Finding a Periodic Attractor of a Boolean Network

Tatsuya Akutsu[1], Sven Kosub[2], Avraham A. Melkman[3] and Takeyuki Tamura[1]

[1] Bioinformatics Center, Institute for Chemical Research, Kyoto University
[2] Department of Computer and Information Science, University of Konstanz
[3] Department of Computer Science, Ben-Gurion University of the Negev

Abstract

In this paper, we study the problem of finding a periodic attractor of a Boolean network (BN), which arises in computational systems biology and is known to be NP-hard. Since a general case is quite hard to solve, we consider special but biologically important subclasses of BNs. For finding an attractor of period 2 of a BN consisting of n OR functions of positive literals, we present a polynomial time algorithm. For finding an attractor of period 2 of a BN consisting of n AND/OR functions of literals, we present an $O(1.985^n)$ time algorithm. For finding an attractor of a fixed period of a BN consisting of n nested canalyzing functions and having constant treewidth w, we present an $O(n^{2p(w+1)}poly(n))$ time algorithm.
Keywords: Boolean network, periodic attractor, SAT, nested canalyzing function, treewidth.

1 Introduction

The *Boolean network* (BN) is known as a discrete mathematical model of gene regulatory networks. In a BN, the value of a node at a given time instant is determined according to a regulation rule that is a Boolean function of the values of the predecessors of the node at the previous time instant. The values of nodes are updated synchronously, and the (global) *state* of a network at a given time instant is the vector of its node values. An important characteristic of any BN is the existence of an *attractor*, which is classified into two types: a *singleton attractor* corresponding to a stable state, and a *periodic attractor* corresponding to a sequence of states that repeats periodically.

Since a correspondence between different steady states and different types of cells was pointed out [20], the analysis of steady states in biological networks has been an important research topic in bioinformatics and computational biology. Several heuristic algorithms have been proposed for the detection and/or enumeration of singleton and/or periodic attractors [8, 9, 13, 16, 23, 26], and extensive studies have been done on the distribution of the numbers and lengths of attractors [10, 25] in random BNs. From an algorithmic viewpoint, it is known that the problem of finding an attractor of the shortest period is NP-hard [1] even for BNs with maximum indegree 2 consisting of AND/OR of literals [27].

Due to this hardness and the fact that there exist 2^n global states for a BN with n nodes, previous studies focused on the development of $o(2^n)$ time algorithms. Although some of the above mentioned heuristic algorithms may work efficiently in practice, there is no proof that any of them works in $o(2^n)$ time in the worst case. In order to achieve such improved times some restrictions have to be imposed on the permitted types of Boolean functions because there are 2^{2^n} different Boolean functions with n inputs and thus 2^n bits are required just to specify a particular Boolean function. For example, it is known that detection of a singleton attractor for BNs with maximum indegree K can be transformed into $(K+1)$-SAT (Boolean satisfiability problem in which each clause consists of at most $K+1$ literals) [27]. For AND/OR BNs, in which each Boolean function is restricted to either a conjunction or a disjunction of literals, a succession of algorithms for the detection of a singleton attractor were

developed with the currently best algorithm working in $O(1.587^n)$ time [24, 27, 28]. Recently, Akutsu et al. developed an $O(1.799^n)$ time algorithm for BNs consisting of nested canalyzing functions [2], which are considered to cover most biologically important Boolean functions [15]. Aracena et al. developed a polynomial time algorithm for detection of a singleton attractor in a strongly connected BN without negative cycles [4], and Goles and Salinas developed an algorithm for detection of a singleton attractor in a general BN without negative cycles [14], which was further simplified in [3]. On the contrary, Just proved that detection of 2-periodic attractor is NP-hard for BNs without negative cycles [19]. Zhang et al. [30] developed recursive algorithms for the detection of singleton and periodic attractors of BNs with bounded indegree, and analyzed their average case time complexities for randomly generated BNs. Detection of a singleton attractor has also been studied in the context of discrete dynamical systems [5, 21]. However, to our knowledge there are no algorithms for the detection of a periodic attractor, even for AND/OR BNs, that have been proven to work in $o(2^n)$ time.

In this paper, we focus on the theoretical aspect of finding a periodic attractor of a given BN and present the following three algorithms:

(1) a polynomial time algorithm for finding a 2-periodic attractor of a BN whose regulation rules are OR-functions of non-negated variables,

(2) an $O(1.985^n)$ time algorithm for finding a 2-periodic attractor of an AND/OR BN,

(3) an $O(n^{2p(w+1)}poly(n))$ time algorithm for finding a p-periodic attractor of a BN having bounded treewidth w and consisting of nested canalyzing functions, where p and w are constants.

All of these three algorithms are based on the same strategy: transform the problem of finding a p-periodic attractor of the given BN with n nodes into the problem of finding a singleton attractor of a BN with pn nodes (with some constraint). However, simply applying existing algorithms for the detection of a singleton attractor of the transformed network does not lead to an $o(2^n)$ time algorithm. For example, the current best algorithm for detecting a singleton attractor of an AND/OR BN requires $O(1.587^n)$ time, and thus using it on the transformed network for finding a 2-periodic attractor yields an $O(1.587^{2n}) = O(2.519^n)$ time algorithm, which is much worse than $O(2^n)$. Therefore, in these algorithms, we make use of the special properties of the transformed networks.

2 Preliminaries

A BN $N(V, F)$ consists of a set V of n nodes and a corresponding set $F = \{f_v : v \in V\}$ of n non-constant Boolean functions. Let $v(t) \in \{0, 1\}$ represent the value of a node v at time t, and denote by $\mathbf{v}(t) = \langle v(t) : v \in V \rangle$ the *state of the network* at time t. The values of all nodes are updated simultaneously according to the corresponding Boolean functions, $v(t + 1) = f_v(\mathbf{v}(t))$. A directed graph $G = (V, E)$ can be associated with the network, with a directed edge $(u, v) \in E$ if and only if f_v depends on u. Edges may be self-loops because many biological networks contain self-loops. The initial assignment of values $\mathbf{v}(1) = \langle v(1) : v \in V \rangle$ uniquely determines the state of the network at all $t > 1$. The dynamics of a BN is well represented by a *state transition diagram*, which is a directed graph defined by a set of nodes corresponding to 2^n possible states of the network and a set of edges corresponding to the transitions from $\mathbf{v}(t)$ to $\mathbf{v}(t + 1)$. An initial state is called a *periodic attractor with period* p if $\mathbf{v}(1) = \mathbf{v}(p+1)$ and $\mathbf{v}(1) \neq \mathbf{v}(q)$ holds for all $1 < q < p+1$. An attractor with period $p = 1$ is called a *singleton attractor*.

Example 1 *A BN with regulation rules* $v_1(t + 1) = v_2(t)$, $v_2(t + 1) = v_1(t) \wedge \overline{v_3(t)}$ *and* $v_3(t + 1) = v_1(t) \wedge \overline{v_2(t)}$ *is shown in Fig. 1 (A) and its state transition diagram is shown in Fig. 1 (B). This network has two singleton attractors,* $\langle 0, 0, 0 \rangle$ *and* $\langle 1, 1, 0 \rangle$*, and one attractor of period 2,* $\langle 0, 1, 1 \rangle$ *(or, equivalently* $\langle 1, 0, 0 \rangle$*).*

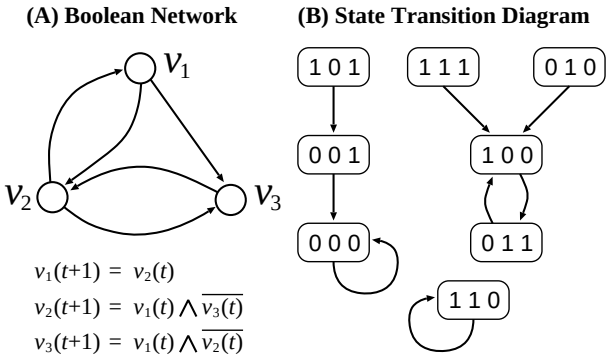

Figure 1: Example of a Boolean network and its state transition diagram.

The periodic attractor detection problem is defined as follows.

Problem 1 (Detection of Periodic Attractor)
Given a BN and a period p, decide whether or not there exists an attractor of period p, and output one if it exists.

As mentioned in Section 1, we consider some subclasses of BNs. The various classes will all be special cases of the class of sign-definite Boolean functions, which is characterized by the fact that when a function in the class is written in disjunctive normal form each variable occurs either only non-negated or only negated, if it appears at all.

If all regulation rules are sign-definite, then each edge (u, v) can be assigned a positive or negative sign $\sigma(u, v)$, depending on whether u appears non-negated or negated, respectively, in the disjunctive normal form of f_v. Upon doing so the associated graph becomes a *signed graph* (G, σ).

One subclass of BNs comprises the AND/OR BNs, defined as BNs in which every f_v is either an AND or an OR of literals. Seemingly even more restricted are OR BNs, defined as BNs in which each regulation rule is an OR of literals. It is known however that each AND/OR BN can be transformed into an OR BN of the same size in polynomial time when considering attractors [24].

Another subclass of BNs we consider is characterized by regulation rules that are *nested canalyzing* [20], *nc-function*s for short, which are biologically of interest [15].

