เอกสารอ้างอิง
(References)

1. Martin S, Zhang Z, Martino A, Faulon J-L: **Boolean Dynamics of Genetic Regulatory Networks Inferred from Microarray Time Series Data**. *Bioinformatics* 2007, **23**:866-874.
2. Cseke LJ, Kirakosyan A, Kaufman PB, Warber SL, Duke JA, Brielmann HL: **Natural Products from Plants**. *Molecular regulation* 2006:587.
3. Theuns, Jessie, VanBroeckhoven, Christine: **Transcriptional regulation of Alzheimer's disease genes: implications for susceptibility** *Human Molecular Genetics* 2000, **9**(16):2383-2394.
4. Villard J: **Transcription regulation and human diseases**. *SWISS MED WKLY* 2004, **134**:571-579.
5. Long TA, Brady SM, Benfey PN: **Systems Approaches to Identifying Gene Regulatory Networks in Plants**. *Annual Review of Cell and Developmental Biology* 2008, **24**:81-103
6. Shen-Orr SS, Milo R, Mangan S, Alon U: **Network motifs in the transcriptional regulation network of *Escherichia coli***. *nature genetics* 2002, **31**:64-68.
7. Babu MM: **Computational approaches to study transcriptional regulation**. *Biochemical Society Transactions* 2008, **36**:758-765.
8. Barrett CL, Palsson BO: **Iterative Reconstruction of Transcriptional Regulatory Networks: An Algorithmic Approach**. *PLoS Computational Biology* 2006, **2**(5):429-438.
9. Kwon AT, Hoos HH, Ng R, : **Inference of transcriptional regulation relationships from gene expression data**. In: *Proceedings of the 2003 ACM symposium on Applied computing 2003; Melbourne, Florida* ACM New York, NY, USA 135 - 140
10. Perrin B, Ralaivola L, Mazurie A, Bottani S, Mallet J, d'Alché-Buc F: **Gene networks inference using dynamic Bayesian networks**. *Bioinformatics* 2003, **9**(2):138-148.
11. Kauffman SA: **The Origins of Order: Self-Organization and Selection in Evolution**. Oxford University Press, New York; 1993.
12. Mehra S, Hu W-S, Karypis G: **A Boolean algorithm for reconstructing the structure of regulatory networks**. *Metabolic Engineering* 2004, **6**:326-339.
13. Pensa R, Leschi C, Besson J, Boulicaut J-F: **Assessment of discretization techniques for relevant pattern discovery from gene expression data**. In: *In 4th ACM SIGKDD Workshop on Data Mining in Bioinformatics*. In Press.; 2004: 24-30.

14. Smith SM, Fulton DC, Chia T, Thorneycroft D, Chapple A, Dunstan H, Hylton C, Zeeman SC, Smith AM: **Diurnal Changes in the Transcriptome Encoding Enzymes of Starch Metabolism Provide Evidence for Both Transcriptional and Posttranscriptional Regulation of Starch Metabolism in Arabidopsis Leaves**. *Plant Physiology* 2004, **136**:2687-2699.
15. DeRisi JL, Iyer VR, Brown PO: **Exploring the Metabolic and Genetic Control of Gene Expression on a Genomic Scale**. *Science* 1997, **278**:680-686.
16. Ideker T, Thorsson V, Ranish JA, Christmas R, Buhler J, Eng JK, Bumgarner R, Goodlett DR, Aebersold R, Hood L: **Integrated genomic and proteomic analyses of a systematically perturbed metabolic network**. *Science* 2001, **292**(5518):929-934.
17. Li C, Wong WH: **Model-based analysis of oligonucleotide arrays: expression index computation and outlier detection**. *Proceedings of the National Academy of Sciences of the United States of America* 2001, **98**(1):31-36.
18. Workman C, Jensen LJ, Jarmer H, Berka R, Gautier L, Nielser HB, Saxild HH, Nielsen C, Brunak S, Knudsen S: **A new non-linear normalization method for reducing variability in DNA microarray experiments**. *Genome biology* 2002, **3**(9):research0048.
19. Storey JD, Xiao W, Leek JT, Tompkins RG, Davis RW: **Significance analysis of time course microarray experiments**. *Proceedings of the National Academy of Sciences of the United States of America* 2005, **102**(36):12837-12842.
20. Shannon P, Markiel A, Ozier O, Baliga NS, Wang JT, Ramage D, Amin N, Schwikowski B, Ideker T: **Cytoscape: a software environment for integrated models of biomolecular interaction networks**. *Genome Res* 2003, **13**(11):2498-2504.
21. Hache H, Lehrach H, Herwig R: **Reverse engineering of gene regulatory networks: a comparative study**. *EURASIP journal on bioinformatics & systems biology* 2009:617281.

ผลงานตีพิมพ์

Bumee S., Liamwirat C., Saithong T., and Meechai A., (2010) Extended Constraint-Based Boolean Analysis: A Computational Method in Genetic Network Inference, Communications in Computer and Information Science, 1, Vol. 115, Computational Systems-Biology and Bioinformatics (CSBio2010), p. 71-82. (ภาคผนวก ง)

