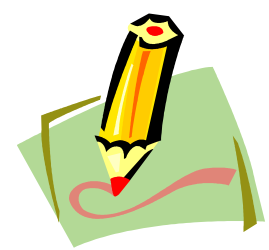
An Improved Algorithm for Detecting a Singleton Attractor in a Boolean Network Consisting of AND/OR Nodes

Algebraic Biology 2008

Takeyuki Tamura and Tatsuya Akutsu
Kyoto University, Japan

Content of Talk

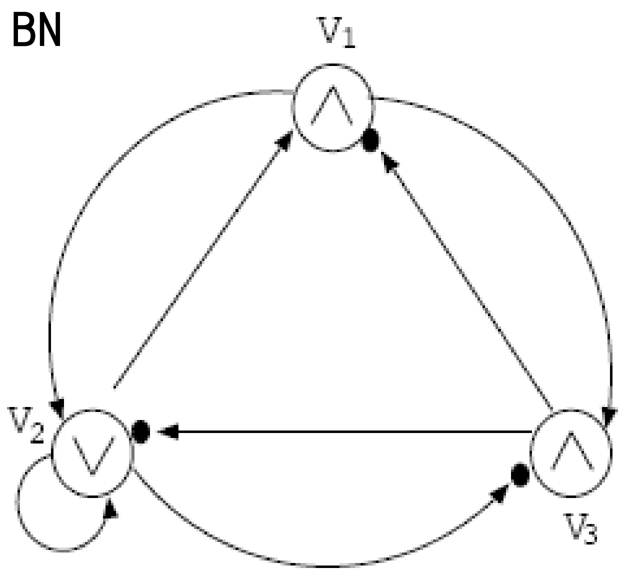
1. Detecting a singleton attractor of a Boolean network
   - What is an (AND/OR) Boolean network?
   - What is a singleton attractor?
     (also called a fixed point)
   - The main problem
     - Is there a singleton attractor in a given AND/OR Boolean network?
     - An $O(1.757^n)$ time algorithm is presented in this talk.
       - This improves the previous $O(1.787^n)$ time algorithm. (Tamura and Akutsu, FCT2007)



What is a Boolean network (BN)?

- Mathematical model of genetic network
- Very simple model
  - Each node takes either 0 or 1.
    - Node → gene
    - 1 → active, 0 → inactive
  - States of nodes change synchronously
    - According to regulation rules (= Boolean functions)

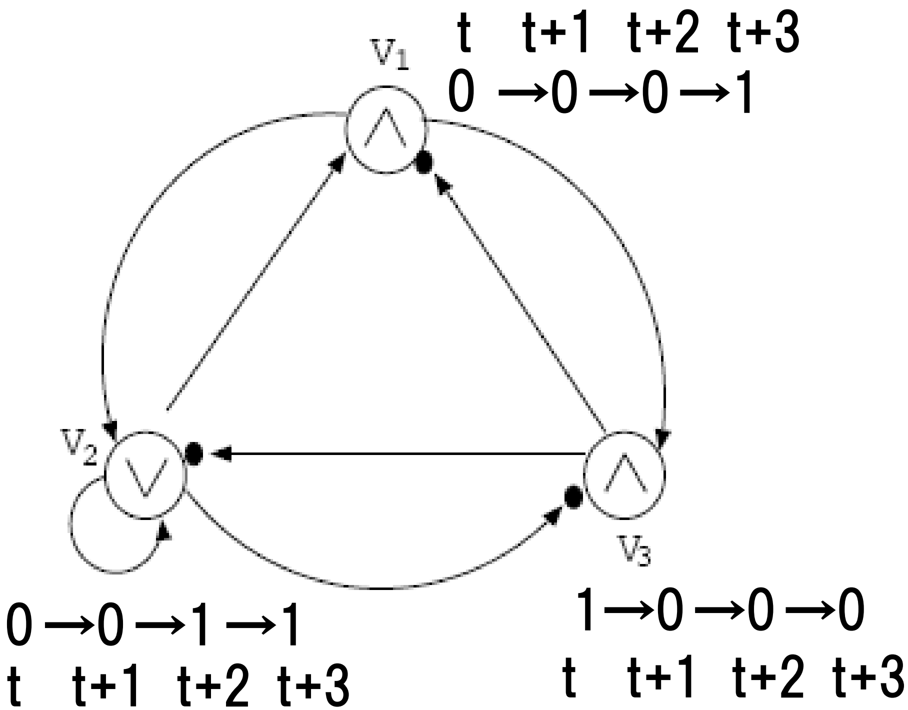
AND/OR BN



AND/OR BN
Regulation rules are limited to disjunction or conjunction of parent nodes.

$$v_1(t+1) = v_2(t) \wedge \overline{v_3(t)}$$
$$v_2(t+1) = v_1(t) \vee v_2(t) \vee \overline{v_3(t)}$$
$$v_3(t+1) = v_1(t) \wedge \overline{v_2(t)}$$

Example of AND/OR BN



Gene Activity Profile (GAP)

$[v_1, v_2, v_3]$ =[0,0,1]

↓

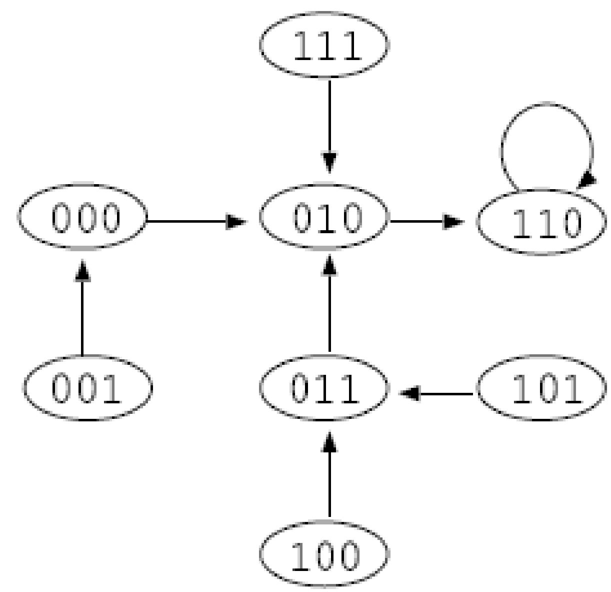
[0,0,0]

↓

[0,1,0]

↓

[1,1,0]

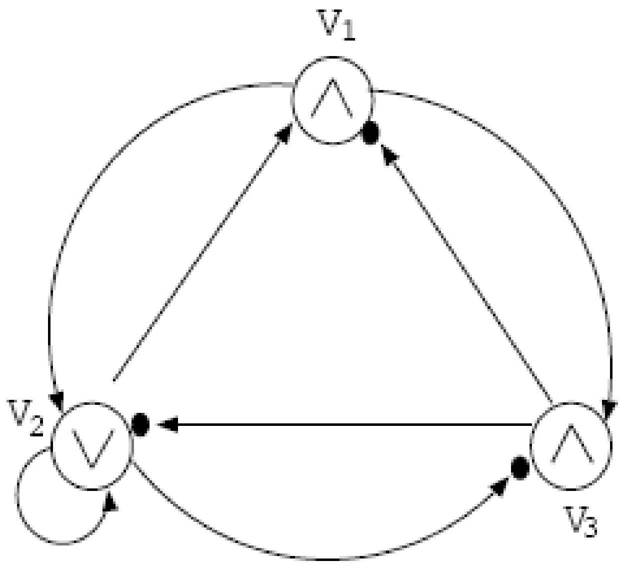


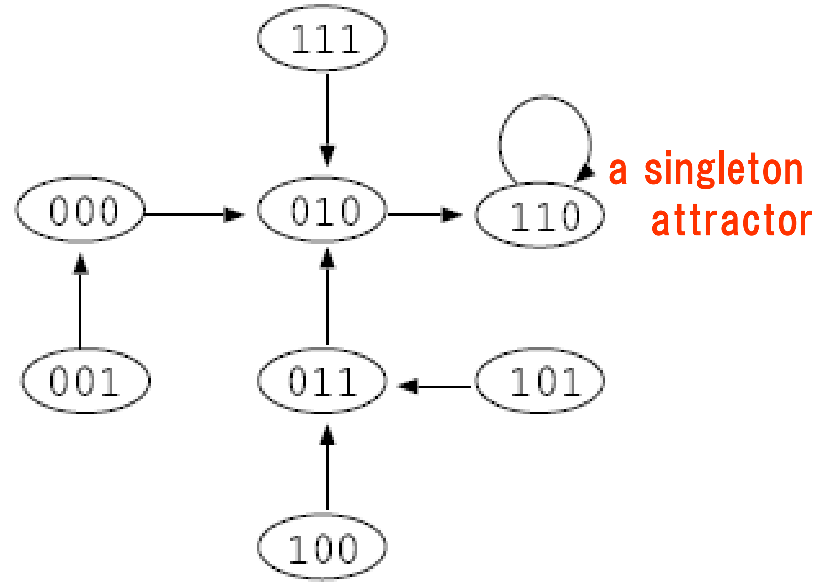
$$v_1(t+1) = v_2(t) \wedge \overline{v_3(t)}$$

$$v_2(t+1) = v_1(t) \vee v_2(t) \vee \overline{v_3(t)}$$

$$v_3(t+1) = v_1(t) \wedge \overline{v_2(t)}$$

What is a singleton attractor (fixed point)?