We use the following fact as the definition of a nested canalyzing function.

Fact 1 ([18]) *A Boolean function is nested canalyzing over $\langle v_1, ..., v_k \rangle$, if and only if it can be represented as*

$$f_v \;=\; \ell_1 \vee \cdots \vee \ell_{k_1-1} \vee (\ell_{k_1} \wedge \cdots \wedge \ell_{k_2-1} \wedge (\ell_{k_2} \vee \cdots \vee \ell_{k_3-1} \vee (\cdots))),$$

where ℓ_i is either v_j or $\bar{v}_j$, and $1 \leq k_1 < k_2 < \cdots$.

We call BNs consisting of nc-functions *nc-BN*s. From this definition, one can see that any AND/OR BN is also an nc-BN. Following the discussion in [2], we assume without loss of generality that each variable appears at most once in any nc-function. In this paper we assume that nc-functions are given in the above form, because if the representation is not hierarchical (e.g. if it is in conjunctive normal form or in the form of a truth table) the representation may itself take space that is exponential in the number of nodes. Moreover, the algorithm for obtaining the hierarchical representation of an nc-function from its truth table takes an exponential number of table lookups [22].

As mentioned in Section 1, the first step of all algorithms in this paper is to transform a periodic attractor detection problem into a singleton attractor detection problem. Here, we describe the method

of transformation. From a given BN $N(V,F)$ with n nodes, we construct a *p-multiplied network* $N^p(V^p,F^p)$ by

$$\begin{aligned} V^p &= \{v(1),v(2),\cdots,v(p) \mid v\in V\},\\ F^p &= \{f_{v(1)}(v_1(p),\ldots,v_k(p)),\ f_{v(2)}(v_1(1),\ldots,v_k(1)),\cdots,\ f_{v(p)}(v_1(p-1),\ldots,v_k(p-1)) \\ &\quad \mid f_v(v_1,\ldots,v_k)\in F \text{ and } f_{v(t)}=f_v\}, \end{aligned}$$

where each $v(t)$ is regarded as a node. Finding a p-periodic attractor of N is then equivalent to finding a singleton attractor of $N^p(V^p,F^p)$ (with pn nodes) under the constraint that for all $1<q<p+1$, $v(1)\neq v(q)$ holds for some v. Since we only need to examine $O(n^{(p-1)})$ cases of constraints, we have

Proposition 2 *The p-periodic attractor detection problem can be solved in $O((1+\delta)^{pn})$ time if the singleton attractor detection problem can be solved in $O((1+\delta')^n)$ time for any $\delta'<\delta$, where p, δ, and δ' are constants.*

When discussing the detection of a singleton attractor, we will write $v=1$ to denote an assignment of 1 to v.

3 Finding a 2-Periodic Attractor of a Positive OR-BN

Note that an OR-network whose signed graph contains only negative edges always has a trivial 2-periodic attractor, namely one in which all nodes have the value 0 at time 0 and the value 1 at time 1 (or vice versa). In this section we observe that the problem of finding a 2-periodic attractor is also easy for a network whose signed graph has only positive edges, a positive OR-network. In [3] we proved that for a general positive sign-definite network a sufficient condition for the existence of a 2-periodic attractor is that its associated graph G contains a *source* strongly connected component that is bipartite and of size 2 at least, while a necessary condition is the existence of a cycle of even length. For positive OR-networks, however, there is a condition that is both necessary and sufficient.

Theorem 3 *A positive OR-network has a 2-periodic attractor if and only if its associated graph G contains a strongly connected component that is bipartite and of size 2 at least. It takes therefore polynomial time to determine whether a positive OR-network has a 2-periodic attractor, and if so to construct one.*

We outline first how the necessity of the condition can be deduced from [17]. The following Lemma paraphrases the relevant parts of their Lemma 2.11.

Lemma 4 *Call the greatest common divisor of the lengths of all cycles of the non-singleton strongly connected component C of G the* period of C. *Denote by $p(G)$ the least common denominator of the periods of the non-singleton strongly connected components of G. Then the period of any attractor of the network divides $p(G)$.*

According to the Lemma, if the network has a 2-periodic attractor then $p(G)$ is even. Thus the period of at least one non-singleton strongly connected component C is even, meaning that all its cycles have even length. As is well-known, this implies that C is bipartite.

We show next that the necessary condition of Theorem 3 is also sufficient to ensure the existence of a 2-periodic attractor. To this end we first detail an algorithm for constructing a 2-periodic attractor, given that the condition holds, and then argue its correctness. Our construction is similar to the more general construction encapsulated in Definition 5.4 of [17] of what they call a J-regular attractor for

some maximal antichain J of strongly connected components. The difference is that we have to deal with the case that J is not necessarily maximal and consists of a single (bipartite) component.

Algorithm PosOR2p
precondition: G contains a strongly connected component that is of size 2 at least and bipartite, with parts X and Y.

1. Build the network N^2.
2. Pick a node $s \in X$ (with copies $s(1), s(2) \in V^2$).
3. Assign the value 0 to $s(1)$ and to all its immediate and distant predecessors in G^2.
4. Assign the value 1 to $s(2)$ and to all its immediate and distant successors in G^2.
5. Assign the value 0 any node of G^2 that does not have a value yet although all of its predecessors have the value 0.
6. Assign all the remaining nodes of G^2 the value 1, completing the singleton attractor of N^2.
7. Convert the singleton attractor of N^2 into a 2-periodic attractor of N.

Note that STEP 3 assigns 0 to $x(1)$ for each $x \in X$, and to $y(2)$ for each $y \in Y$, and STEP 4 assigns 1 to $x(2)$ for each $x \in X$, and to $y(1)$ for each $y \in Y$.

The following lemma completes the proof of Theorem 3.

Lemma 5 *If the graph G associated with a positive OR-network contains a strongly connected component that is bipartite and of size 2 at least, then Algorithm PosOR2p constructs a 2-periodic attractor of N.*

(Proof) We prove first of all that the values assigned in steps 3 and 4 of the algorithm are consistent. Suppose to the contrary that there is a node $v \in V^2$ that is assigned both the value 0 and the value 1. That means there is a path from $s(1)$ to v, and a path from v to $s(0)$, and hence there is a path from $s(1)$ to $s(0)$. Such a path corresponds to a cycle in G that is of odd length, and in which s participates. Thus the strongly connected component of s contains an odd-length cycle, contrary to the precondition of the algorithm.

We prove next that the values assigned in these two steps can form the core of a singleton attractor of N^2, because for each node $v \in V^2$ whose value is assigned, that value equals the OR-value of its predecessors. This fact is trivially true for the case of a node whose assigned value is 0, and in particular for $s(1)$, since the algorithm assigns the value 0 to all its predecessors. Consider next $s(2)$. The choice of s as a node from a bipartite connected component ensures that it is on an even-length cycle, say $s, v_2, \ldots, v_{2\ell}$, so that STEP 4 of the algorithm assigns 1 to $v_2(1), v_3(2), \ldots, v_{2\ell}(1)$. Since $v_{2\ell}(1)$ is a predecessor of $s(2)$ in G^2, the value 1 assigned to $s(2)$ indeed equals the OR-value of its predecessors. Any other node of G^2 to which the algorithm assigns the value 1 in STEP 4 has a predecessor with value 1 because there is a path to it from $s(2)$.

After execution of STEP 5 any node that does not have a value assigned yet must have a predecessor that does not have a value assigned either (or else it would have been assigned 0 in STEP 4). Thus assigning the value 1 to all these nodes indeed completes the construction of a singleton attractor. □

4 Finding a 2-Periodic Attractor of an AND/OR BN

In this section, we present an $O(1.985^n)$ time algorithm for the detection of a 2-periodic attractor in an AND/OR BN.[1] From the discussions in Section 2, we focus on the singleton attractor detection of an OR BN $N^2(V^2, F^2)$ with $2n$ nodes and denote its associated signed graph by (G^2, σ). It is easily seen that G^2 is a bipartite graph without self-loops (even if G did have self-loops). Let V_1^2 and V_2^2 be the sets of nodes corresponding to $t = 1$ and $t = 2$ respectively, which give a bipartition of V^2.

The basic strategy for finding a singleton attractor in $N^2(V^2, F^2)$ is similar to that in [24] although the details are very different and novel ideas are introduced here. Let $U(v)$ denote the number of unassigned neighboring nodes of v. The following is a high-level description of the algorithm, where K is a parameter to be determined later.

Algorithm AND-OR2p

1. Construct $N^2(V^2, F^2)$. Let all nodes be unassigned.
2. Recursively examine 0-1 assignments on unassigned nodes v with $U(v) \geq 3$ until there does not exist such a node or the number of assigned nodes is more than K.
3. Let A be the set of assigned nodes. Let $A_1 = A \cap V_1^2$ and $A_2 = A \cap V_2^2$.
   Without loss of generality assume that $|A_1| \geq |A_2|$.
4. If $|A| > K$, then examine all possible assignments on $V_1 - A_1$.
5. Otherwise, recursively examine assignments on paths and cycles and then solve SAT.