ลงชื่อ_____

วันที่_____

ภาคผนวก

ภาคผนวก ก

โค้ดสคริป MATLAB สำหรับคำนวณหา similarity score ในการเปรียบเทียบข้อมูลดิบของการแสดงออกของยีน (raw data) กับข้อมูลที่แปลงค่าเป็น 0 หรือ 1 แล้ว (Discretized data) ซึ่งประกอบด้วย 2 ฟังก์ชั่น

```
function [leaf_set, intern_set] = find_leaf_set(data,dist,method)
% Usage: [leaf_set, intern_set] = find_leaf_set(data,'correlation','average')

Y = pdist(data,dist);
Z = linkage(Y,method);
leaf_amt = size(data,1);
leaf_set = cell(leaf_amt-1,1);
intern_set = cell(leaf_amt-1,1);
for i=1:leaf_amt-1
    if (Z(i,1) > leaf_amt)
        leaf_set[2] = union(leaf_set[2],leaf_set{Z(i,1)-leaf_amt});
        intern_set{i} = union(intern_set{i},Z(i,1)-leaf_amt);
        intern_set{i} = union(intern_set{i},intern_set{Z(i,1)-leaf_amt});
    else
        leaf_set{i} = union(leaf_set{i},Z(i,1));
    end
    if (Z(i,2) > leaf_amt)
        leaf_set{i} = union(leaf_set{i},leaf_set{Z(i,2)-leaf_amt});
        intern_set{i} = union(intern_set{i},Z(i,2)-leaf_amt);
        intern_set{i} = union(intern_set{i},intern_set{Z(i,2)-leaf_amt});
    else
        leaf_set{i} = union(leaf_set{i},Z(i,2));
    end
end
```

```
function [SB, SB_self] = cal_BScore(leaf_set_ref,leaf_set,intern_set)
% Usage: [SB, SB_self] = cal_BScore(leaf_set_ref,leaf_set,intern_set)

SB = zeros(length(leaf_set),1);
SB_self = zeros(length(leaf_set),1);
```

```
  for i=1:length(leaf_set)
     for j=1:length(leaf_set_ref);
        disp(['[' num2str(i) ',' num2str(j) ']']);
        if (isempty(setdiff(leaf_set_ref{j},leaf_set{i})) &&
isempty(setdiff(leaf_set{i},leaf_set_ref{j})))
           disp('yes');
           SB_self(i) = 1/length(leaf_set{i});
        end

        if isempty(intern_set{i}) == 0
           bset = intern_set{i};
           SB(i) = sum(SB_self(bset));
        end
     end
                                          end
```