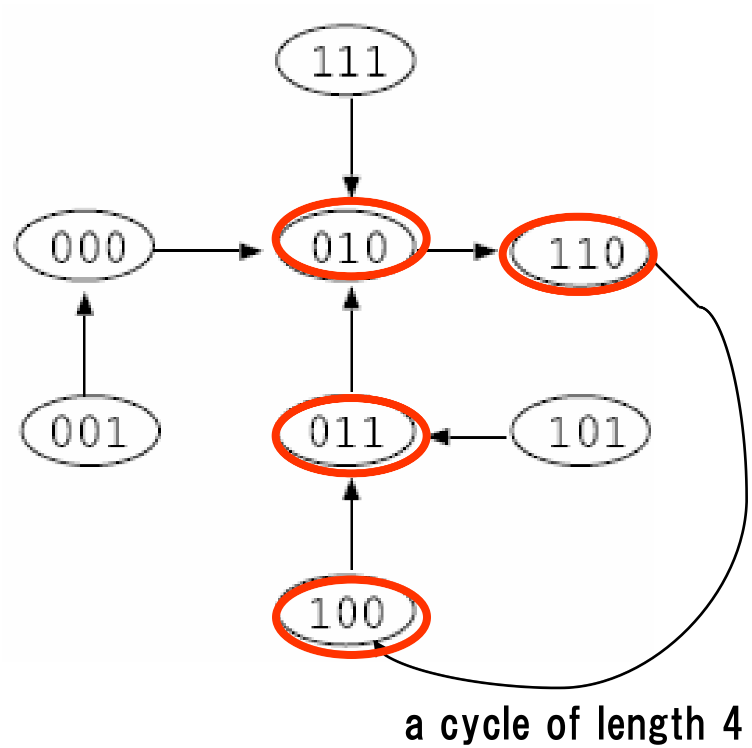


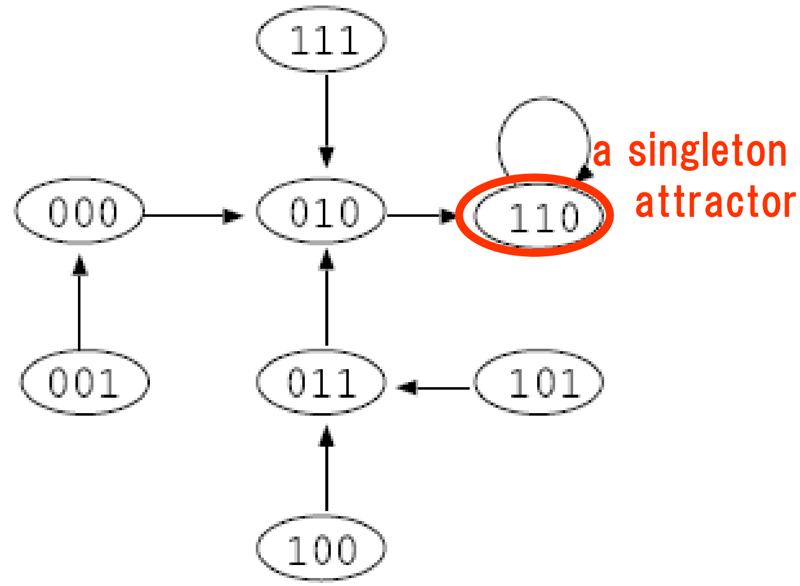


- [V_1, V_2, V_3]=[1, 1, 0] → a singleton attractor
  - The state of [1,1,0] never changes.
  - [1,1,0] has a self-loop in the state-transition.

  - One of the most stable states
    - play an important role in biological systems

Cyclic attractor





・[0,1,0]→[1,1,0]→[1,0,0]→[0,1,1]
・An attractor with period 4
(cyclic attractor)

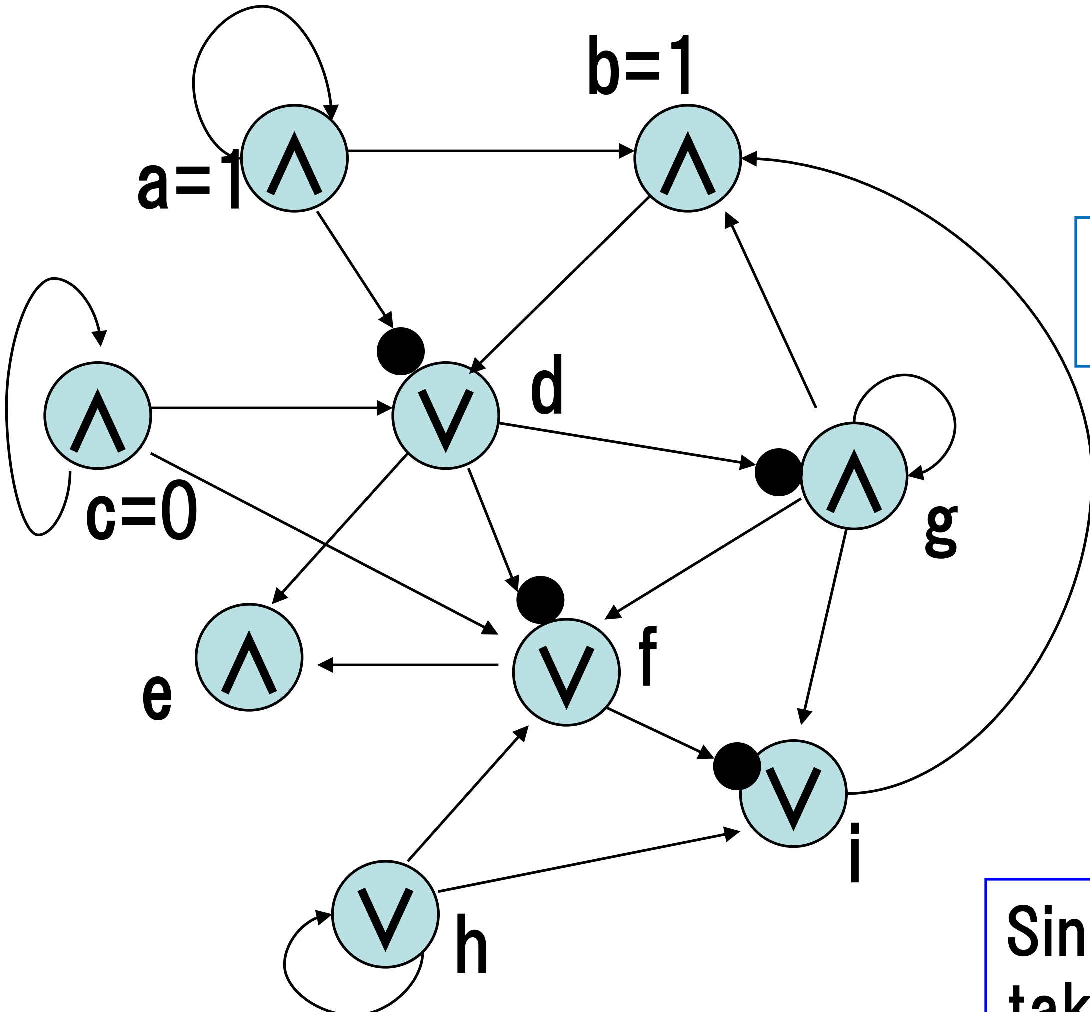
・[1,1,0]
・An attractor with period 1
(singleton attractor)

・In this talk, we deal with only singleton attractors.

Singleton attractor detecting problem

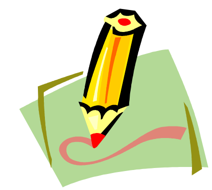
- Is there a singleton attractor in a Boolean network (BN)?
  - For random BN
    - An $O(1.19^n)$ time algorithm is known with maximum indegree 2 (Zhang et al. 2007).
    - However, it may take $O(2^n)$ or more time in the worst case.
    - Worst case analysis is necessary.
  - For the worst case
    - NP-hard (Akutsu et al. 1998)
    - If the maximum indegree is K, the problem can be reduced to (K+1)-SAT.
    - If K is not limited, no $O((2-\epsilon)^n)$ $(\epsilon > 0)$ time algorithms are known
      - Even for AND/OR BN, no $O((2-\epsilon)^n)$ $(\epsilon > 0)$ time algorithms had been known until we proposed an $O(1.787^n)$ time algorithm. (Tamura and Akutsu, FCT2007)
      - In this talk, $O(1.757^n)$ time algorithm is presented.

Previous algorithm (Tamura and Akutsu, FCT2007)



① assign values to all nodes
② consistency checking

Singleton attractor
→values of nodes never change.