In this algorithm, we propagate the assignment whenever a new assignment (to a node) is given, where 'propagate' means that we assign Boolean values to a set of nodes to which an assignment is uniquely determined from the current partial assignment (see [2, 24] for the details of propagation).

Hereafter, we explain the details of each step. Since STEP 1 is trivial, we begin with STEP 2. For example, consider a node v shown in Fig. 2. If we assign 0 to v, in order to have a singleton attractor, assignments on all three neighboring nodes are uniquely determined and two additional constraints must be satisfied. If we assign 1 to v, no additional assignment is given but one additional constraint must be satisfied. As seen from Fig. 2, these constraints are given as clauses. The algorithm keeps all generated clauses until the final stage of STEP 5, at which Yamamoto's SAT algorithm [29] is applied to solve all constraints simultaneously (per final recursive step).

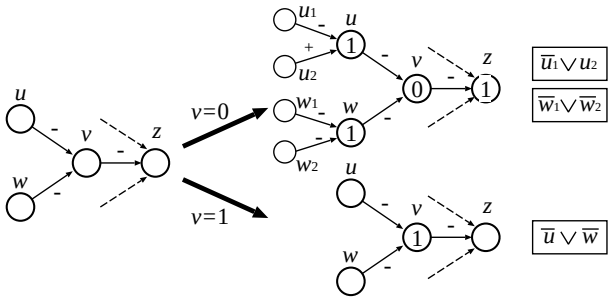

Figure 2: Elimination of an unassigned node with three unassigned neighbors.

If we consider all cases on $U(v) \geq 3$, we can see that either one of the following holds:

[1] It should be noted that polynomial factors are hidden in $O(a^n)$ notation in this paper because the precise value of a is slightly smaller than a.

- At least four nodes are assigned in one case and at least one node is assigned in the other case,
- At least three nodes are assigned in one case and at least two nodes are assigned in the other case.

We can also see that the number of additional constraints is at most k if k nodes are newly assigned.

Let $h(k)$ denote the maximum number of cases generated by recursive execution of STEP 2 under a condition that at most k nodes are assigned (per case). Then, we have

$$h(k) \leq \max \left\{ \begin{array}{l} h(k-1) + h(k-4), \\ h(k-2) + h(k-3). \end{array} \right.$$

By solving the following equations

$$\begin{aligned} x^4 &= x^3 + 1, \\ x^3 &= x + 1, \end{aligned}$$

and taking the larger solution, we have $h(k) = O(1.381^k)$. Since the number of assigned nodes per recursion is bounded by K, we have

Proposition 6 *The number of times that STEP 4 (resp. STEP 5) is executed is* $O(1.381^K)$.

Analysis of STEP 4 is straight-forward. If STEP 4 is executed, we can see from the bipartiteness of $N^2(V^2, F^2)$ that detection of 2-periodic attractor can be done correctly (under the partial assignment given by STEP 2) in $O(2^{n-(K/2)} \cdot poly(n))$ time per case.

Proposition 7 *STEP 4 works in* $O(2^{n-(K/2)} \cdot poly(n))$ *time per execution.*

STEP 5 is a rather complicated part of the algorithm. The following proposition is straight-forward because STEP 2 recursively assigns any node having at least three non-assigned neighbors.[2]

Proposition 8 *After STEP 2, the graph induced by non-assigned nodes is a set of paths and cycles (with bidirectional edges) in which every node has indegree greater than 0.*

Based on this proposition, we eliminate paths and cycles. We present a procedure for elimination as a part of the proof of the following lemma.

Lemma 9 *STEP 5 works in* $O(1.338^{2n})$ *time per execution.*

(Proof) First, we explain the basic elimination strategy for handling bidirectional edges, where elimination means assigning values to nodes and generating constraints (i.e., clauses). Suppose that there is a bidirectional edge (u, v) whose signs are (+,+) and there is a unidirectional edge from w to v as shown in Fig. 3 (a1). In this case, it is enough to examine $u = v = 0$ and $u = v = 1$ because other assignments do not lead to a singleton attractor. Furthermore, w is uniquely determined when $u = 0$. Other types of edges between v and w can be handled similarly. Suppose that (u, v) is (-,+) as shown in Fig. 3 (a2). Then, $v = 1$ must hold because otherwise u becomes 1 which makes $v = 1$. In addition, one constraint (i.e., one clause) is generated. For example, suppose that (w, v) is (+,+). Then, in order to satisfy $v = 1$, $u = 1$ or $w = 1$ must hold. Therefore, a clause $u \vee w$ is generated. Let $f(k)$ be the exponential factor of the time complexity for k nodes. The time complexities for k nodes in the above two cases are as shown in Table 1. Here the factor of 1.234 in Table 1 stems from the fact that SAT with k clauses can be solved in $O(1.234^k)$ time [29]. Furthermore, if a cycle contains an edge of

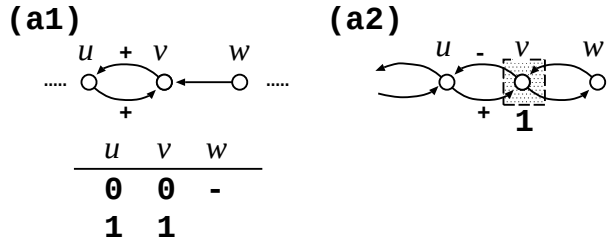

Figure 3: Types of basic elimination. Each bidirectional edge is represented by two directed edges. '-' in (a1) means that the value of w is uniquely determined.

Table 1: Complexity for (+,+) and (+,-) bidirectional edges.

type of path	#ASS	#MAXCL	complexity
(a1)	2	0	$f(k-2)+f(k-3)$
(a2)	1	1	$1.234 \cdot f(k-1)$

#ASS and #MAXCL denote the number of assignments and the maximum number of added clauses.

type (+,+) or a (+,-), the cycle will be decomposed into a path. Hereafter, we can assume that all bidirectional edges in chains or cycles of lengths 3 at least are of the type (-,-).

A path of length 1 necessarily has a bidirectional edge, since all nodes have at least one in-edge, see Fig. 4 (b1). Such a path is consistent with a singleton attractor only if the signs of (u, v) and (v, u) are equal. If the signs are + then $u = v$, and if the signs are - then $u = \overline{v}$. In both cases u can be eliminated, substituting v or $\overline{v}$ respectively for u, and v becomes a free variable.

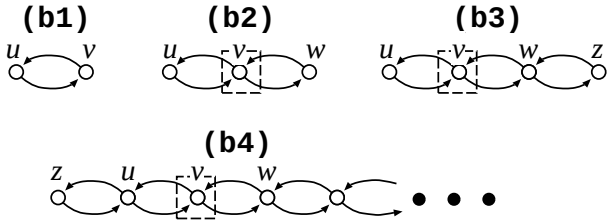

Figure 4: Types of fragments in paths.

Next, we consider elimination of paths of length 2 at least consisting of bidirectional edges. The other types of paths can be handled within the same complexity. We consider paths given in Fig. 4 and the resulting complexity is summarized in Table 2. For example, consider case (b4) with all edges of type (-,-). If $v = 0$ is assigned, we have $u = w = 1$ and $z = 0$ and thus 4 nodes are eliminated. Otherwise a clause of $\overline{u} \vee \overline{w}$ is added and the value of z is uniquely determined from the value of u (i.e., $z = \overline{u}$), which means that three nodes are eliminated and one clause is added. Consequently $f(k)$

[2]We will say that in the network there is a bidirectional edge between u and v if both edges (u, v) and (v, u) are present, regardless of the signs of these two edges.

must satisfy

$$f(k) \leq \max \begin{cases} f(k-2), \\ 2 \cdot f(k-3), \\ 2 \cdot f(k-4), \\ 1.234 \cdot f(k-3) \ + \ f(k-4), \end{cases}$$

from which $f(k) = O(1.266^k)$ follows. Therefore, the time complexity of eliminating paths with a total of k nodes is $O(1.266^k)$.

Table 2: Complexity of elimination of paths.

type of path	#ASS	#MAXCL	complexity
(b1)	0	0	$f(k-2)$
(b2)	2	0	$2 \cdot f(k-3)$
(b3)	2	0	$2 \cdot f(k-4)$
(b4)	2	1	$1.234 \cdot f(k-3)$ $+ f(k-4)$

Next, we consider elimination of cycles, where we need to consider two cases: all edges are bidirectional, and some edges are unidirectional. We begin with the former case. Since the given network is bipartite and any of our assignment strategies does not change the length of cycles, no cycle is of odd length. If there is a cycle of length of 4 or 6, there are 2 or 5 patterns of assignments as shown in Fig. 5 (c1) or (c2) respectively. If there is a cycle of length longer than or equal to 8, we transform the cycle into a path by selecting a set of consecutive nodes as shown in Fig. 5 (c3). The complexity is summarized in Table 3. For the latter case, it is enough to consider the fragment types shown in Fig. 6[3] since there is no node with indegree 0. The complexity of elimination is then as shown in Table 4. By letting $f(k)$ be at most the maximum of (a1)-(a2), (b1)-(b4), (c1)-(c3) and (d1)-(d2), we have $f(k) = O(1.338^k)$.