ภาคผนวก ข

โค้ดสคริป C++ สำหรับสร้าง gene regulatory network ของกลุ่มยีนที่สนใจด้วยวิธี Constraint-based Boolean network

```
//-------------------------------------------------------------

#include <algorithm>
#include <iostream>
#include <iomanip>
#include <fstream>
#include <string>
#include <vector>
#include <map>

#include "../flx/algutl.hpp"
#include "../flx/iosutl.hpp"
#include "../flx/strutl.hpp"

//-------------------------------------------------------------

#define IS_TF( i, tfbeg, tfend ) ( i >= tfbeg && i <= tfend )
#define LINE_PREDICT_NETWORK 5
#define LINE_INFER_CASE      1

//-------------------------------------------------------------

enum regulation
{
  type_none =  0,
  type_gene =  1,
  type_join =  2,
  type_up   =  4,
  type_down =  8,
  type_actv = 16,
  type_inhb = 32,
  type_nspc = 64,
};
```

```
//-------------------------------------------------------------

struct expression
{
  std::vector< int > expr; // gene expression
  std::string        name; // gene name
};

typedef std::vector< expression >     profile;
typedef std::vector< int >            pattern;
typedef std::map< pattern, int >      inferer;
typedef std::pair< inferer, pattern > predict; // [ inferer, conclusion ]
typedef std::vector< predict >        network;

//-------------------------------------------------------------

char regulate_to_char( int val )
{
  return ( val == type_none ? '-' :
         ( val == type_gene ? 'x' :
         ( val == type_join ? 'v' :
         ( val == type_up   ? '1' :
         ( val == type_down ? '0' :
         ( val == type_actv ? 'a' :
         ( val == type_inhb ? 'i' : '-' ) ) ) ) ) ) );
}

//-------------------------------------------------------------

bool load_profile( std::istream & istr,
                   profile & prof )
{
  if( false == istr.good( ) )
    return false;

  // Parse header
  while( false == istr.eof( ) )
```

```
    {
      std::string line;
      if( false == std::getline( istr, line ) )
        continue;

      std::string trim = flx::str_trim( line );
      if( "" == trim )
        continue;

      std::vector< std::string > names = flx::str_split( trim, "," );
        prof.resize( names.size( ) );

      for( size_t i = 0 ; i < names.size( ) ; ++i )
        prof[ i ].name = names[ i ];

      break;
    }

    // Parse expression
    while( false == istr.eof( ) )
    {
      std::string line;
      if( false == std::getline( istr, line ) )
        continue;

      std::string trim = flx::str_trim( line );
      if( "" == trim )
        continue;

      // Split each value in current line
      std::vector< std::string > temp = flx::str_split( trim, "," );

      // Fill each gene expression value
      for( size_t i = 0 ; i < temp.size( ) ; ++i )
        prof[ i ].expr.push_back( flx::val_cast< int >( temp[ i ] ) > 0 ? type_up : type_down
);
    }
```

```
  return true;
}

//-----------------------------------------------------------------

bool save_profile( std::ostream & ostr,
                   const profile & prof )
{
  if( false == ostr.good( ) )
    return false;

  size_t size_prof = prof.size( );
  if( 0 == size_prof )
    return false;

  size_t size_expr = prof[ 0 ].expr.size( );
  if( 0 == size_expr )
    return false;

  // Print header name
  for( size_t i = 0 ; i < size_prof ; ++i )
  {
    ostr << prof[ i ].name;
    for( size_t j = 0 ; j < size_expr ; ++j )
      ostr << ", " << regulate_to_char( prof[ i ].expr[ j ] );
    ostr << std::endl;
  }

  return true;
}

void save_pattern( const pattern & p,
                   std::ostream  & ostr )
{
  pattern::const_iterator
    beg = p.begin( ),
    end = p.end( );
```

```
  for( beg; beg != end; ++beg )
    ostr << regulate_to_char( *beg );
}

void save_inferer( const inferer & f,
                   std::ostream  & ostr )
{
  inferer::const_iterator
    beg = f.begin( ),
    end = f.end( );

  for( beg; beg != end; ++beg )
  {
    save_pattern( beg->first, ostr );

    ostr << " == "
         << regulate_to_char( beg->second )
         << std::endl;
  }
}

void save_network( const network & n,
                   std::ostream  & ostr )
{
  for( size_t i = 0; i < n.size( ); ++i )
  {
    ostr << "Network: " << ( i + 1 ) << std::endl;
      save_inferer( n[ i ].first, ostr );
      save_pattern( n[ i ].second, ostr );
    ostr << " == ?" << std::endl << std::endl;
  }
}

void save_predict( const network & n,
                   std::ostream  & ostr )
{
  ostr << "Result:" << std::endl << std::endl << "    ";
  for( size_t i = 0; i < n.size( ); ++i )
```

```
    ostr << "v" << std::left << std::setw( 2 ) << ( i + 1 ) << " ";
  ostr << std::endl;

  for( size_t i = 0; i < n.size( ); ++i )
  {
    pattern::const_iterator
      beg = n[ i ].second.begin( ),
      end = n[ i ].second.end( );

    ostr << "g" << std::left << std::setw( 2 ) << ( i + 1 ) << " ";
    for( beg; beg != end; ++beg )
      ostr << std::left << std::setw( 4 ) << regulate_to_char( *beg );
    ostr << std::endl;
  }
}

//---------------------------------------------------------------

bool infer_function( const profile & p,
                     const pattern & v,
                          inferer & f )
{
  f.clear( );

  size_t size_prof = p.size( );
  if( 0 == size_prof )
    return false;

  size_t size_expr = p[ 0 ].expr.size( );
  if( 0 == size_expr )
    return false;

  pattern t; t.resize( size_prof, (int)type_none );
  for( size_t i = 0 ; i < size_expr - 1 ; ++i )
  {
    int pval = type_none;

    for( size_t j = 0 ; j < size_prof ; ++j )
```

```
    {
      if( type_gene == v[ j ] )
      {
        pval = p[ j ].expr[ i + 1 ];
        t[ j ] = type_gene;
      }
      else
      if( type_join == v[ j ] )
      {
        t[ j ] = p[ j ].expr[ i ];
      }
      else
      {
        t[ j ] = type_none;
      }
    }

    if( f[ t ] == type_none )
    {
      f[ t ] = pval;
    }
    else
    if( f[ t ] != pval )
    {
      f.clear( );
      return false;
    }
  }

  return true;
}

//-------------------------------------------------------------

void infer_network_helper( const profile & p,
                           const pattern & v,
                           const inferer & f,
                                 pattern & conc )
```

```
{
  inferer::const_iterator
    beg = f.begin( ),
    end = f.end( );

  size_t size_prof = p.size( );
  for( beg; beg != end; ++beg )
  {
    const pattern & pat = beg->first;
    int             res = beg->second;

    for( size_t i = 0; i < size_prof; ++i )
    {
      if( v[ i ] == type_gene )
        conc[ i ] = type_gene;
      else
      if( pat[ i ] == type_up )
      {
        if( conc[ i ] == type_none )
          conc[ i ] = res == type_up ? type_actv : type_inhb;
        else
        if( conc[ i ] == type_actv )
          conc[ i ] = res == type_up ? type_actv : conc[ i ];
        else
        if( conc[ i ] == type_inhb )
          conc[ i ] = res == type_up ? type_actv : conc[ i ];
      }
      else
      if( pat[ i ] == type_down )
      {
        if( conc[ i ] == type_none )
          conc[ i ] = res == type_up ? type_inhb : conc[ i ];
        else
        if( conc[ i ] == type_actv )
          conc[ i ] = res == type_up ? type_nspc : conc[ i ];
      }
    }
  }
```

```
}

//-------------------------------------------------------------------

void infer_network_with_starch( profile & p,
                                network & n,
                                size_t    k,
                                size_t    tf )
{
  size_t size_prof = p.size( );

  k = k > tf ? tf : k;
  for( size_t r = tf; r < size_prof; ++r )
  {
    for( flx::combinator comb( tf, k );
         comb.valid( ); comb.next( ) )
    {
      pattern v;
        v.resize( size_prof, (int)type_none );

      v[ r ] = type_gene;
      for( size_t i = 0 ; i < k ; ++i )
        v[ comb[ i ] ] = type_join;

      inferer f;
      if( true == infer_function( p, v, f ) )
      {
          pattern & conc = n[ r ].second;
          inferer & memo = n[ r ].first;

          infer_network_helper( p, v, f, conc );
          memo.insert( f.begin( ), f.end( ) );
      }
    }
  }
}

//-------------------------------------------------------------------
```

```
void infer_network_no_starch( profile & p,
                              network & n,
                              size_t    k,
                              size_t    tf )
{
  size_t size_prof = p.size( );

  k = k + 1 > tf ? tf : k + 1;
  for( flx::combinator comb( tf, k );
       comb.valid( ); comb.next( ) )
  {
    for( size_t r = 0 ; r < k ; ++r )
    {
      pattern v;
        v.resize( size_prof, (int)type_none );

      v[ comb[ r ] ] = type_gene;
      for( size_t i = 1 ; i < k ; ++i )
        v[ comb[ ( i + r ) % k ] ] = type_join;

      inferer f;
      if( true == infer_function( p, v, f ) )
      {
        pattern & conc = n[ comb[ r ] ].second;
        inferer & memo = n[ comb[ r ] ].first;

        infer_network_helper( p, v, f, conc );
        memo.insert( f.begin( ), f.end( ) );
      }
    }
  }
}

//.....

bool infer_network( profile & p,
                    network & n,
```

```
                size_t    k,
                size_t    tf )
{
  n.clear( );

  size_t size_expr = p[ 0 ].expr.size( );
  if( 0 == size_expr )
    return false;

  size_t size_prof = p.size( );
  if( 0 == size_prof )
    return false;

  n.resize( size_prof );
  for( size_t i = 0; i < size_prof; ++i )
    n[ i ].second.resize( size_prof, (int)type_none );

  if( tf == 0 )
  {
    // no tf, do all combinations
    infer_network_no_starch( p, n, k, size_prof );
  }
  else
  {
    infer_network_with_starch( p, n, k, tf );
    infer_network_no_starch( p, n, k, tf );
  }

  return true;
}

//-----------------------------------------------------------------------------
int proc( int argc, char ** argv )
{
  if( argc != 5 )
  {
    std::cerr << argv[ 0 ]
              << " <k-value> <tf-value> <input> <output>"
```

```
            << std::endl;

  return -1;
}

size_t k = -1;
if( false == flx::val_cast< size_t >( argv[ 1 ], k ) )
{
  std::cerr << "Error: Invalid k-value ("
            << argv[ 1 ]
            << ") !!!"
            << std::endl;

  return -1;
}

size_t tf = -1;
if( false == flx::val_cast< size_t >( argv[ 2 ], tf ) )
{
  std::cerr << "Error: Invalid tf-value ("
            << argv[ 2 ]
            << ") !!!"
            << std::endl;

  return -1;
}

std::ifstream input( argv[ 3 ] );
if( false == input.is_open( ) )
{
  std::cerr << "Error: Open file "
            << argv[ 3 ]
            << " failed !!!"
            << std::endl;

  return -1;
}
```

```
  std::ofstream output( argv[ 4 ] );
  if( false == output.is_open( ) )
  {
    std::cerr << "Error: Create file "
              << argv[ 4 ]
              << " failed !!!"
              << std::endl;

    return -1;
  }

  profile p;
  if( false == load_profile( input, p ) )
  {
    std::cout << "Error: Load profile failed !!!"
              << std::endl;

    return -1;
  }

  network n;
  if( false == infer_network( p, n, k, tf ) )
  {
    std::cout << "Error: Infer network failed !!!"
              << std::endl;

    return -1;
  }
  save_network( n, output );
  save_predict( n, output );
  return 0;
}

int main( int argc, char ** argv )
{
  return proc( argc, argv );
}
//.....
```