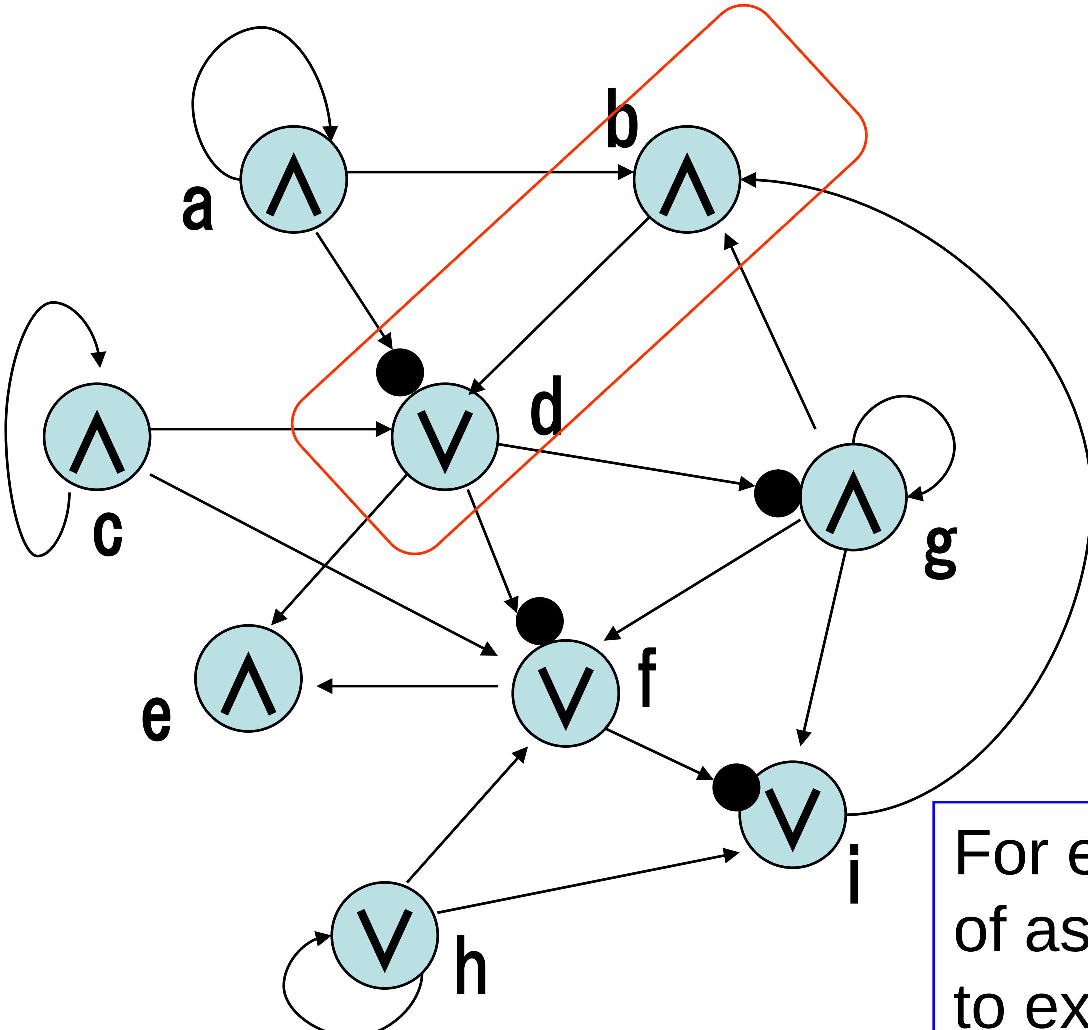


The consistency checking can be done in $O(n^2)$ time.

Consistency checking for node d
-d=1 → OK
-d=0 → contradiction

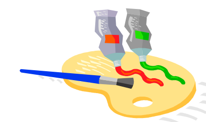
Since the main algorithm takes exponential time, we can ignore the time for consistency checking.

Previous Algorithm (Tamura and Akutsu, 2007)



① assign values to all nodes
② consistency checking

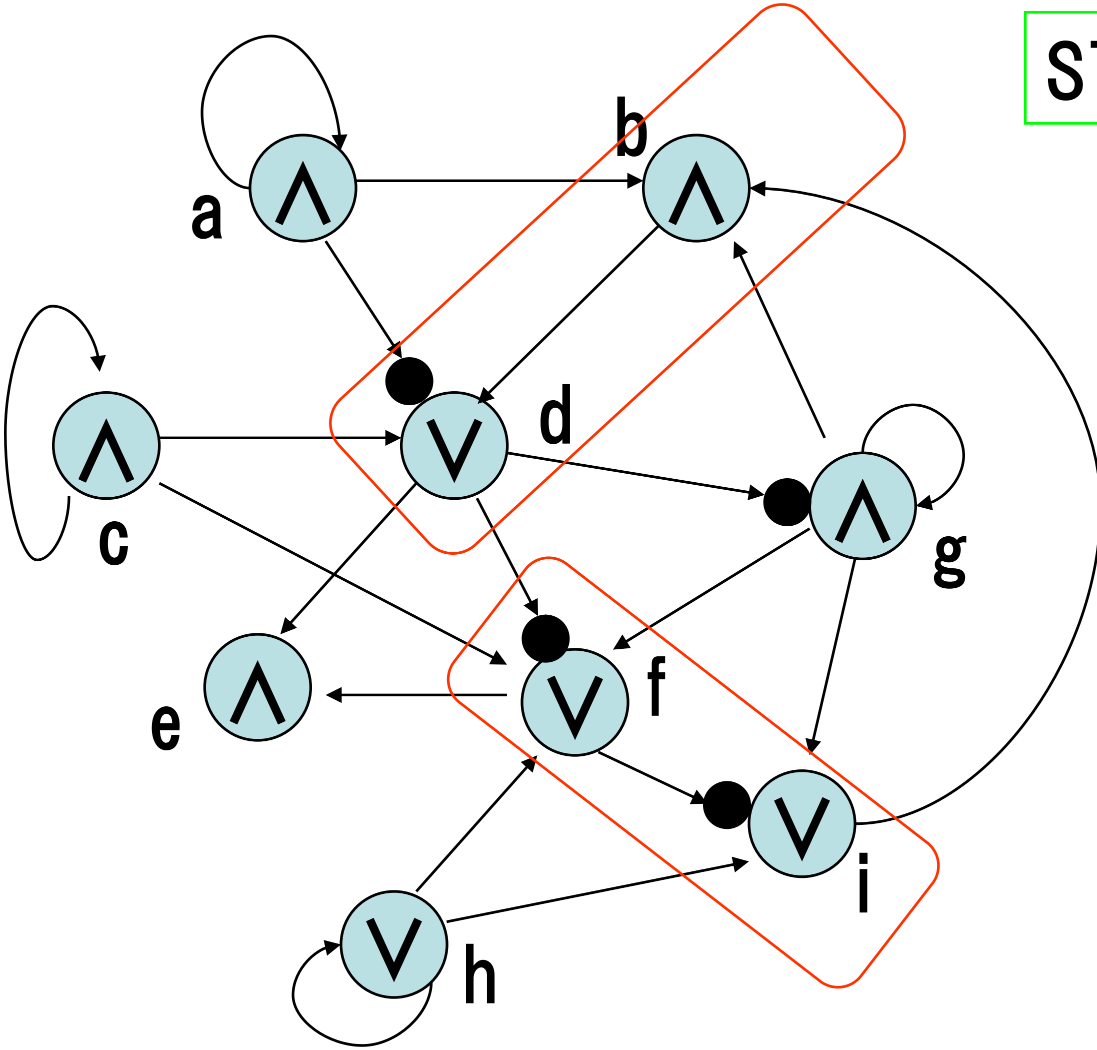
If all assignment are examined, it takes $O(2^n)$ time.



If (b,d)=[1,0], the value of d changes from 0 to 1.
It contradicts the condition of a singleton attractor.

For every node pair, the number of assignments which we have to examine is at most 3 of 4 assignments

By using this fact, we can reduce the computational time.



STEP 1 of the previous algorithm

Initial state:
All nodes are non-assigned

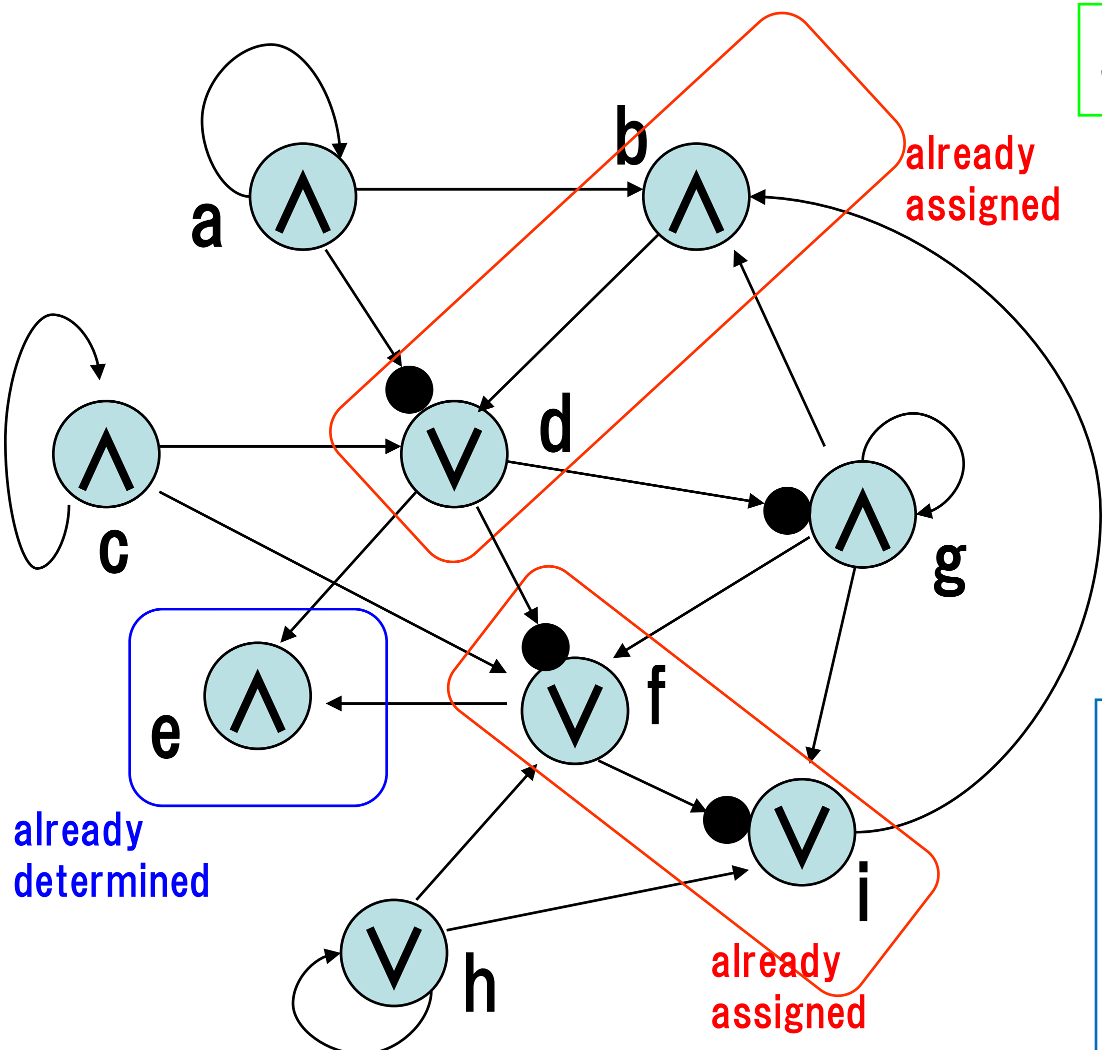
While there exists a non-assigned edge (u,v), examine all possible 3 assignments on (u,v).