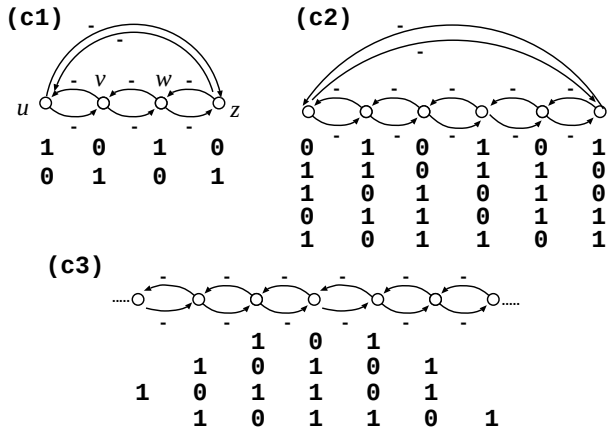

Figure 5: Elimination strategies for cycles consisting of bidirectional edges.

[3]Cases containing positive unidirectional edges can be handled in a similar way.

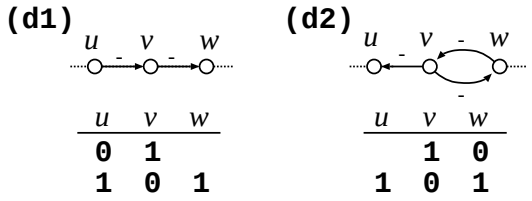

Figure 6: Types of fragments in cycles containing unidirectional edges.

Table 3: Complexity of elimination of cycles consisting of bidirectional edges.

type of fragment	#ASS	#MAXCL	complexity
(c1)	2	0	$2 \cdot f(k-4)$
(c2)	5	0	$5 \cdot f(k-6)$
(c3)	4	0	$f(k-3)+f(k-5)$ $+2 \cdot f(k-6)$

Suppose that STEP 5 is executed after k nodes are assigned in STEP 2. Then STEP 5 handles at most $2n-k$ nodes. We also need to combine at most k clauses generated in STEP 2 with the clauses generated in STEP 5 for solving SAT. Therefore, the time complexity per execution of STEP 5 is

$$O(1.234^k \cdot 1.338^{2n-k}) \ \leq \ O(1.338^{2n}).$$

□

Here we analyze the total time complexity. Since the number of times that each of STEP 4 and STEP 5 is executed is $O(1.381^K)$, the total time complexity is $O(1.381^K) \cdot \left[O(2^{n-(K/2)}) + O(1.338^{2n})\right]$. By letting $2^{n-(K/2)} = 1.338^{2n}$, we have $K = 0.3196n$ and thus the total time complexity is $O(1.985^n)$.[4]

Theorem 10 *The problem of detection of a 2-periodic attractor of an AND/OR BN can be solved in* $O(1.985^n)$ *time.*

5 Finding a Periodic Attractor of an nc-BN with Bounded Treewidth

In this section, we consider detection of a periodic attractor of an nc-BN whose treewidth is bounded by a constant w. It is known that many NP-hard graph problems can be solved in polynomial time using dynamic programming if the treewidth of the graph is bounded by a constant [11]. It is to be noted that Tamaki developed a practically efficient algorithm for enumerating all attractors of a BN by using the *directed path decomposition* [26]. Though the path decomposition has some similarity with the treewidth, he did not analyze the time complexity, and our approach is considerably different from his approach.

To define treewidth we need the notion of tree decomposition [11]. To simplify notations we assume that the nodes have been numbered $1 \ldots n$, $V = \{1, \ldots, n\}$, although in the examples we continue to label the nodes alphabetically for greater clarity. A *tree decomposition* of a graph $G = (V, E)$ is a pair $\langle \mathcal{T}(\mathcal{V}_\mathcal{T}, \mathcal{E}_\mathcal{T}), (B_t)_{t \in \mathcal{V}_\mathcal{T}} \rangle$, where $\mathcal{T}(\mathcal{V}_\mathcal{T}, \mathcal{E}_\mathcal{T})$ is a rooted tree and $(B_t)_{t \in \mathcal{V}_\mathcal{T}}$ is a family of subsets of V such that (see also Fig. 7)

[4]This result remains valid even if there exist $c \cdot \log n$ nodes with non AND/OR functions, where c is any constant.

Table 4: Complexity of elimination of cycles containing unidirectional edges.

type of fragment	#ASS	#MAXCL	complexity
(d1)	2	0	$f(k-2)+f(k-3)$
(d2)	2	0	$f(k-2)+f(k-3)$

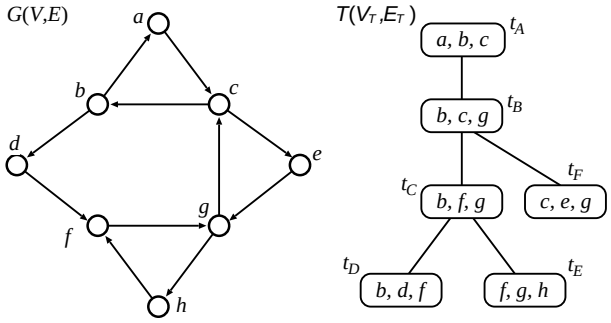

Figure 7: Example of tree decomposition with treewidth 2.

- for every $i \in V$, $B^{-1}(i) = \{t \in \mathcal{V}_{\mathcal{T}} | i \in B_t\}$ is nonempty and connected in $\mathcal{T}$, and
- for every edge $\{i_1, i_2\} \in E$, there exists $t \in \mathcal{V}_{\mathcal{T}}$ such that $i_1, i_2 \in B_t$.

The *width* of the decomposition is defined as $\max_{t \in \mathcal{V}_{\mathcal{T}}}(|B_t| - 1)$ and the *treewidth of* G is the minimum of the widths among all the tree decompositions of G. Graphs with treewidth at most k are also known as *partial* k*-trees* [11]. It is known that any graph of treewidth k has a separator of size $O(k)$, and conversely any n-vertex graph with separator of size k has treewidth $O(k \log n)$ [6].

It is known that planar graphs have $O(\sqrt{n})$ treewidth [6]. Tamura and Akutsu developed an $2^{O((\log n)\sqrt{n})}$ time algorithm for singleton attractor detection of a planar AND/OR BN [28]. We show first how their approach can be extended to the detection of a singleton attractor in an nc-BN with bounded treewidth w, and then show how to modify the algorithm so that it can detect an attractor of periodicity exactly p.

The extension is based on a simple but important observation on nc-functions. We explain it using an example. Let the nc-function associated with node 4 be $f_4 = x_1 \vee (x_2 \wedge \overline{x_3} \wedge (x_4 \vee x_5 \vee (x_6 \wedge (\overline{x_7} \vee x_8))))$. In order to satisfy $f = 1$, we need not consider 2^8 assignments. Instead, it is enough to consider the following partial assignments, where '*' means "don't care".

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
1	*	*	*	*	*	*	*
0	1	0	1	*	*	*	*
0	1	0	0	1	*	*	*
0	1	0	0	0	1	0	*
0	1	0	0	0	1	1	1

Similarly, in order to satisfy $f = 0$, it is enough to consider 4 partial assignments. Observe that a singleton attractor satisfies the equation $f_4(\mathbf{x}) = x_4$, and that among the partial assignments satisfying $f_4(\mathbf{x}) = 1$ only the first two satisfy this equation.

To describe this result in general let us formally define a *partial assignment*.

Definition 11 *A* partial assignment *is any non-empty set $\phi = b_1 \times \cdots \times b_n$ such that $b_i \subseteq \{0,1\}$. A complete assignment is a partial assignment ϕ in which b_i is a singleton for all i. It will be convenient to denote b_r, the r-th component of ϕ, also by $(\phi)_r$.*

Note that the intersection of two partial assignments ϕ, ψ is a partial assignment itself, unless it is empty in which case we say that the partial assignments are *disjoint*; note also that the two assignments are disjoint, $\phi \cap \psi = \emptyset$, if and only if there is a component $r \in \{1, \ldots, n\}$ such that $(\phi)_r \cap (\psi)_r = \emptyset$.

Definition 12 *The partial assignment ϕ is a local fixed point at node i, with associated nc-function f_i, if the set $(\phi)_i$ is a singleton and $f_i(\phi) = (\phi)_i$.*

In these terms, a singleton attractor is a complete assignment that is a local fixed point at each of the nodes.

Proposition 13 *Let f_i be an nc-function with m inputs, associated with node i. Then the set of all complete assignments that are local fixed points at node i can be partitioned into $m+1 \leq n+1$ disjoint partial assignments each of which is a local fixed point at node i.*

(Proof) The proof is by induction on m. For the base case, $m = 1$, recall that we assume that each function has at least one input. Hence in this case f_i is of the form x_k or $\overline{x_k}$. If, for example, $f_i = x_k$ then the complete assignments that are a local fixed point at node i can be partitioned into those that have the value 0 at x_i and the value 0 at x_k, and those that have the value 1 at x_i and the value 1 at x_k, for a total of 2 partial assignments.