ภาคผนวก ค

ยีนที่ใช้ในการสร้างเครือข่ายควบคุมการแสดงออกของยีน (gene regulatory network) ของกระบวนการสังเคราะห์แป้ง ซึ่งแยกเป็นกลุ่มยีนที่ทำหน้าที่เป็นเอนไซม์ที่เกี่ยวข้องกับการสร้างแป้ง (starch genes) และกลุ่มยีนที่ทำหน้าที่เป็นโปรตีนควบคุม (transcription factors) และยีนที่เกี่ยวข้องกับเวลา (clock genes)

ค1: Starch genes

No.	ID	Gene Name	Enzyme
1	At5g51820	PGM1	Phosphoglucomutase
2	At5g24300	STS1	Starch synthase I
3	At3g01180	STS2	Starch synthase II
4	At4g18240	STS4	Starch synthase IV
5	At1g32900	GBS1	Granule-bound starch synthase
6	At2g36390	SBE3	Starch branching enzyme III
7	At2g39930	ISA1	Starch debranching enzyme: Isoamylase I
8	At1g03310	ISA2/DBE1	Starch debranching enzyme: Isoamylase II
9	At4g09020	ISA3	Starch debranching enzyme: Isoamylase III
10	At1g10760	GWD1/SEX1	Glucan water dikinase 1
11	At5g26570	GWD3	Glucan water dikinase-like 3
12	At5g64860	DPE1	Glucanotransferase
13	At2g40840	DPE2	Trasglucosidase
14	At3g29320	PHS1	Glucan phosphorylase (plastidial)
15	At3g46970	PHS2	Glucan phosphorylase (cytosolic)
16	At1g76130	AMY2	α-Amylase2
17	At1g69830	AMY3	α-Amylase3
18	At4g17090	BAM3/BMY8	β-Amylase3
19	At2g32290	BAM6	β-Amylase6
20	At5g18670	BAM9/BMY3	β-Amylase9
21	At5g11720	AGL4	α-Glucosidase-like 4