Possible assignments for (b,d) are [0,0], [0,1] and [1,1]. Note that [1,0] is not allowed.

Possible assignments for (f,i) are [0,1], [1,0] and [1,1]. Note that [0,0] is not allowed.

When K nodes are assigned, the number of cases are bounded by $f(K)=3 \cdot f(K-2)$, $f(2)=3$.

Then, $f(K)$ is $O(3^{K/2})$, which is at most $O(1.733^K)$.



STEP 2 of the previous algorithm

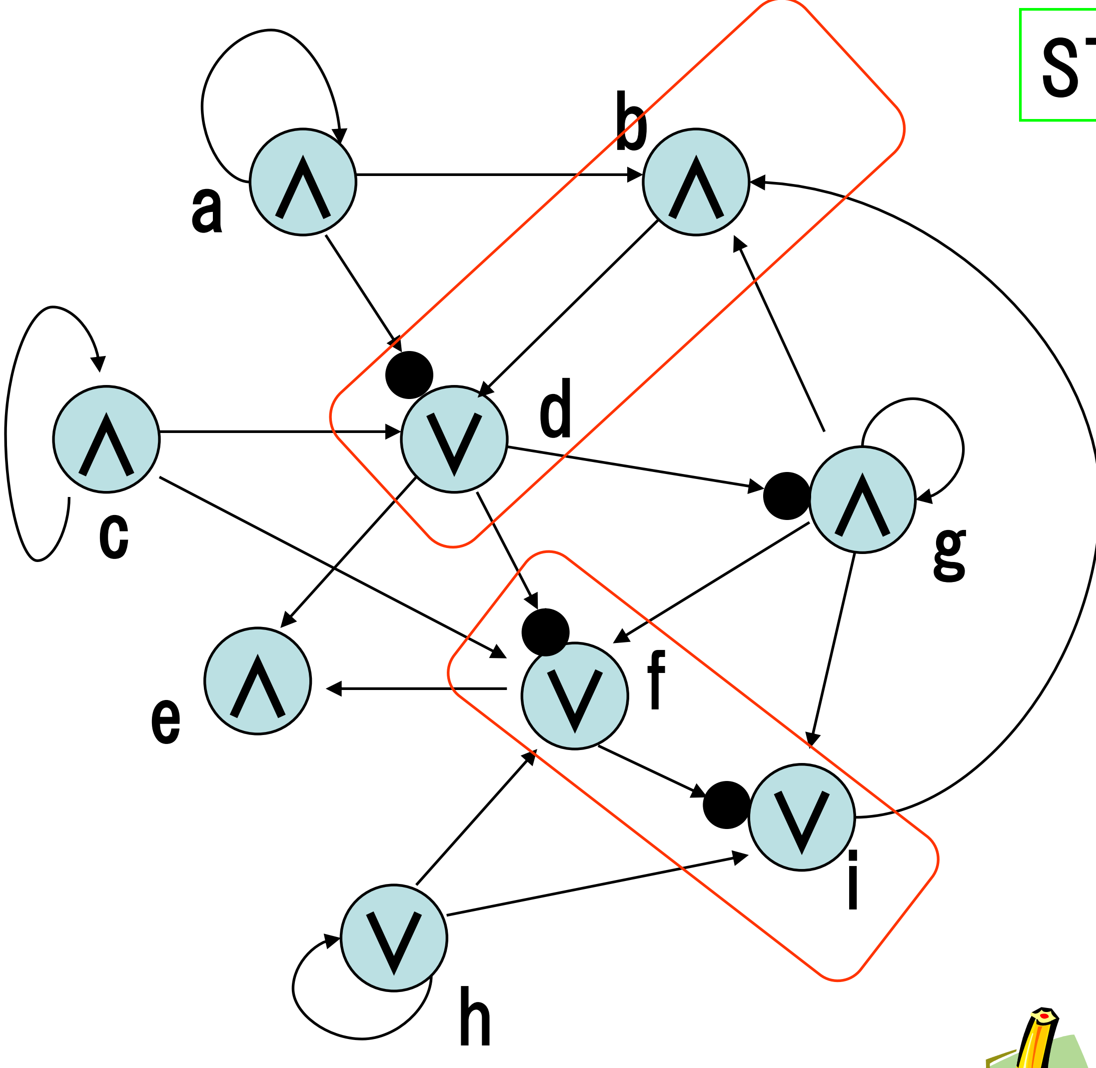
Let W be nodes whose values have not been determined yet.

If $|W| \leqq n - \alpha n$, examine all possible assignments on W

For example, a,c,g,h ∈W

All 2^4 assignments for a,c,g,h are examined if STEP2 is executed.

If STEP 2 is executed, the computational time is at most $O(2^{n-K} \cdot 1.733^K)$.



STEP 3 of the previous algorithm

If $|W|>n-\alpha n$, solve a SAT problem.

If (b,d)=[0,1] is assigned, $(\bar{a}\vee g)(\bar{a}\vee c)=1$ must be satisfied.

If (f,i)=[1,1] is assigned, $(c\vee g\vee h)(g\vee h)=1$ must be satisfied.

When K nodes are assigned, the condition of a singleton attractor can be represented by at most K clauses.

SAT problem with K clauses can be solved in $\tilde{O}(1.234^K)$ time.
where $\tilde{O}(f(m))$ means $O(f(m)poly(m,n))$. (Yamamoto, 2005).

→the overall computational time is bounded by $O(1.234^K \cdot 1.733^K)$.

Theorem 1

The detection of a singleton attractor can be done in $O(1.792^n)$-time for AND/OR BNs. (worst case)

After STEP1

if $|w| \leqq n - \alpha n$,

then STEP 2 is executed.

the computational time is $O(2^{n-K} \cdot 1.733^K)$.

else, STEP 3 is executed.

the computational time is $O(1.234^K \cdot 1.733^K)$.

By setting K=0.767n (α=0.767),

$$2^{n-0.767n} \cdot 1.733^{0.767n} < 1.792^n$$

$$1.234^{0.767n} \cdot 1.733^{0.767n} < 1.792^n$$

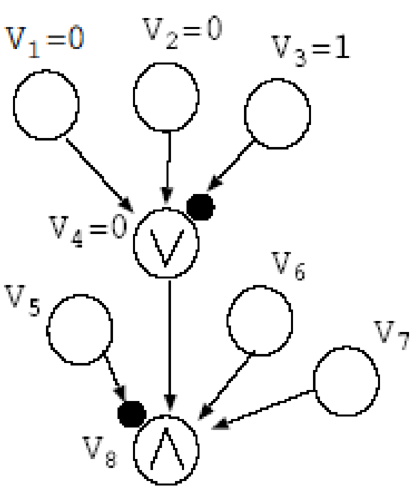
are obtained.

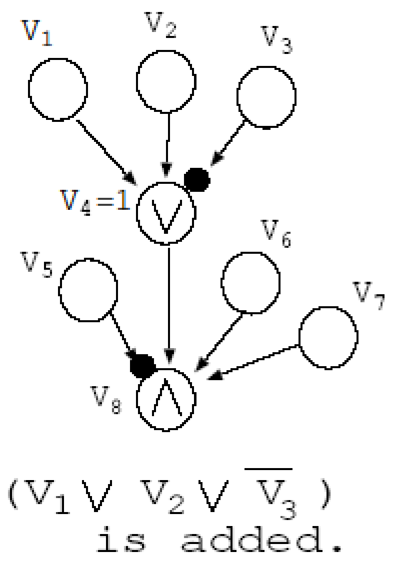
Improved analysis

In the previous analysis,
the number of SAT clauses constructed in STEP 1
is estimated as same as
the number of assigned nodes in STEP 1.

However, there are cases in which SAT clauses are not constructed.

example





When 0 is assigned to v4,
no SAT clauses are constructed

When 1 is assigned to v4,
a SAT clause is constructed.

Theorem 2

Detection of a singleton attractor can be done in $O(1.787^n)$ -time for AND/OR BNs.

(Tamura and Akutsu, FCT2007)

After STEP1

if $|w| \leqq n - \alpha n$,

then STEP 2 is executed.

the computational time is $O(2^{n-K} \cdot 1.733^K)$.

else, STEP 3 is executed.

the computational time is $O(2.089^K)$.