For the induction step we assume without loss of generality that the variable that appears first in f_i is x_1, i.e. f_i has one of the forms $x_1 \vee g$, $x_1 \wedge g$, $\overline{x_1} \vee g$, or $\overline{x_1} \wedge g$, where g is a nc-function with inputs $x_2, \ldots, x_m$. We only consider the case $f_i = x_1 \vee g$. Then in a complete assignment that is a local fixed point at node i either $x_1 = 1$ (and the assignment has the value 1 at i) or $x_1 = 0$ and the assignment to the variables other than x_1 forms a local fixed point at node i of the function g.

Since the number of partial assignments of $x_2, \ldots, x_m$ needed to cover all assignments to g is m, by the induction hypothesis, it follows that the the number of partial assignments of $x_1, \ldots, x_m$ needed to cover all assignments to f is $m + 1$. □

Notation: We will denote the partial assignments appearing in the statement of this proposition by $\phi_i^1, \ldots, \phi_i^{m_{i+1}}$.

5.1 Finding a singleton attractor

The first step of the algorithm for finding a singleton attractor is to construct

$$\mathcal{A}_t^0 = \{ \bigcap_{i \in B_t} \phi_i^{j_i} \mid \bigcap_{i \in B_t} \phi_i^{j_i} \neq \emptyset \}.$$

It follows immediately from Proposition 13 that this set can be characterized as follows.

Lemma 14 *$\mathcal{A}_t^0$ is a partition of the set of assignments that are local fixed points at all nodes in B_t: if $\alpha \neq \alpha' \in \mathcal{A}_t^0$ then both are local fixed points at all nodes in B_t and $\alpha \cap \alpha' = \emptyset$; and if β is a local fixed point at all nodes in B_t then there is an $\alpha \in \mathcal{A}_t^0$ such that $\beta \subseteq \alpha$.*

Clearly the size of $\mathcal{A}_t^0$ is at most $(m+1)^{|B_t|}$. We will denote its elements by $\alpha_t^j,\ j = 1, \ldots$.

To ease the description of the next step we introduce two definitions.

Definition 15 *Two partial assignments ϕ and ψ are said to be* compatible *if $\phi \cap \psi \neq \emptyset$. A partial assignment ρ is a* refinement *of $\alpha \in \mathcal{A}_t^0$ if $\rho \subseteq \alpha$ and ρ is a local fixed point at all nodes in B_s for all s in the subtree rooted at t.*

The next step computes for each node a set of disjoint partial assignments, $\mathcal{A}_t$, each of which is a local fixed point at all nodes in B_t and has at least one refinement with respect to $\mathcal{T}$, and which together cover all such local fixed points. As we will see this can be achieved, using dynamic programming, by checking for each partial assignment in $\mathcal{A}_t^0$ that it is compatible with at least one partial assignment in $\mathcal{A}_{t_j}$, for all children t_j of t, and removing it if it does not pass this test. For a leaf $\mathcal{A}_t = \mathcal{A}_t^0$, of course.

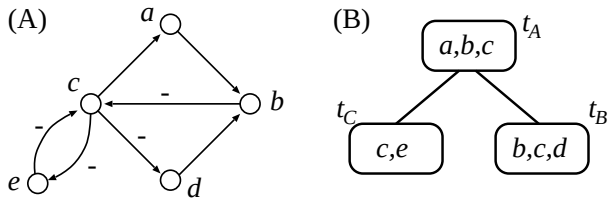

Figure 8: (A) BN and (B) tree decomposition used in Example 16.

Example 16 *Consider a BN defined by*

$$f_a = c,\ f_b = a \vee d,\ f_c = \overline{b} \vee \overline{e},\ f_d = \overline{c},\ f_e = \overline{c}.$$

One possible tree decomposition is shown in Figure 8. Possible partitions of partial assignments that are local fixed points, for each node, are:

$$\begin{aligned}
\phi_a^1 &= \langle\{1\},\{0,1\},\{1\},\{0,1\},\{0,1\}\rangle,\\
\phi_a^2 &= \langle\{0\},\{0,1\},\{0\},\{0,1\},\{0,1\}\rangle,\\
\phi_b^1 &= \langle\{0\},\{0\},\{0,1\},\{0\},\{0,1\}\rangle,\\
\phi_b^2 &= \langle\{0\},\{1\},\{0,1\},\{1\},\{0,1\}\rangle,\\
\phi_b^3 &= \langle\{1\},\{1\},\{0,1\},\{0,1\},\{0,1\}\rangle,\\
\phi_c^1 &= \langle\{0,1\},\{0\},\{1\},\{0,1\},\{0,1\}\rangle,\\
\phi_c^2 &= \langle\{0,1\},\{1\},\{1\},\{0,1\},\{0\}\rangle,\\
\phi_c^3 &= \langle\{0,1\},\{1\},\{0\},\{0,1\},\{1\}\rangle\\
\phi_d^1 &= \langle\{0,1\},\{0,1\},\{1\},\{0\},\{0,1\}\rangle,\\
\phi_d^2 &= \langle\{0,1\},\{0,1\},\{0\},\{1\},\{0,1\}\rangle,\\
\phi_e^1 &= \langle\{0,1\},\{0,1\},\{1\},\{0,1\},\{0\}\rangle,\\
\phi_e^2 &= \langle\{0,1\},\{0,1\},\{0\},\{0,1\},\{1\}\rangle.
\end{aligned}$$

These result in the following $\mathcal{A}_{t_A}^0 = \{\alpha_{t_A}^i, i = 1,2\}$, $\mathcal{A}_{t_B}^0 = \{\alpha_{t_B}^i, i = 1,2,3,4\}$ $\mathcal{A}_{t_C}^0 = \{\alpha_{t_C}^i, i = 1,2\}$, *with*

$$\begin{aligned}
\alpha_{t_A}^1 &= \langle\{1\},\{1\},\{1\},\{0,1\},\{0\}\rangle,\\
\alpha_{t_A}^2 &= \langle\{0\},\{1\},\{0\},\{1\},\{1\}\rangle,\\
\alpha_{t_B}^1 &= \langle\{1\},\{1\},\{1\},\{0\},\{0\}\rangle,\\
\alpha_{t_B}^2 &= \langle\{0\},\{1\},\{0\},\{1\},\{1\}\rangle,\\
\alpha_{t_B}^3 &= \langle\{1\},\{1\},\{0\},\{1\},\{1\}\rangle,\\
\alpha_{t_B}^4 &= \langle\{0\},\{0\},\{1\},\{0\},\{0,1\}\rangle,\\
\alpha_{t_C}^1 &= \langle\{0,1\},\{0,1\},\{1\},\{0,1\},\{0\}\rangle,\\
\alpha_{t_C}^2 &= \langle\{0,1\},\{1\},\{0\},\{0,1\},\{1\}\rangle.
\end{aligned}$$

$\alpha^1_{t_A}$ is compatible only with $\alpha^1_{t_B}$ and $\alpha^1_{t_C}$, and $\alpha^2_{t_A}$ is compatible only with $\alpha^2_{t_B}$ and $\alpha^2_{t_C}$. $\alpha^1_{t_B}$ happens to also be a refinement of $\alpha^1_{t_A}$.

The following high-level code summarizes the computation of $\mathcal{A}_t$ and the detection of a singleton attractor.

Algorithm KtreeAtt

1. Compute a tree decomposition $\mathcal{T}(\mathcal{V}_\mathcal{T}, \mathcal{E}_\mathcal{T})$ of $G(V, E)$.
2. For each node $t \in \mathcal{V}_\mathcal{T}$, compute a set of assignments $\mathcal{A}^0_t$ by
$$\mathcal{A}^0_t = \{ \bigcap_{i \in B_t} \phi_i^{j_i} \mid \bigcap_{i \in B_t} \phi_i^{j_i} \neq \emptyset \}.$$
3. Execute STEP 4 in a bottom-up manner by using dynamic programming; begin by setting $\mathcal{A}_t = \mathcal{A}^0_t$ for each leaf node t.
4. For each non-leaf node $t \in \mathcal{V}_\mathcal{T}$, compute $\mathcal{A}_t$ by
$$\mathcal{A}_t \;=\; \{\alpha^j_t \in \mathcal{A}^0_t \mid (\forall t_i \in \{t_1, \ldots, t_d\} \exists \alpha^k_{t_i} \in \mathcal{A}_{t_i})(\alpha^j_t \cap \alpha^k_{t_i} \neq \emptyset)\},$$
where $t_1, t_2, \ldots, t_d$ are the children of t.
5. If $\mathcal{A}_{root} \neq \emptyset$ for the *root* of $\mathcal{T}$, then output a singleton attractor by using a traceback procedure starting from any $\alpha_{root} \in \mathcal{A}_{root}$; else return $false$.

When it is found in step 4 that $\alpha^j_t \in \mathcal{A}_t$ because $\alpha^j_t \cap \alpha^{j_i}_{t_i} \neq \emptyset, i = 1, \ldots, d$, then a refinement, ρ^j_t, associated with α^j_t can be computed as $\rho^j_t = \alpha^j_t \cap (\cap_{i=1}^d \rho^{j_i}_{t_i})$, where $\rho^{j_i}_{t_i}$ is the refinement associated with $\alpha^{j_i}_{t_i}$. Thus the traceback of step 5 in fact computes a refinement ρ_{root} associated with α_{root}.