ค2: Transcription factors and clock genes

No.	Gene ID	family name
1	At1g01060	MYB-related
2	At1g01520	MYB-related
3	At1g02340	bHLH
4	At1g04250	AUX-LAA
5	At1g04990	C3H
6	At1g05690	TAZ
7	At1g05805	bHLH
8	At1g07050	C2C2-CO-like
9	At1g07520	GRAS
10	At1g10200	LIM
11	At1g14510	Alfin
12	At1g17460	MYB-related
13	At1g18570	MYB
14	At1g19700	HB
15	At1g20693	HMG
16	At1g20696	HMG
17	At1g22070	bZIP
18	At1g22190	AP2-EREBP
19	At1g25580	NAC
20	At1g26790	C2C2-Dof
21	At1g28050	C2C2-CO-like
22	At1g29160	C2C2-Dof
23	At1g33240	Trihelix
24	At1g35560	TCP
25	At1g47270	TLP
26	At1g49560	GARP-G2-like
27	At1g50420	GRAS
28	At1g51700	C2C2-Dof
29	At1g51950	AUX-IAA

No.	Gene ID	family name
30	At1g53320	TLP
31	At1g56170	CCAAT-HAP5
32	At1g58110	bZIP
33	At1g69570	C2C2-Dof
34	At1g70000	MYB-related
35	At1g73870	C2C2-CO-like
36	At1g76590	PLATZ
37	At1g77850	ARF
38	At2g02070	C2H2
39	At2g18280	TLP
40	At2g20570	GARP-G2-like
41	At2g22430	HB
42	At2g28200	C2H2
43	At2g28550	AP2-EREBP
44	At2g31070	TCP
45	At2g34720	CCAAT-HAP2
46	At2g35940	HB
47	At2g39900	LIM
48	At2g40140	C3H
49	At2g42400	VOZ
50	At2g43010	BHLH
51	At2g46830	MYB-related
52	At2g47680	C3H
53	At2g47890	C2C2-CO-like
54	At3g02380	C2C2-CO-like
55	At3g02830	C3H
56	At3g06160	ABI3-VP1
57	At3g07650	C2C2-CO-like
58	At3g09600	MYB-related
59	At3g10030	Trihelix
60	At3g17609	bZIP
61	At3g21175	ZIM

No.	Gene ID	family name
62	At3g28910	MYB
63	At3g47500	C2C2-Dof
64	At3g50700	C2H2
65	At3g55770	LIM
66	At3g58680	MBF1
67	At3g59060	BHLH
68	At3g61150	HB
69	At3g61890	HB
70	At4g00050	BHLH
71	At4g16750	AP2-EREBP
72	At4g16780	HB
73	At4g17490	AP2-EREBP
74	At4g18390	TCP
75	At4g25480	AP2-EREBP
76	At4g28610	GARP-G2-like
77	At4g31420	C2H2
78	At4g32280	AUX-IAA
79	At4g34610	HB
80	At4g36730	bZIP
81	At4g39410	WRKY
82	At4g39780	AP2-EREBP
83	At5g02810	C2C2-CO-like
84	At5g02840	MYB-related
85	At5g05090	GARP-G2-like
86	At5g06770	C3H
87	At5g06960	bZIP
88	At5g07690	MYB
89	At5g08520	MYB
90	At5g12840	CCAAT-HAP2
91	At5g15850	C2C2-CO-like
92	At5g17300	MYB-related
93	At5g18680	TLP

No.	Gene ID	family name
94	At5g24470	C2C2-CO-like
95	At5g37020	ARF
96	At5g37260	MYB-related
97	At5g38140	CCAAT-HAP5
98	At5g39760	ZF-HD
99	At5g44190	GARP-G2-like
100	At5g46710	PLATZ
101	At5g48250	C2C2-CO-like
102	At5g51980	C3H
103	At5g56860	C2C2-GATA
104	At5g57660	C2C2-CO-like
105	At5g60100	C2C2-CO-like
106	At5g60850	C2C2-Dof
107	At5g61380	C2C2-CO-like
108	At5g62430	C2C2-Dof
109	At5g63420	Trihelix
110	At5g65310	HB
111	At5g65670	AUX-IAA
112	At5g67030	FHA
113	At5g67480	TAZ

ภาคผนวก ง

Extended Constraint-Based Boolean Analysis: A Computational Method in Genetic Network Inference

Somkid Bumee[1], Chalothorn Liamwirat[2], Treenut Saithong[1,3], and Asawin Meechai[1,4]

[1] Systems Biology and Bioinformatics Research Laboratory, Pilot Plant Development and Training Institute
[2] Division of Biotechnology, School of Bioresources and Technology
[3] Bioinformatics and Systems Biology Program
[4] Department of Chemical Engineering, Faculty of Engineering,
King Mongkut's University of Technology Thonburi, Bangkok, Thailand
somkid@pdti.kmutt.ac.th, chalothorn09@yahoo.com,
treenut.sai@kmutt.ac.th, asawin.mee@kmutt.ac.th

Abstract. Reconstruction of a genetic network, which describes gene regulation of cellular response processes, has been widely studied by using various approaches. Some of which are computational expensive and require enormous efforts. Herein, we proposed an *extended constraint-based Boolean* to infer genetic network. Our method incorporated the specific constraints for a particular system in addition to the general conceptual constraints of a typical genetic circuit, to improve the performance of the existing constraint-based Boolean algorithm. This method was demonstrated in inference of the genetic network underlying circadian rhythms from microarray time series data. The results showed that the proposed method provides good accuracy, specificity, and precision under the trade-off of computational efforts. Moreover, the resulting network showed that prior knowledge is a useful bias for modeling genetic network. The proposed method is therefore a promising alternative approach for inferring genetic network from high-throughput data, such as microarray.

Keywords: Genetic network, extended constraint-based Boolean, conceptual constraints, specific constraints.