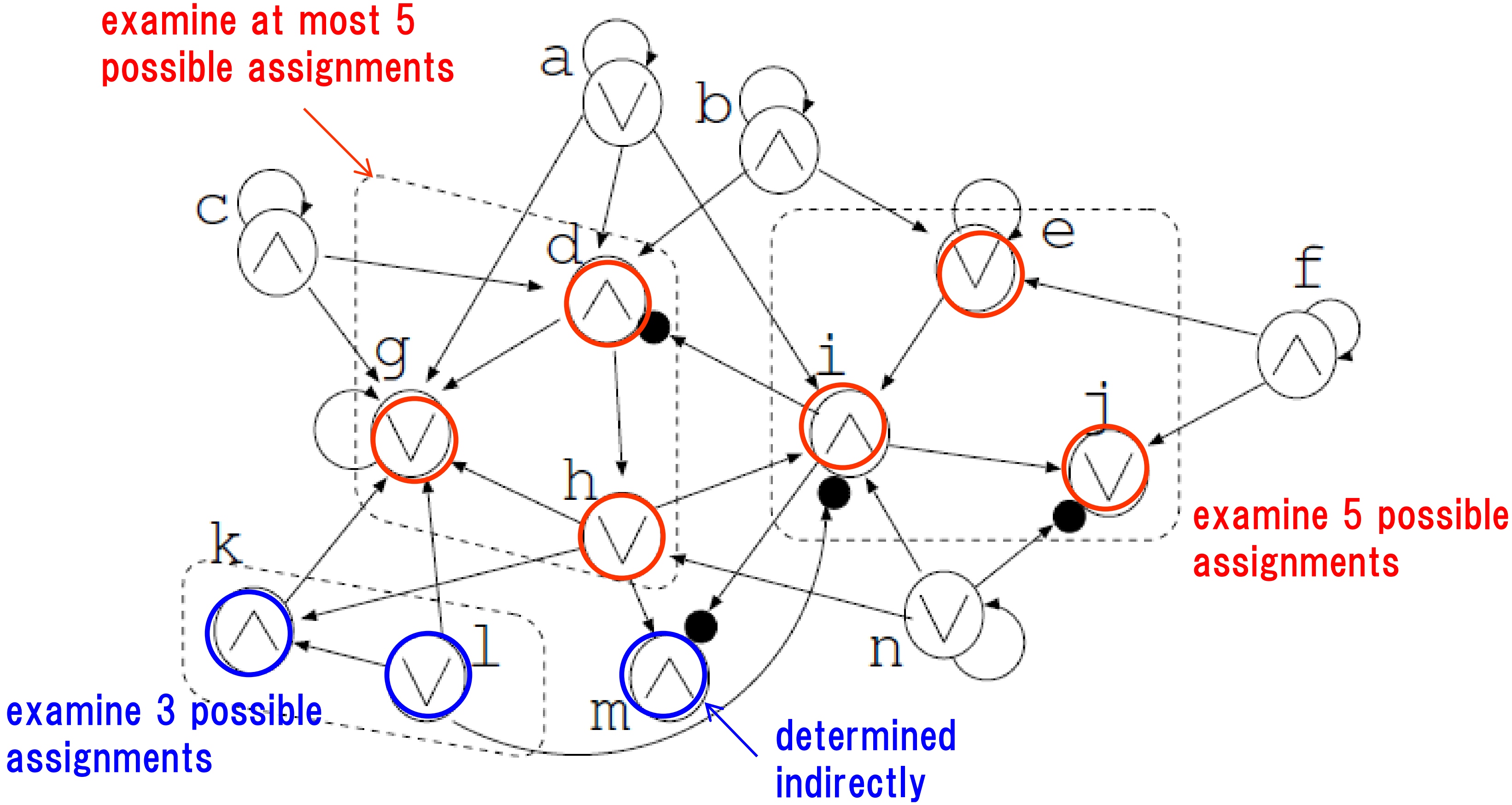
By setting K=0.7877n (α=0.7877),

$$2^{n-0.7877n} \cdot 1.733^{0.7877n} < 1.7866^n$$

$$2.089^{0.7877n} < 1.7866^n$$

are obtained.

More improved algorithm (main topic)

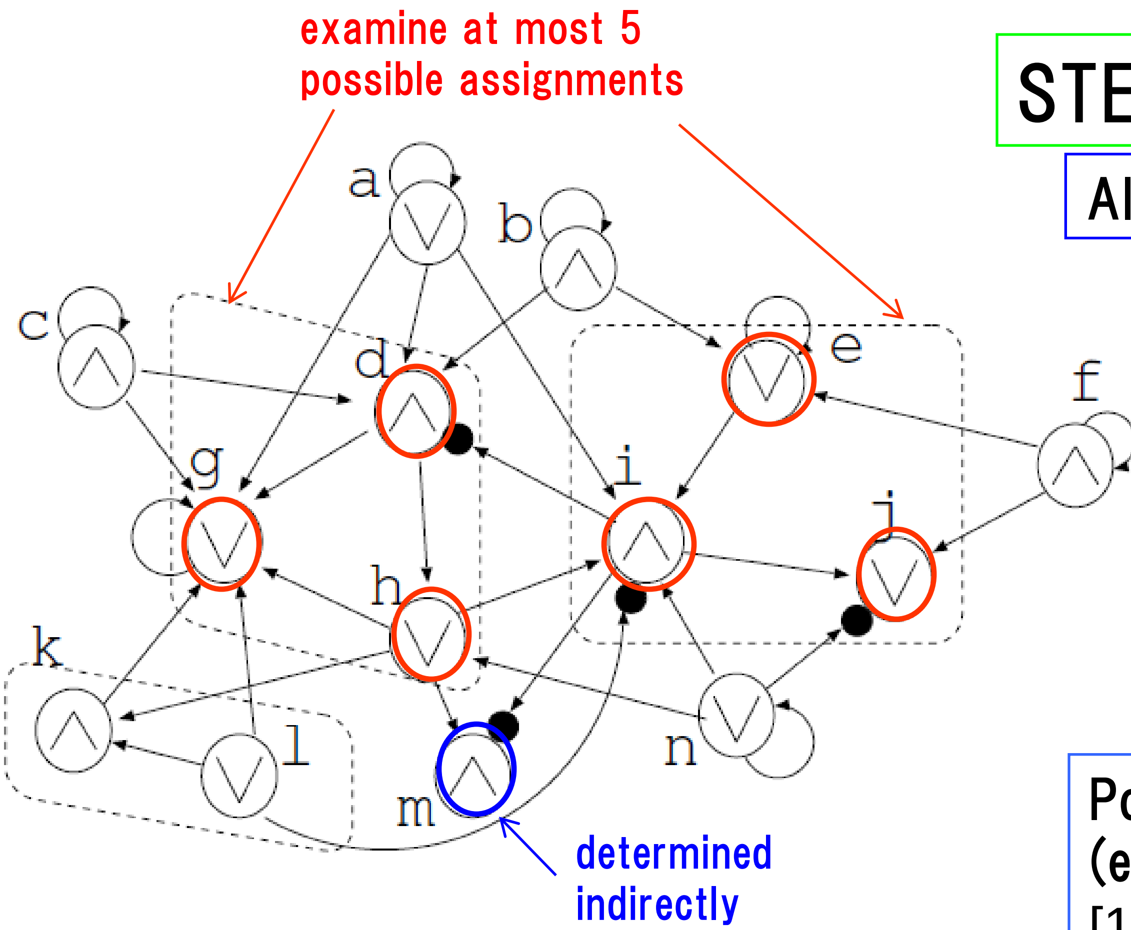


While there exist non-assigned neighboring edges, examine all possible assignment, which are at most 5.

For example, possible assignments for (e,i,j) are [0,0,0],[0,0,1],[1,0,0],[1,0,1],[1,1,1]
since [0,1,0],[0,1,1],[1,1,0] are impossible assignments.

STEP 1 of the proposed algorithm

All nodes are non-assigned



While there exists a non-assigned neighboring edges {(u,v),(v,w)}, examine all possible 5 assignments on (u,v,w).

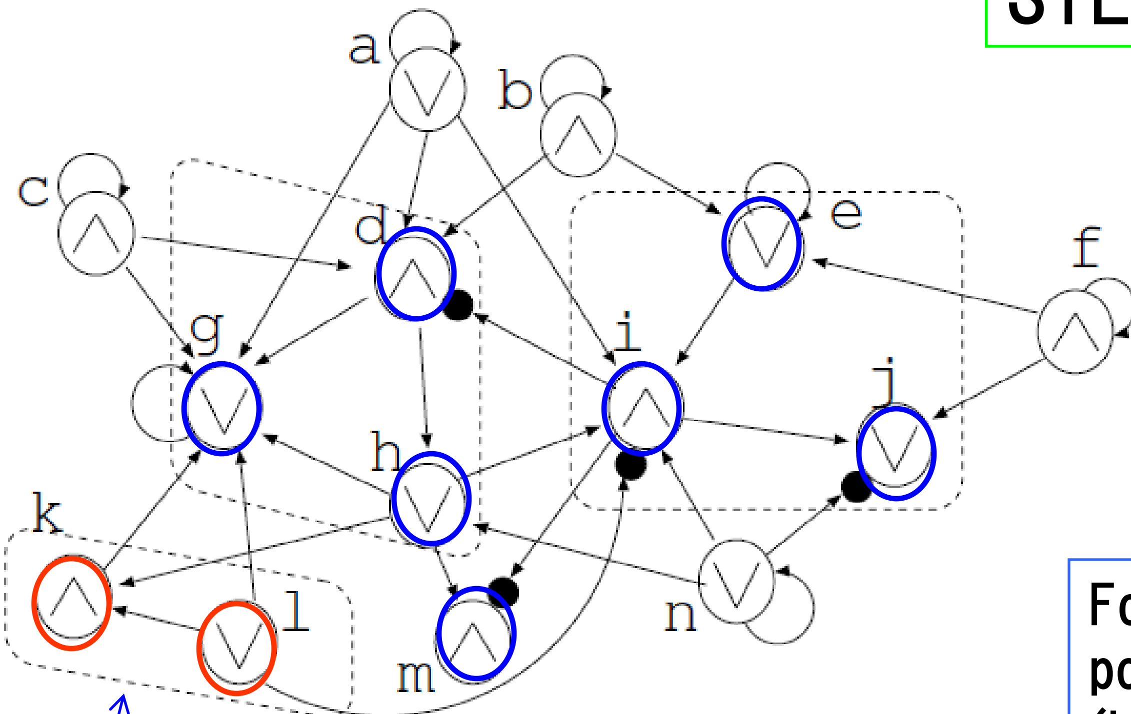
Possible assignments for (e,i,j) are [0,0,0],[0,0,1],[1,0,0], [1,0,1] and [1,1,1].