Theorem 17 *Algorithm* **KtreeAtt** *outputs a singleton attractor of an nc-BN with bounded treewidth w in $O(n^{2(w+1)}poly(n))$ time if and only if the network possesses one.*

(Proof) First we prove the correctness. For the 'if' part, it suffices to observe that any singleton attractor is a refinement of some partial assignment in $\mathcal{A}^0_t$, for all t, and since it is clearly compatible with an assignment in every child of t it will also appear in all $\mathcal{A}_t$.

The correctness of the 'only if' part follows from Proposition 13 and the facts that $\alpha^k_{t_i}$ and $\alpha^l_{t_h}$ are compatible if α^j_t is compatible with both $\alpha^k_{t_i}$ and $\alpha^l_{t_h}$, and there is no edge in the network between a node in $B_{t_i} \cap (V - B_t)$ and a node in $B_{t'} \cap (V - B_t)$ where t' is any ancestor of t in $\mathcal{T}$ or any node in a subtree of $\mathcal{T}$ rooted at $t_{i'}$, $i' \neq i$. Therefore, we analyze the time complexity below.

Since there exist at most $n+1$ partial assignments per node and each B_t consists of at most $w+1$ nodes, we need to check $O((n+1)^{w+1}) = O(n^{w+1})$ combinations of partial assignments per $t \in \mathcal{V}_\mathcal{T}$. Since the intersection of two partial assignments can be computed in $O(n)$ time, the consistency of each combination, which also constitutes a partial assignment (i.e., $\bigcap_{i \in B_t} \phi_i^{j_i}$), can be checked in $O(wn)$ time, which is a polynomial of n. Therefore, for each leaf $t \in \mathcal{V}_\mathcal{T}$, it takes $O(n^{w+1}poly(n))$ time to construct $\mathcal{A}^0_t$.

For each non-leaf node $t \in \mathcal{V}_\mathcal{T}$, we examine the compatibility for $O(n^{w+1} \times n^{w+1} \times h)$ pairs of partial assignments, where h is the number of children of t. Since the compatibility between two partial assignments can be checked in $O(wn)$ time and the total number of children is $O(n)$, $O(n^{2(w+1)}poly(n))$ time is required to construct $\mathcal{A}_t$s for all non-leaf nodes.

For the traceback procedure, it is enough to keep one $\alpha^j_{t_i}$ compatible with α_t for each $i = 1, \ldots, d$. Thus the traceback can be done in $O(poly(n))$ time for constant w by a depth-first search starting from the *root*.

Since it is known that tree decomposition of a graph with n nodes can be computed in linear time for fixed w [11], the total time complexity of singleton attractor detection is $O(n^{2(w+1)}poly(n))$. □

Before showing how this approach can be extended to finding a p-periodic attractor we outline how the problem of finding a singleton attractor of a network with bounded treewidth can be solved by reduction to a certain Constraint Satisfaction problem with bounded treewidth, which is known to be solvable in time that is polynomial in the size of the problem. We find this approach of interest, even though we have been unable to find a similar reduction for the problem of constructing a p-periodic attractor.

5.2 Reduction to a CSP

Recall the characterization of a singleton attractor as an assignment α that is a local fixed point at each of the nodes of the network,

$$f_i(\alpha) = (\alpha)_i, \text{for all } i. \tag{1}$$

Recall also that the set of all complete assignments that are local fixed points at node i can be partitioned into at most $m+1$ disjoint partial assignments $\phi_i^1, \phi_i^2, \ldots$ each of which is a local fixed point at node i. Note that in each of these partial assignments the i-th component is a singleton.

The basis of the reduction is the following alternative characterization of a singleton attractor.

Theorem 18 *An nc-network with underlying graph $G = (V, E)$ has a singleton attractor if and only if for each node i there is an index j_i such that the partial assignments $\phi_i^{j_i}, i = 1, \ldots, m_i + 1$, satisfy*

$$\phi_i^{j_i} \cap \phi_\ell^{j_\ell} \neq \emptyset, \text{ for all } \{i, \ell\} \in E. \tag{2}$$

(Proof) If the network has a singleton attractor ϕ then it is a local fixed point at each of the nodes, and in particular there exists for each i a j_i such that $\phi \subseteq \phi_i^{j_i}$. Clearly equation (2) is satisfied by these partial assignments.

The first step in proving the converse is to establish that equation (2) holds if and only if

$$\phi_i^{j_i} \cap \phi_\ell^{j_\ell} \neq \emptyset, \text{for all } 1 \leq i \leq \ell \leq n. \tag{3}$$

Suppose to the contrary that $\phi_i^{j_i} \cap \phi_\ell^{j_\ell} = \emptyset$ for some $\{i, \ell\} \notin E$. This means that the two partial assignments are disjoint in at least one component; for ease of notation assume this is the first component. Thus $|(\phi_i^{j_i})_1| = |(\phi_\ell^{j_\ell})_1| = 1$, say $(\phi_i^{j_i})_1 = \{0\}, (\phi_\ell^{j_\ell})_1 = \{1\}$. It follows that x_1 appears in f_i as well as in f_ℓ, i.e. $\{1, i\} \in E$ and $\{1, \ell\} \in E$. Therefore, by assumption, $\phi_1^{j_1} \cap \phi_i^{j_i} \neq \emptyset$ and $\phi_1^{j_1} \cap \phi_\ell^{j_\ell} \neq \emptyset$, and in particular $(\phi_1^{j_1})_1 \cap \{1\} \neq \emptyset$ and $(\phi_1^{j_1})_1 \cap \{0\} \neq \emptyset$, which is possible only if $(\phi_1^{j_1})_1 = \{0, 1\}$ contradicting the requirement that $(\phi_1^{j_1})_1$ be a singleton.

To complete the proof of the theorem we now prove that if there are indices j_i such that equation (3) holds then the partial assignment $\cap_{i=1}^n \phi_i^{j_i}$ is not empty, and hence is a singleton attractor. Suppose to the contrary that $\cap_{i=1}^n \phi_i^{j_i} = \emptyset$, and let k be the largest index such that $\psi = \cap_{i=1}^{k-1} \phi_i^{j_i} \neq \emptyset$. Since $\psi \cap \phi_k^{j_k} = \emptyset$. there exists a component r such that $(\psi)_r \cap (\phi_k^{j_k})_r = \emptyset$. This means that $(\psi)_r$ and $(\phi_k^{j_k})_r$ are singletons, and therefore $(\phi_\ell^{j_\ell})_r = (\psi)_r$ for some ℓ. Thus $(\phi_\ell^{j_\ell})_r \cap (\phi_k^{j_k})_r = \emptyset$, contradicting the assumption that equation (3) holds. □

A *constraint satisfaction problem* (CSP) consists of triples $(Z, \mathcal{D}, \mathcal{C})$, where $Z = \{z_1, \ldots, z_n\}$ is a set of variables, a corresponding set of domains $\mathcal{D} = \{D_1, \ldots, D_n\}$ in which D_i constitutes the set of possible values of variable Z_i, and $\mathcal{C}$ is a set of constraints $\{C_1, \ldots, C_m\}$ each of which is a restriction on the possible values in the domain of some subset of the variables. The *constraint graph* $\Gamma(Z, \mathcal{C})$ of a CSP $(Z, \mathcal{D}, \mathcal{C})$ consists of the node set $\{z_1, \ldots, z_n\}$ and the edge set $\{\{z_i, z_j\} \mid z_i, z_j$ occur in the same constraint $C \in \mathcal{C}\}\}$.

Theorem 19 (Freuder 1990 [12]) *Detection of a satisfying assignment of a CSP $(Z, \mathcal{D}, \mathcal{C})$ can be found in time $O((\#\mathcal{D})^{w+1} poly(|Z|+|\mathcal{D}|+|\mathcal{C}|))$, where w is the treewidth of the constraint graph $\Gamma(X, \mathcal{C})$ (given in an appropriate tree decomposition).*

Theorem 20 *Detection of a singleton attractor of an nc-BN with bounded treewidth w can be converted into a constraint satisfaction problem that is solvable in $O(n^{2(w+1)} poly(n))$ time.*

(Proof) We use Theorem 18 to convert the problem of detecting a singleton attractor of a nc-BN N with underlying graph $G = (V, E)$ into a CSP.

Define $\mathrm{CSP}(N) = (Z, \mathcal{D}, \mathcal{C})$ to be the constraint satisfaction problem in which the domain of z_i is the Cartesian product $D_i = \{i\} \times \{\phi_i^1, \ldots, \phi_i^{m_i}\}$, and the set of constraints $\mathcal{C}$ is indexed by $(u, v), u \leq v$ and $\{u, v\} \in E$. An assignment $z_i = (i, \phi_i^{j_i}), i = 1, \ldots, n$, satisfies $C_{(u,v)}$ if $\phi_u^{j_u} \cap \phi_v^{j_v} \neq \emptyset$.

The CSP so constructed contains n variables and $m = |E|$ constraints. By Proposition 13, $\mathcal{D}$ contains at most $\sum_{i \in V}(m_i + 1) = O(m)$ elements each of which has length n, i.e., $\#\mathcal{D} = O(n^2)$ and $|\mathcal{D}| = O(n^3)$, and the size of the set of constraint relations can be bounded by

$$c \cdot \textstyle\sum_{\{i,j\} \in E}(4 + m_i + m_j) \cdot n^2 \leq O(n^2 m^2),$$

for some constant $c > 0$. All in all, this easily implies that the size of $\mathrm{CSP}(N)$ is polynomial in n. Hence, $\mathrm{CSP}(N)$ is computable in time polynomial in the number of nodes.