1 Introduction

Relationship between gene in a genetic network is important information in understanding the cellular response processes, which involve the regulation of gene expression [1]. The regulation of gene expression lies on a huge number of components that comprise a genetic network, multiple levels of regulation as well as the elaborated interaction between levels [2]. Though the number of network constituents is a barrier of network reconstruction, the (differential) expression of such components is often employed in network inference. This strategy becomes more and more popular once the measurement of thousands of gene components (or whole genome) can simultaneously be performed with the aid of microarray techniques.

J.H. Chan, Y.-S. Ong, and S.-B. Cho (Eds.): CSBio 2010, CCIS 115, pp. 71–82, 2010.

In the last decade, availability of high-throughput technologies, allowing the levels of transcripts to be measured for the whole genome at the same time, enables scientists to understand cellular system by reconstructing genetic network [3, 4]. Various computational approaches have been developed on the purpose of genetic network reconstruction, such as Boolean network [5-7], graphical Gaussian model [8], and Bayesian network [7, 9]. Among these approaches Boolean network and Bayesian network methods are mostly used in the context of reconstructing genetic network from microarray data [10]. These two approaches have distinct advantages and disadvantages. Bayesian network provides a more accurate result yet with a huge requirement of prior data and computational efforts in an iterative learning algorithm, while Boolean network is the simpler method to reconstruct genetic network. Under the trade-off of computational effort, Boolean network is considerably a competitive method to Bayesian network.

Boolean network was originally introduced by Kauffman [5,11]. Later, Shmulevich and Zhang used Boolean network to infer genetic network of cell cycle regulation based on gene expression data [12]. In Boolean network, gene expression is simply considered as binary values, ON or OFF, and the regulation between genes is set by Boolean function. Boolean network was then extended to be Probabilistic Boolean network [13]. This model consists of a family of Boolean networks that combine more than one transition Boolean functions. Inferred network which composes of a set of Boolean functions is selected by using the highest score based on probability. In 2007, Martin and colleagues [6] used Boolean dynamics to infer genetic regulatory network. Possible Boolean networks were generated from microarray time series data. The genetic network was then inferred from selection of possible Boolean networks by using steady-state dynamics. However, the result from Boolean network often includes a number of false positive, resulting in a complex inferred network. To resolve such a problem of Boolean network, recently, our group proposed a constraint-based Boolean network to formulate genetic network by taking prior knowledge into account [14]. The prior knowledge in this work, called the *conceptual constraints*, *i.e.*, enzymatic coding genes do not control and regulate regulatory genes, were included in the filtering process before generating Boolean functions. The result showed the achievement of this model to reduce the complexity of the inferred genetic network by eliminating a certain false prediction.

One of the most studied genetic networks is circadian clock system (*i.e.* a genetic circuit generates about 24h rhythm or circadian rhythm) because it is an important system controlling many biological processes in a wide range of organisms, including plants. Circadian clock in plants has mostly been studied in *Arabidopsis thaliana* [15, 16] in which a certain network components and regulations are revealed. The core circadian clock composes of multiple interlocked feedback loops such as interlock with the timing of cab expression 1 (*TOC1*)/CIRCADIAN AND CLOCK ASSOCIATED1 (*CCA1*)/LATE ELONGATED HYPOCOTYL (*LHY*) loop and (Pseudo-response regulator; *PRR5/PRR7/PRR9*)/*CCA1/LHY* loop. Experimental results show that *CCA1* and LHY are partially redundant genes which are negative regulators of *TOC1* [15]. *CCA1* and *LHY* are also positive regulators of two *TOC1* relatives, *PRR7*, and *PRR9* [16], while *TOC1* acts as a positive regulator of *CCA1* and *LHY*.

The circadian clock network has been used for computational method demonstration in various works [7, 14, 17], mainly due to the appropriate size of the network and (microarray) data availability (http://www.ncbi.nlm.nih.gov/geo/). Needham and colleagues [7] inferred a relationship between circadian-clock genes from an initial set of key genes and iteratively learned to increase network members around genes. A circadian clock was also used as a seed for inferring genetic network by finding co-regulation patterns between gene pairs in circadian clock [8]. The genetic network of *Arabidopsis* genome was performed by using an iterative random sampling strategy.

In this work, we extended the previous constraint-based Boolean analysis [14] to acquire a more accurate network inference by focusing on the filtering process. The specific biological constraints derived from prior knowledge of a particular system under study were introduced to the Boolean network in addition to the conceptual constraints. The algorithm takes a set of genes in a standard input format that is easy in the data preparation process. The extended method was demonstrated in inference of a genetic network underlying circadian rhythms in *A. thaliana* using microarray time series data. Finally, the inferred circadian network was validated with literature [15, 16, 18] and the performance of the extended algorithm was evaluated. The genetic network inferred from our algorithm was compared with that of the constraint-based Boolean approach. The results showed that our algorithm, which considers both conceptual and specific constraints, can increase the accuracy, specificity, and precision of the inferred network. Also the degree of complexity of the inferred network is substantially reduced, resulting more understandable results. The proposed method is therefore a promising alternative approach for inferring larger-scale genetic network from high-throughput microarray time-series data.

2 Methods for Reconstructing Constraint-Based Boolean Network of Circadian Rhythms

The overview of methodology is shown in Fig. 1A. Briefly, expression data was pre-processed before discretization. The binary values from discretization step are the inputs for the extended constraint-based Boolean program to generate Boolean functions and types of regulating genes, i.e., activation and inhibition. The output from the constraint-based Boolean program is Boolean relationship which can be visualized by Cytoscape [19].