Possible assignments for (d,g,h) are [0,0,0],[0,1,0],[0,1,1][1,1,1]. Impossible assignments are [1,0,0],[1,0,1],[0,0,1],[1,1,0].

When K nodes are assigned, the number of cases are bounded by f(K)=5・f(K-3), f(3)=5.

Then, $f(K)$ is $O(5^{K/3})$, which is at most $O(1.710^K)$ $(=O(1.71^K))$.

STEP 2 of the proposed algorithm

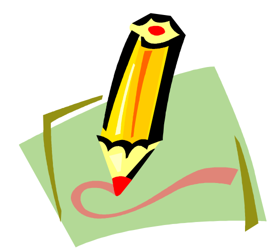


examine (at most) 3 possible assignments

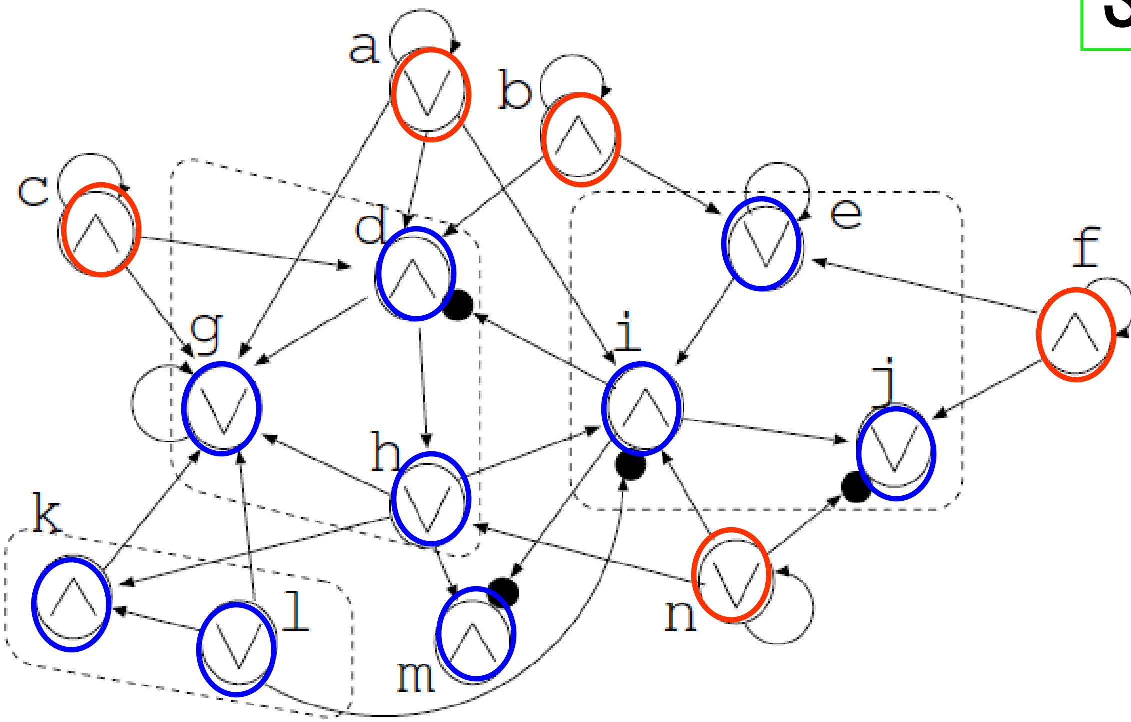
While there exists a non-assigned edge (u,v), examine all possible 3 assignments on (u,v).

For example, possible assignments for (k,l) are [0,0], [0,1] and [1,1]. Note that [1,0] is not allowed.

When L nodes are assigned, the number of cases are bounded by f(L)=3・f(L-2), f(2)=3.



Then, $f(L)$ is $O(3^{L/2})$, which is at most $O(1.733^L)$.



STEP 3

Let **W** be nodes whose values have not been determined yet.

If $K > \alpha(n-L)$,
examine all possible assignments on W

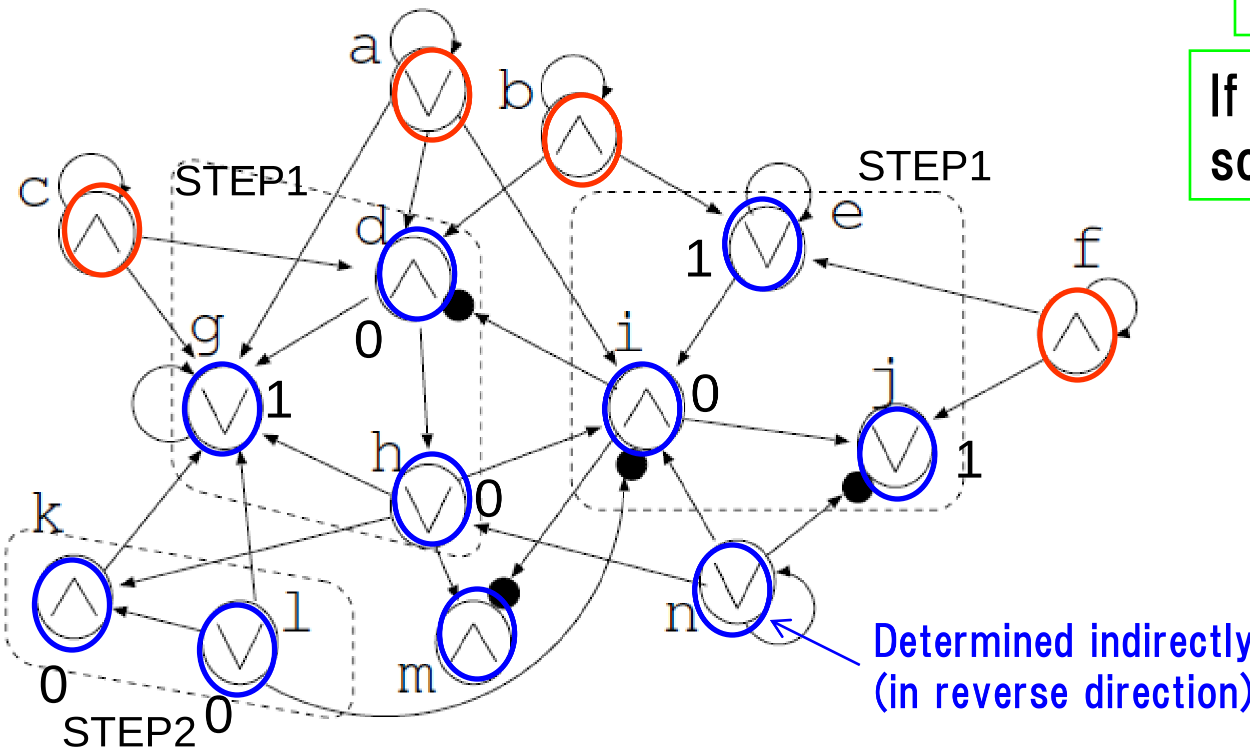
Note that values of red-circled nodes may be determined indirectly.

The consistency checking can be done in polynomial time.

If STEP 3 is executed, the computational time is at most $O(2^{n-K-L} \cdot 1.71^{K} \cdot 1.733^{L})$.

STEP 4

If K≦α(n-L),
solve a **SAT problem**.



d=0 → $(\overline{a} \vee \overline{b} \vee \overline{c})$

g=1 → $(a \vee c)$

h=0 → n=0

e=1 → $(b \vee f)$

i=0 → $(\overline{a} \vee \overline{n})$

j=1 → $(f \vee \overline{n})$

When K nodes are assigned in STEP1, **the condition of a singleton attractor** can be represented by at most K clauses.

Note that STEP2 never adds SAT clauses.

SAT problem with K clauses can be solved in $\tilde{O}(1.234^K)$ time.
where $\tilde{O}(f(m))$ means $O(f(m)poly(m,n))$. (Yamamoto, 2005).

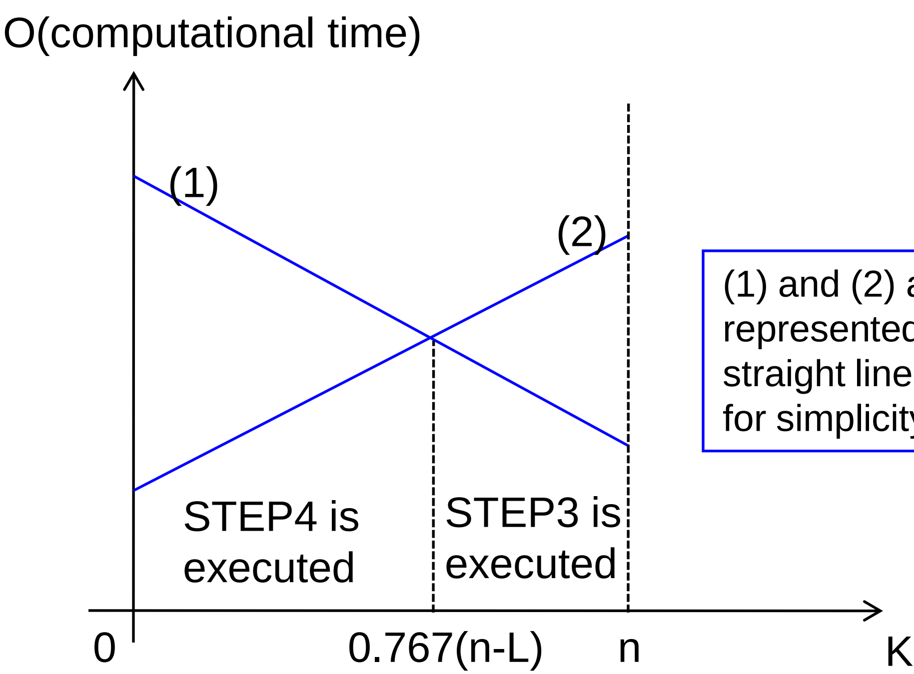
→**the overall computational time** is bounded by $O(1.234^K \cdot 1.71^K \cdot 1.733^L)$.