Theorem 18 ensures that $\mathrm{CSP}(N)$ has a satisfying assignment if and only if N has a singleton attractor. Moreover, the constraint graph of $\mathrm{CSP}(N)$ is isomorphic to the graph G underlying N, and so has treewidth w. Applying Freuder's Theorem 19 yields an $O(n^{2(w+1)}\ poly(n))$ algorithm. □

5.3 Finding a p-periodic attractor

The basic idea for the extension of our approach to the construction of p-periodic attractors, is to compute a singleton attractor of $N^p(V^p, F^p)$ using **KtreeAtt**. For the tree decomposition of $N^p(V^p, F^p)$, we use the natural extension of the decomposition for $N(V, F)$.

Proposition 21 *If the graph G associated with $N(V, F)$ has treewidth w, then the graph G^p associated with $N^p(V^p, F^p)$ has a decomposition whose treewidth is less than $p(w + 1)$, and is such that for each $v \in V$ all $v(1), \cdots, v(p)$ are included in the same bag.*

(Proof) Consider a tree decomposition of G, $\langle \mathcal{T}(\mathcal{V}_\mathcal{T}, \mathcal{E}_\mathcal{T}), (B_t)_{t \in \mathcal{V}_\mathcal{T}} \rangle$. For each $B_t = \{i_1, \ldots, i_k\}$ where $k \leq w + 1$, we define B_t^p for G^p by

$$B_t^p = \{i_1(q), \ldots, i_k(q) \mid q = 1, \ldots, p\}.$$

It is straight-forward to see that the B_t^ps give a tree decomposition of G^p. Since the maximum size of B_t^p is at most $p(w + 1)$, the proposition holds. □

It seems that the above bound is nearly tight up to a constant factor in the worst case. Suppose that $N(V, F)$ is a linear chain with n nodes, where each edge is bidirectional. Then, the treewidth of $N(V, F)$ is 1. On the other hand, the size of the separator of $N^n(V^n, F^n)$ is $\Theta(n)$ where $p = n$ since the structure of $N^n(V^n, F^n)$ is similar to that of a grid of size $n \times n$. Therefore, the treewidth of $N^n(V^n, F^n)$ is $\Omega(n)$. We can extend this discussion for larger w by replacing a linear chain with an $n \times n$ grid-like BN $N(V, F)$. Then, $N(V, F)$ has treewidth $w = \Theta(n)$ and $N^n(V^n, F^n)$ has treewidth $\Theta(n^2) = \Theta(wp)$ where $p = n$.

It is straightforward to convert a singleton attractor of $N^p(V^p, F^p)$ into an attractor of $N(V, F)$. However, if no further steps are taken, all we can say about its period is that it divides p. The main

task of this section is, therefore, to show how to modify the algorithm so as to ensure that an attractor is constructed whose period is exactly p.

It is not difficult to verify that Algorithm **KtreeAtt** can be modified so as to generate all possible singleton attractors, and to check the precise periodicity of each. This is not a feasible option, at least in principle, because the generation itself may take too long. Indeed, it yields much more information than is needed if all we want is one attractor of period exactly p. To ensure the latter it suffices to know that for each i, $2 \leq i \leq p$, there is some node v in the network such that its values in the attractor satisfy $v(i) \neq v(1)$. However, in order to be able to deduce this for an assignment in the root node, we have to allow for all patterns of periodicity at other nodes of the tree decomposition. These are the ideas behind the construction that we describe now.

Definition 22 *The partial assignment $\alpha \in \mathcal{A}_t$ is* consistent *with the 0-1 vector $T[2 \ldots p]$ if there is a refinement ρ of α such that for all $2 \leq i \leq p$*

$$T[i] = 1 \iff \text{there is a } v \in \cup_{t'} B_{t'} \text{ such that } (\rho)_{v(1)} \neq (\rho)_{v(i)},$$

where t' ranges over t and its descendants.

In this terminology we are looking for an attractor that is consistent with the vector T whose values are $T[i] = 1, i = 2, \ldots, p$. Denote this vector T_1.

A given partial assignment $\alpha \in \mathcal{A}_t$ may, of course, be consistent with several of the 2^{p-1} possible vectors T. We index these vectors as T_ℓ. Our modification of the algorithm adjoins to each $\alpha \in \mathcal{A}_t$ a 0-1 vector of length 2^{p-1}, a *consistency vector* $\mathbf{C}_\alpha$, the ℓ-th entry of which indicates whether α is consistent with T_ℓ.

Definition 23 *The semantics of the vector $\mathbf{C}_\alpha$ is*

$$\mathbf{C}_\alpha[\ell] = 1 \Longleftrightarrow T_\ell \text{ is consistent with } \alpha \ .$$

Thus the task of the algorithm is to search for an $\alpha \in \mathcal{A}_{root}$ such that $\mathbf{C}_\alpha[1] = 1$, meaning that there is at least one refinement of α that does not have a period smaller than p.

Definition 24 *Given a partial assignment α denote by $\mathbf{C}^0_\alpha$ the consistency vector defined as follows: $\mathbf{C}^0_\alpha[i] = 0$ except that $\mathbf{C}^0_\alpha[\ell] = 1$ where ℓ is the index of the one 0-1 vector with which α is consistent, i.e. the T_ℓ with the property that*

$$T_\ell[i] = 1 \iff |(\alpha)_{v(1)}| = |(\alpha)_{v(i)}| = 1 \text{ and } (\alpha)_{v(1)} \neq (\alpha)_{v(i)} \text{ for some } v \in B_t$$

Note that if $\alpha \in \mathcal{A}^0_t$ is a local fixed point at $v(1)$ then it is also a local fixed point at $v(i)$ because, by Proposition 21, for each $v \in V$ all $v(1), \cdots, v(p)$ are included in the same bag.

For 0-1 vectors T, T' and T'', we write $T = T' \vee T''$ if

$$T[i] = 1 \iff T'[i] = 1 \text{ or } T''[i] = 1.$$

Finally, to compute $\mathbf{C}_\alpha$, $\alpha \in \mathcal{A}_t$ using dynamic programming from the leaves up, it will be convenient to introduce the binary operator $\otimes$, as follows.

Definition 25 *Given consistency vectors $\mathbf{C}_\alpha$ and $\mathbf{C}_\beta$ the consistency vector $\mathbf{C}_\alpha \otimes \mathbf{C}_\beta$ has values*

$$(\mathbf{C}_\alpha \otimes \mathbf{C}_\beta)[\ell] = 1 \iff (\exists j)(\exists k)(T_\ell = T_j \vee T_k \text{ and } \mathbf{C}_\alpha[j] = \mathbf{C}_\beta[k] = 1).$$

It is not difficult to verify that this operator is commutative and associative; thus it will not be necessary to indicate the order of computation for expressions such as $\mathbf{C}_\alpha \otimes \mathbf{C}_\beta \otimes \mathbf{C}_\gamma$.

Computing $\mathbf{C}_\alpha$

1. For each node $t \in \mathcal{V}_{\mathcal{T}}$ and each $\alpha \in \mathcal{A}_t$, compute $\mathbf{C}^0_\alpha$.

2. Execute STEP 3 in a bottom-up manner by using dynamic programming; begin by setting, for each leaf node t, $\mathbf{C}_\alpha = \mathbf{C}^0_\alpha$ for all $\alpha \in \mathcal{A}_t$.

3. For each non-leaf node t, with children $t_1, \ldots, t_d$, and for each $\alpha \in \mathcal{A}_t$, do

   (a) for each t_i set $\mathbf{D}_{\alpha,t_i} = \bigvee_{\beta \in \mathcal{A}_{t_i}(\alpha)} \mathbf{C}_\beta$, where $\mathcal{A}_{t_i}(\alpha) = \{\beta \in \mathcal{A}_{t_i} : \beta \cap \alpha \neq \emptyset\}$.

   (b) $\mathbf{C}_\alpha = \mathbf{C}^0_\alpha \otimes \mathbf{D}_{\alpha,t_1} \otimes \cdots \otimes \mathbf{D}_{\alpha,t_d}$.

In explanation of step 2 recall that for a leaf node t each $\alpha \in \mathcal{A}_t$ is in fact its own refinement.

As explained previously, there exists a p-periodic attractor if and only if there is an $\alpha \in \mathcal{A}_{root}$ such that $\mathbf{C}_\alpha[1] = 1$. Furthermore, if it exists a p-periodic attractor can be constructed by using a standard traceback technique. We denote the resulting algorithm by **KtreePAtt**.

Theorem 26 *Detection of a p-periodic attractor of an nc-BN with bounded treewidth w can be done in $O(n^{2p(w+1)}poly(n))$ time for any constants p and w.*

(Proof) Since it is straight-forward to see the correctness of **KtreePAtt**, we only analyze the time complexity.