2.1 Microarray Data and Data Pre-Processing

Gene expression time series datasets from the Affymetrix microarray under diurnal changes of *Arabidopsis* leaves were downloaded from NCBI database (http://www.ncbi.nlm.nih.gov, experiment reference number is GSE8365) [20]. *Arabidopsis* were grown in light/dark cycles for 7 days and then transferred to constant light. After 24 hours in constant light, 12 samples were harvested at four hours intervals over the next 44 hours for RNA extraction and hybridization on Affymetrix microarrays. The expression data were preprocessed using a package of Bioconductor [21, 22].

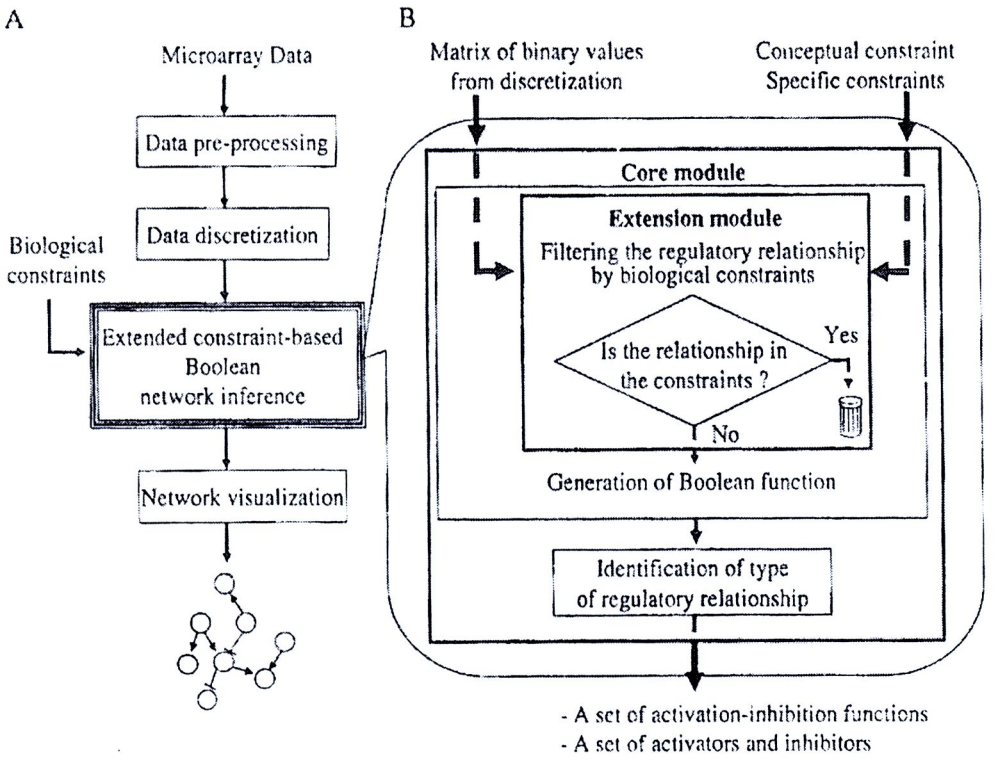


Fig. 1. Overall methodology for genetic network reconstruction by using (A) our constraint-based Boolean algorithm where the extension to the previous method is described in (B)

2.2 Data Discretization

The continuous expression level of gene was discretized into two levels, either '0' or '1', based on the concept of Boolean analysis that, herein, represents the strength of expression, or state, of gene at particular time. In other words, the expression level of gene was converted into either '0' for weak expression or '1' for strong expression. In this work, we used the maximum value of expression level as the simple criteria for discritization that the expression level of gene greater than the determined percentage of the maximum value was discretized into '1', '0' otherwise. The discretized value, $s_{i,t}$, for gene i at time t is defined as Equation (1).

$$s_{i,t} = \begin{cases} 1 & \text{if } x_{i,t} > \text{Max}(G_i) - r \cdot \text{Max}(G_i) \\ 0 & \text{others,} \end{cases} \tag{1}$$

where $x_{i,t}$ is the expression level of gene i at time t, G_i is the set of all expression levels of gene i over the time series, and r is the percentage of the maximum value of expression level of gene i. Here, the expression level greater than 30% of the highest value was converted to 1, *i.e.* r = 0.3. The data matrix of discretized values of all gene expression, *i.e.* $S = [s_{i,t}]$, is called matrix of binary values. It was then passed to extended constraint-based Boolean algorithm to generate Boolean functions

representing the regulatory relationship that is necessary for the construction of the Boolean network.

2.3 Constraint-Based Boolean Network Inference

Our method, called the *extended constraint-based Boolean algorithm* was adapted and implemented based on the previous published works [6, 14]. The algorithm consists of two modules, core and extension modules (Fig. 1B). The core module slightly adapted from the algorithm in Martin et al. [6] includes generation of Boolean function and identification of type of regulatory relationship, i.e. either activation or inhibition. The latter module is filtering the relationship of genes by biological constraints provided by a user. It is the extension we added in order to improve the performance of the classical Boolean algorithm by reduction of the number of relationships considered in the core module. Nevertheless, the network inference can be performed without this module if a set of biological constraints is not submitted.