After STEP1 and STEP2
if K> α(n-L),
then STEP 3 is executed.
the computational time is

$$O(2^{n-K-L} \cdot 1.71^K \cdot 1.733^L) \quad (1)$$

else if STEP 4 is executed.
the computational time is

$$O(1.234^K \cdot 1.71^K \cdot 1.733^L) \quad (2)$$



(1) and (2) are represented in straight lines for simplicity.

Assume that L is obtained. If n is large enough constant,

$O(2^{n-K-L} \cdot 1.71^K \cdot 1.733^L)$ $\rightarrow$ Monotone decreasing with K

$O(1.234^K \cdot 1.71^K \cdot 1.733^L)$ $\rightarrow$ Monotone increasing with K

Therefore, the computational time is bounded by that of the case where $1.234^K = 2^{n-K-L}$ holds. $\rightarrow K = 0.767n - 0.767L$

Thus the computational time of the proposed algorithm can be bounded by

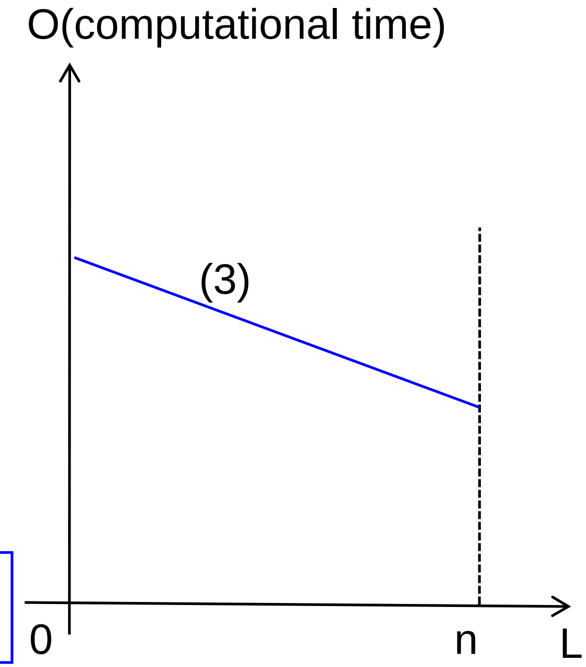
$$\max_{0 \leq L \leq n} \{1.234^{0.767n-0.767L} \cdot 1.71^{0.767n-0.767L} \cdot 1.733^L\}$$

Since $(1.234 \cdot 1.71)^{0.767} = 1.773 > 1.733,$

$$\max_{0 \le L \le n} \{1.234^{0.767n - 0.767L} \cdot 1.71^{0.767n - 0.767L} \cdot 1.733^{L}\} \quad (3)$$

is a monotone decreasing function of L
if n is a large enough constant.

Therefore, (3) takes the maximum value when L=0.



(3) is represented in straight lines for simplicity.

Thus, the computational time of the proposed algorithm can be bounded by

(by assigning L=0 to (3))

$$1.234^{0.767n} \cdot 1.71^{0.767n} < 1.774^{n}.$$

Therefore,

The detection of a singleton attractor can be done in $O(1.774^n)$ -time for AND/OR BNs.

Theorem 3

The detection of a singleton attractor can be done in $O(1.774^n)$ -time for AND/OR BNs.

After STEP1

if K>0.767(n-L),

then STEP 3 is executed.

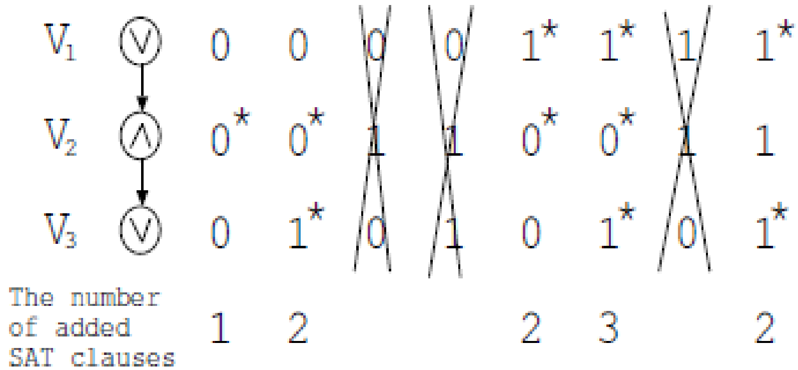
the computational time is $O(2^{n-K-L} \cdot 1.71^K \cdot 1.733^L)$

else if STEP 4 is executed.

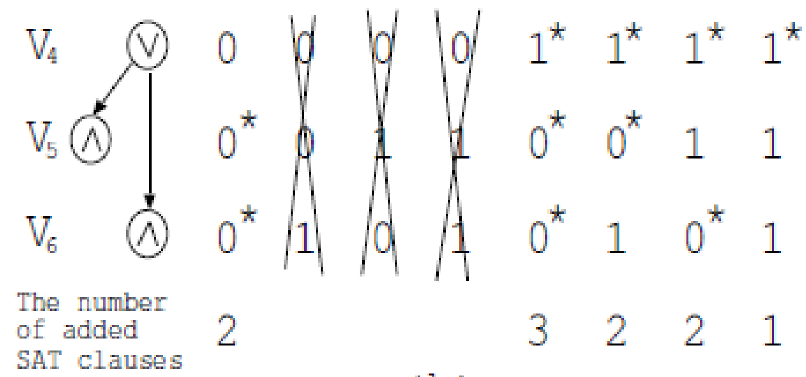
the computational time is $O(1.234^K \cdot 1.71^K \cdot 1.733^L)$

Improved analysis (especially in STEP4)

There are cases where SAT clauses are not constructed.

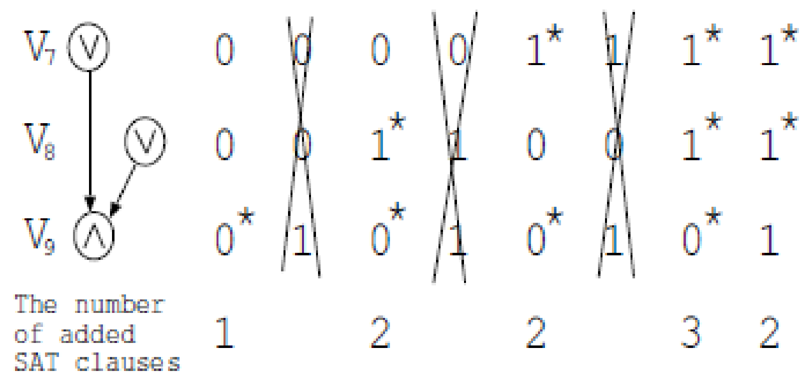


(a)



(b)

Note that negations can be erased by De Morgan's law.



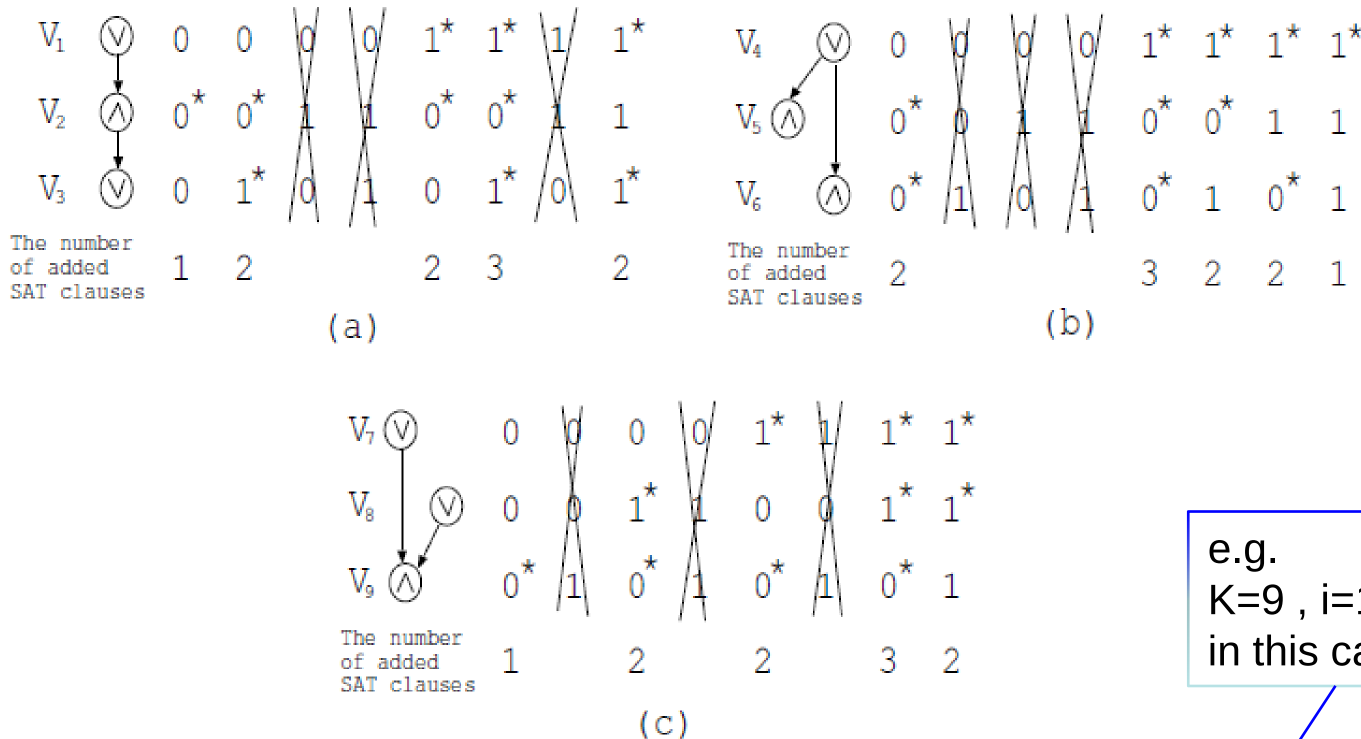
(c)

The worst case is as follows:

(1) One of the five assignments adds one clause.
(2) Three of the five assignments add two clauses.
(3) One of the five assignments adds three clauses.