From Proposition 21, the treewidth of $N^p(V^p, E^p)$ is less than $p(w+1)$, so that the size of $\mathcal{A}_t$ is $O(n^{p(w+1)})$. As in the proof of Theorem 17, for each non-leaf node $t \in \mathcal{V}_{\mathcal{T}}$, we examine the compatibility for $O(n^{p(w+1)} \times n^{p(w+1)} \times d)$ pairs of partial assignments. Therefore, the total computation time for constructing $\mathcal{A}_r$ is $O(n^{2p(w+1)}poly(n))$.

As long as p is a constant the size of the additional information $\mathbf{C}_\alpha$ that the algorithm adjoins is constant, and the time taken by **Computing $\mathbf{C}_\alpha$** is polynomial. □

It is to be noted that if the maximum indegree is bounded by a constant d, the number of possible partial assignment per B_t is bounded by $(d+2)^{p(w+1)}$. Therefore, **KtreePAtt** gives a fixed parameter algorithm when p, w and d are parameters. Furthermore, in such a case, even for a general Boolean function with at most d inputs, it is enough to consider 2^d partial assignments (instead of $m+1$ partial assignments). Therefore, **KtreePAtt** also gives a fixed parameter algorithm for a BN with general Boolean functions when p, w and d are parameters.

6 Conclusion

We have presented a polynomial time algorithm for the detection of a 2-periodic attractor of a positive OR BN, an $O(1.985^n)$ time algorithm for the detection of a 2-periodic attractor of an AND/OR BN, and an $O(n^{2p(w+1)}poly(n))$ time algorithm for the detection of a p-periodic attractor of an nc-BN having bounded treewidth w. The first result suggests that detection of 2-periodic attractor is easy for networks whose structure is severely restricted. The second result, though not practically useful, is the first to establish that it is possible to develop an $O((2-\delta)^n)$ time algorithm for a relatively general class of BNs. The third result suggests that for certain kinds of biological networks it might be possible to efficiently detect an attractor with a short period because it is reported that metabolic networks (which can be modeled as an AND/OR BN) have small treewidth [7].

Although these algorithms provably work in $o(2^n)$ time, they may not work faster than heuristic algorithms in practice. The main purpose of this paper is to show the very existence of $o(2^n)$ time algorithms, for well-defined classes of BNs, and thereby to stimulate further improvements and developments.

Particularly interesting open problems are the development of $O((2-\delta)^n)$ time algorithms for the detection of 2-periodic attractors in an nc-BN, and for the detection of 3-periodic attractors in

an AND/OR BN. It is also left as an open problem to develop a fixed parameter algorithm for the detection of p-periodic attractors in an nc-BN with bounded treewidth.

Acknowledgments

T.A. was partially supported by MEXT, Japan (Grant-in-Aid 22650045). A.A.M. was supported by a JSPS Invitation Fellowship Program for Research in Japan (Short Term). T.T. was partially supported by JSPS, Japan (Grant-in-Aid for Young Scientists (B) 23700017).

References

[1] T. Akutsu, S. Kuhara, O. Maruyama, and S. Miyano, "A system for identifying genetic networks from gene expression patterns produced by gene disruptions and overexpressions," *Genome Informatics*, vol. 9, pp. 151–160, 1998.

[2] T. Akutsu, A. A. Melkman, T. Tamura, and M. Yamamoto, "Determining a singleton attractor of a Boolean network with nested canalyzing functions," *J. Comput. Biol.*, vol. 18, pp. 1275–1290, 2011.

[3] T. Akutsu, A. A. Melkman, and T. Tamura, "Singleton and 2-periodic attractors of sign-definite Boolean networks," *Inf. Proc. Lett.*, vol. 112, pp. 35–38, 2012.

[4] J. Aracena, J. Demongeot, and E. Goles, "Positive and negative circuits in discrete neural networks," *IEEE Trans. Neural Networks*, vol. 15, pp. 77–83, 2004.

[5] C. Barrett, C., H. B. Hunt III, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, R. E. Stearns, and M. Thakur, "Predecessor existence problems for finite discrete dynamical systems," *Theoret. Comput. Sci.*, vol. 386, pp. 3–37, 2007.

[6] J. Böttcher, K. P. Pruessmann, A. Taraz, and A. Würfl, "Bandwidth, expansion, treewidth, separators and universality for bounded-degree graphs," *Euro. J. Combinatorics*, vo. 31, pp. 1217–1227, 2010.

[7] Q. Cheng, P. Berman, R. Harrison, and A. Zelikovsky, "Efficient alignments of metabolic networks with bounded treewidth," *Proc. 2010 IEEE International Conference on Data Mining Workshops*, pp. 687–694, 2010.

[8] V. Devloo, P. Hansen, and M. Labbé, "Identification of all steady states in large networks by logical analysis," *Bull. Math. Biol.*, vol. 65, pp. 1025–1051, 2003.

[9] E. Dubrova and M. Teslenko, "A SAT-based algorithm for finding attractors in synchronous Boolean networks," *IEEE/ACM Trans. Compute. Biol. Bioinform.*, vol. 8, pp. 1393–1398, 2011.

[10] B. Drossel, T. Mihaljev, and F. Greil, "Number and length of attractors in a critical Kauffman model with connectivity one," *Phys. Rev. Lett.*, vol. 94, 088701, 2005.

[11] J. Flum and M. Grohe, *Parameterized Complexity Theory.* Berlin: Springer, 2006.

[12] E. C. Freuder. Complexity of k-tree structured constraint satisfaction problems. In *Proceedings of the 8th National Conference on Artificial Intelligence (AAAI'90)*, pages 4–9. AAAI Press/The MIT Press, Menlo Park, CA, 1990.

[13] A. Garg, A. DiCara, I. Xenarios, L. Mendoza, and G. DeMichel, "Synchronous versus asynchronous modeling of gene regulatory networks," *Bioinformatics*, vol. 24, pp. 1917–1925, 2008.

[14] E. Goles and L. Salinas, "Sequential operator for filtering cycles in Boolean networks," *Adv. Appl. Math.*, vol. 45, pp. 346–358, 2010.

[15] S. E. Harris, B. K. Sawhill, A. Wuensche, and S. A. Kauffman, "A model of transcriptional regulatory networks based on biases in the observed regulation rules," *Complexity*, vol. 7, pp. 23–40, 2002.

[16] D. J. Irons, "Improving the efficiency of attractor cycle identification in Boolean networks," *Physica D*, vol. 217, pp. 7–21, 2006.

[17] A. S. Jarrah, R. Laubenbacher, and A. Veliz-Cuba, "The dynamics of conjunctive and disjunctive Boolean network models," *Bull. Math. Biol.*, vol. 72, pp. 1425–1447, 2010.

[18] A. S. Jarrah, B. Raposa, and R. Laubenbacher, "Nested canalyzing, unate cascade, and polynomial functions," *Physica D*, vol. 233, pp. 167–174, 2007.

[19] W. Just, "The steady state system problem is NP-hard even for monotone quadratic Boolean dynamical systems," preprint available at http://www.ohio.edu/people/just/publ.html, 2006.

[20] S. A. Kauffman, *The Origins of Order: Self-organization and Selection in Evolution.* New York: Oxford Univ. Press, 1993.

[21] S. Kosub, "Dichotomy results for fixed-points existence problems for Boolean dynamical systems," *Math. Comput. Sci.*, vol. 1, pp. 487–505, 2008.

[22] L. Layne, E. Dimitrova, and M. Macauley, "Nested canalyzing depth and network stability," *Bull. Math. Biol.*, vol. 74, pp. 422-433, 2012.

[23] M. Leone, A. Pagnani, G. Parisi, and O. Zagordi, "Finite size corrections to random Boolean networks," *J. Stat. Mech.: Theory and Experiment*, vol. 2006, P12012, 2006.

[24] A. A. Melkman, T. Tamura, and T. Akutsu, "Determining a singleton attractor of an AND/OR Boolean network in $O(1.587^n)$ time," *Inf. Proc. Lett.*, vol. 110, pp. 565–569, 2010.

[25] B. Samuelsson and C. Troein, "Superpolynomial growth in the number of attractors in Kauffman networks," *Phys. Rev. Lett.*, vol. 90, 098701, 2003.

[26] H. Tamaki, "A directed path-decomposition approach to exactly identifying attractors of Boolean networks," *Proc. 10th International Symposium on Communications and Information Technologies*, pp. 844–849, 2010.

[27] T. Tamura and T. Akutsu, "Detecting a singleton attractor in a Boolean network utilizing SAT algorithms," *IEICE Trans. Fundamentals*, vol. E92-A, pp. 493–501, 2009.

[28] T. Tamura and T. Akutsu, "Algorithms for singleton attractor detection in planar and non-planar AND/OR Boolean networks," *Math. Comput. Sci.*, vol. 2, pp. 401–420, 2009.

[29] M. Yamamoto, "An improved $O^*(1.234^m)$-time deterministic algorithm for SAT," *Proc. 16th International Symposium on Algorithms and Computation*, pp. 644–653, 2005.

[30] S-Q. Zhang, M. Hayashida, T. Akutsu, W-K. Ching, and M. K. Ng, "Algorithms for finding small attractors in Boolean networks," *EURASIP J. Bioinfor. Syst. Biol.*, vol. 2007, 20180, 2007.