, for all $i = 1, \ldots, n$. Moreover, because only unnested for-loops over the number of vertices occur with bodies of constant time complexity, the algorithm can be carried out in linear time in the size of the input. This is significant because the limit cycle structure may be predicted in linear time, whereas “running” the TBN would obviously require exponential time in terms of the input.

For a concrete example consider the TBN $A$ over $K_8$ with sequence of thresholds $t = \langle -2, -1, -1, -1, 3, 5, 6 \rangle$.

Then, the following table shows the contents of the arrays $N, S$ and $Next$ after the execution of the algorithm.

<table>
<thead>
<tr>
<th>$i$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N[i]$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$S[i]$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$Next[i]$</td>
<td></td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

The state space $\{0, 1, \ldots, 8\}$ of the finite dynamical system with function $Next$ is given in Fig. 1. It has a single fixed-point and a limit cycle of length 3. This entails that the state space of the original TBN $A$ also has a single fixed-point and a limit cycle of length 3. A complete picture of that state space is
depicted in Fig. 2. The limit points are the black points in the figure.

4. Limit Cycles of Symmetric FBNs over $K_n$

The basic idea of the algorithm of Sec. 3 may be used to provide an algorithm for computing the number of limit cycles of each length of any symmetric FBN over $K_n$. This generalizes the case dealt with in Sec. 3, since threshold functions are obviously symmetric functions but the converse statement is not true in general. The following is an algorithm that covers the case of FBNs over $K_n$ with symmetric local update functions. Note that such a function $f_i$, $1 \leq i \leq n$, may be efficiently represented by a binary array $\langle t_{ij} : 0 \leq j \leq n \rangle$ of length $n + 1$ that contains in its $j$th position the value of the function when exactly $j$ of its $n$ input variables are equal to 1, $0 \leq j \leq n$.

Algorithm for Computing the Limit Cycle Structure of a Symmetric FBN over $K_n$

Input: An $n \times (n + 1)$ binary array $\langle t_{ij} : 1 \leq i \leq n, \ 0 \leq j \leq n \rangle$ with $t_{ij}$ being the value of $f_i$ when exactly $j$ of its input variables are equal to 1.

// $N[i]$ is set to contain the number of vertices whose local update functions take the value 1 when exactly $t$ of their input variables have the value 1, for $t = 0$ to $n$. So $N[i]$ is initialized to 0 and then increased by 1 each time a local update function is found whose value is 1 when exactly $t$ of its input variables have the value 1. //

For $i = 0$ to $n$, $N[i] := 0$;
For $j = 0$ to $n$, for $i = 1$ to $n$, if $t_{ij} = 1$ then $N[j] := N[j] + 1$;

// For $i = 0, 1, \ldots, n$ Next[i] denotes the number of 1’s that are contained in the state succeeding a state containing $i$ 1’s. Note that this is a well-defined function, since we are dealing with a Symmetric FBN over $K_n$. //

For $i = 0$ to $n$, Next[i] := $N[i]$;

Output: Output the number of limit cycles of each length of the finite dynamical system over $0, \ldots, n$ with dynamics function Next.

This algorithm correctly computes the number of limit cycles of each length of a FBN over $K_n$ with symmetric local update functions. Note that the algorithm in this case is conceptually as simple as in the case of TBNs over $K_n$, but the price for its increased applicability is that it runs in quadratic rather than in linear time in terms of the number $n$ of vertices, although it is still linear in terms of the input length. This is because, in the present case, the input, being more general, requires quadratic space in terms of the number of vertices for its representation. Nevertheless, it is still a polynomial prediction algorithm as opposed to the exponential algorithm that computes the entire state space in detail.

For a concrete example, consider the symmetric FBN over $K_5$ with the symmetric local update
functions given in the following table in the form of the arrays \( t_{ij} : 0 \leq j \leq 5 \), \( 1 \leq i \leq 5 \).

\[
\begin{array}{c|cccccc}
 & 0 & 1 & 2 & 3 & 4 & 5 \\
\hline
f_1 & 0 & 1 & 0 & 1 & 0 & 0 \\
f_2 & 1 & 1 & 0 & 0 & 1 & 0 \\
f_3 & 0 & 1 & 0 & 1 & 0 & 1 \\
f_4 & 0 & 0 & 0 & 1 & 1 & 1 \\
f_5 & 1 & 1 & 0 & 0 & 1 & 0 \\
\end{array}
\]

After running the algorithm on this symmetric FBN, the values of \( N \) and \( \text{Next} \) are as follows

\[
\begin{array}{c|cccccc}
 & 0 & 1 & 2 & 3 & 4 & 5 \\
\hline
i & 0 & 1 & 2 & 3 & 4 & 5 \\
N[i] & 2 & 4 & 0 & 3 & 3 & 2 \\
\text{Next}[i] & 2 & 4 & 0 & 3 & 3 & 2 \\
\end{array}
\]

The state space of the finite dynamical system with function \( \text{Next} \) is given in Fig. 3.

It has a single fixed-point and a limit cycle of length 2. Thus, the state space of the original symmetric FBN also has a single fixed-point and a limit cycle of length 2. It is depicted in Fig. 4. The labels in the figure are meant to exhibit the relation of this actual state space to the state space output by the algorithm and depicted in Fig. 3. They show how many 1’s are contained in the states at each level of the figure.

Finally, we note that the algorithm developed above may be used for computing in polynomial time the numbers of limit cycles of each length of a sequential dynamical system (SDS) over the complete graph in which all functions are symmetric functions of the input. Note that this requirement was present in the original definition of SDSs. Thus, the algorithm that was provided in this paper, appropriately modified for SDSs, would provide a complete solution to the polynomial prediction problem of the limit structure of an SDS over the complete graph. For more details on SDSs the interested reader is referred to the papers in Refs. 2 and 3.

Acknowledgments

The author wishes to thank Dr. Abdallah Taha for his valuable help in setting up FBN and TBN experiments that inspired and verified the results contained in this paper. Thanks also to an anonymous referee that provided valuable corrections and comments and contributed in appropriately increasing the list of references.

References