The number of cases generated by STEP1 is $O(5^{\frac{K}{3}})$ $(\leq O(1.71^{K}))$.

For each case of them, the number of added SAT clauses is determined according to which one of five assignments is selected in each non-assigned neighboring edges.



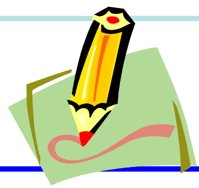
e.g.
K=9 , i=1, j=0
in this case

For example, if [v1,v2,v3]=[1,0,0] ,[v4,v5,v6]=[1,1,1],[v7,v8,v9]=[0,1,0] assigned, the total number of added SAT clauses is 5 (=2+1+2).

Therefore, the number of cases where one clause is added i times and three clauses are added j times is

$$\sum_{i=0}^{\frac{K}{3}} \sum_{j=0}^{\frac{K}{3}-i} 3^{(\frac{K}{3}-i-j)} \cdot {}_{\frac{K}{3}}C_i \cdot {}_{\frac{K}{3}-i}C_j$$

since the number of cases where two clauses are added is $3^{\frac{K}{3}-i-j}$.



Moreover, the total number of added SAT clauses in this case is $\frac{2K}{3} - i + j$.

Therefore, the computational time when STEP4 is executed is bounded by

$$g(K,L) = 1.733^L \cdot \sum_{i=0}^{\frac{K}{3}} \sum_{j=0}^{\frac{K}{3}-i} 1.234^{(\frac{2K}{3}-i+j)} \cdot 3^{(\frac{K}{3}-i-j)} \cdot {}_{\frac{K}{3}}C_i \cdot {}_{\frac{K}{3}-i}C_j.$$

STEP2

Note that STEP2 does not construct any SAT clauses.

Although the proof is omitted in today's talk, it can be proved that

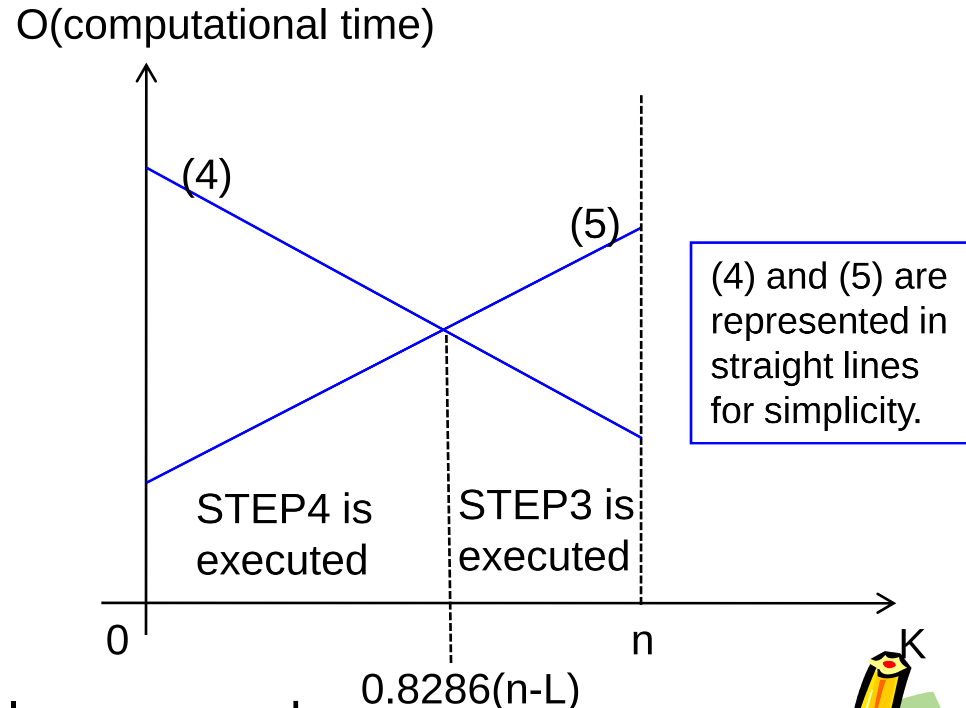
O(g(K,L))< $O((1.234^2 \cdot 3 \cdot 1.683)^{\frac{K}{3}} \cdot 1.733^L)$ < $O(1.974^K \cdot 1.733^L)$

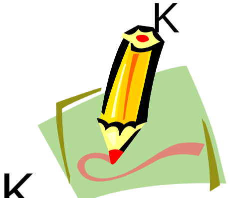
After STEP1 and STEP2
if $K > \alpha(n-L)$,
then STEP 3 is executed.
the computational time is

$$O(2^{n-K-L} \cdot 1.71^K \cdot 1.733^L) \quad (4)$$

else if STEP 4 is executed.
the computational time is

$$O(1.974^K \cdot 1.733^L) \quad (5)$$





Assume that L is obtained. If n is large enough,

$O(2^{n-K-L} \cdot 1.71^K \cdot 1.733^L)$ → Monotone decreasing with K

$O(1.974^K \cdot 1.733^L)$ → Monotone increasing with K

Therefore, the computational time is bounded by that of the case where

$1.974^K = 2^{n-K-L} \cdot 1.71^K$ holds. → $K = 0.8286n - 0.8286L$

Thus the computational time of the proposed algorithm can be bounded by

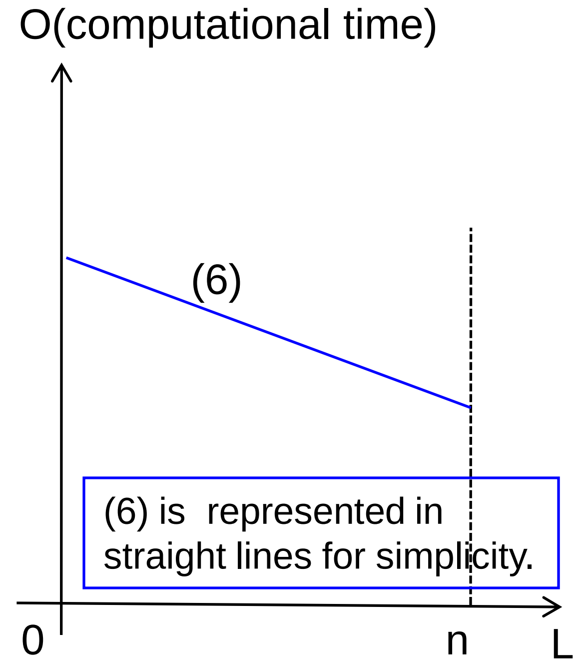
$$\max_{0 \le L \le n} \{1.974^{0.8286n - 0.8286L} \cdot 1.733^L\}$$

Since $1.974^{0.8286} = 1.757 > 1.733$,

$$\max_{0 \le L \le n} \{1.974^{0.8286n - 0.8286L} \cdot 1.733^L\} \quad (6)$$

is a monotone decreasing function of L
if n is a large enough constant.

Therefore, (1) takes the maximum value
when L=0.



Thus, the computational time of the proposed algorithm can be bounded by (by assigning L=0 to (1))

$$O(1.974^{0.8286n}) < O(1.757^n) .$$

Therefore,

The detection of a singleton attractor can be done in $O(1.757^n)$-time for AND/OR BNs.

Theorem 4

The detection of a singleton attractor can be done in $O(1.757^n)$ -time for AND/OR BNs.

After STEP1

if K>0.8286(n-L),

then STEP 3 is executed.

the computational time is $O(2^{n-K-L} \cdot 1.71^K \cdot 1.733^L)$

else if STEP 4 is executed.

the computational time is $O(1.234^K \cdot 1.71^K \cdot 1.733^L)$

Concluding remarks

- Is there a singleton attractor in a given AND/OR Boolean network (AND/OR BN)?
  - An $O(1.774^n)$ time algorithm was presented and then it was improved to $O(1.757^n)$ time in this talk.
    (made use of 3 adjacent nodes)
  - The previous known result was $O(1.787^n)$.
    (made use of 2 adjacent nodes, Tamura and Akutsu, 2007)
  - It is unclear whether further improvement by making use of 4 or more adjacent nodes is possible.
    - At least, the algorithm and analysis would be quite involved.
  - Thus, improvement of the proposed algorithm is left as an open problem.
  - AND/OR BN is considered to be a good model since canalizing functions and nested canalizing functions, which are slightly more involved than AND/OR BN, are known to be one of the most suitable models for regulatory rules of eukaryotic genes.

f is a canalizing function if either

$$v_j = f(v_1, \ldots, v_n) = v_i^r \vee g(v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n)$$

or

$$v_j = f(v_1, \ldots, v_n) = v_i^r \wedge g(v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n)$$

holds.