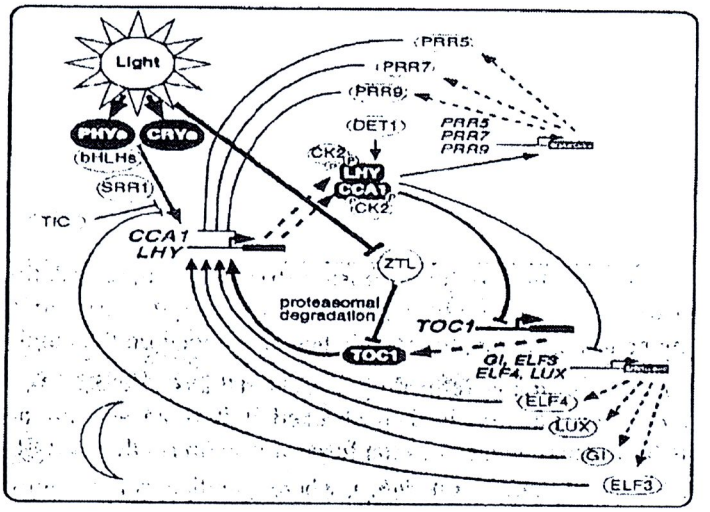
Core module: Generation of Boolean functions and identification of types of regulatory relationship. In brief, at first, the matrix of binary values of all interesting genes is passed into the first module to extract the regulatory relationship between the set of regulating genes and single target gene. The gene expression data, *i.e.* '0' or '1', of the target gene at time t is influenced by the expression strength of the regulating genes at previous time, t-1. This relationship is in the form of Boolean functions having a logic combination of the expression strengths of the regulating genes as an input and the expression strength of target gene as an output. The maximum number of the regulating genes, k, for single target gene is depended on the number of time points of expression data, T, and is defined by Max(k) that $2^k < T$ and $k > 0$.

Each Boolean function is then checked if it is an activation-inhibition function which its logic relationship is in the form of $g_i = (act_1 \text{ OR } act_2 \text{ OR } \ldots \text{ OR } act_{A_i})$ AND NOT $(inh_1 \text{ OR } inh_2 \text{ OR } \ldots \text{ OR } inh_{I_i})$, where g_i indicates the expression strength of the i^{th} target gene, *act* and *inh* indicates the expression strength of activators and inhibitors for the i^{th} target gene respectively, and A_i and I_i are the number of activators and inhibitors for the i^{th} target gene respectively. The activators and inhibitors of each target gene are simultaneously identified in this checking step. The type of regulatory relationship, either activation or inhibition, is finally assigned based on the activation-inhibition function. Other Boolean functions that are not the activation-inhibition function are ignored. Consequently, the output from the algorithm is a set of activation-inhibition functions and a set of activators and inhibitors for each target gene.

Extension module: Filtering the regulatory relationship by biological constraints. In the extension module, the relationship of genes is filtered by a set of biological constraints that are considered as the prior knowledge for a specific system. In this work, two types of biological constraints, *i.e.*, conceptual and specific constraints are added in the program. The conceptual constraint is first added to the previous algorithm [14]. It includes the general concept of the regulation in the transcriptional level, for example, (i) transcription factors directly regulate the gene expression by binding at promoter of genes; and (ii) enzymes and transporters do not regulate the gene expression although some of their downstream products do. Here, this type of

constraints is set based on a presumption that there is not any regulation by products of enzyme-encoding genes within duration of study. The relationship with this type of genes is hence discarded. The specific constraint introduced in this work includes prior knowledge, hypothesis, or existing experimental data indicating the regulatory relationship between the specific set of genes. Here, this set of constraints is specified by pairs of genes having no regulatory relationship which is supported by biological evidences. For example, the relationship between CONSTANTS (*CO*) and phosphoglycerate kinase (PGK) was set as null because the genes have distinct functions in different pathway and there is no experimental data inferring their relationship. The Boolean relationship between these two genes generated was hence

A



B

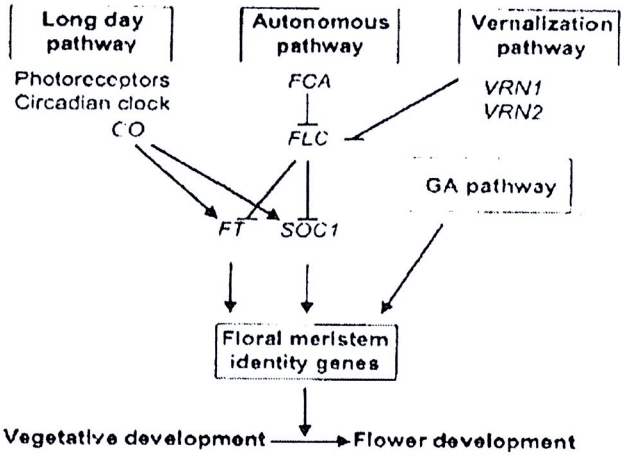


Fig. 2. Gene relationship in (A) circadian clock [18] and (B) flowering pathway [23]

discarded by consideration of that specific constraint. However, the network inference can be performed without this module if a set of biological constraints is not submitted.

2.4 Evaluation of Genetic Network

The obtained genetic network were evaluated by using standard measures that were accuracy (ACC) calculated by (TP+TN)/(TP+TN+FP+FN), specificity (SPC) calculated by TN/(FP+TN), precision or positive prediction value (PPV) calculated by TP/(TP+FP), and false discovery rate (FDR) calculated by FP/(FP+TP), where *TP* refers to correctly inferred edges, either activation or inhibition. *FP* refers to false predicted relationship, including wrong type of regulation. *TN* refers to missing edges in both the inferred and the reference networks. *FN* refers to missing edges, which exist in the reference network, in the inferred network. The network that was used as a reference was based on selected genes in this study [18, 23] (Fig. 2).

Accuracy indicates the percentage of correct predictions. Specificity indicates the percentage of negative predictions which are correctly inferred. Precision indicates the percentage of positive predictions which are correctly inferred. False discovery rate indicates the percentage of false predictions among all predictions.

3 Results and Discussion

3.1 Data Discretization

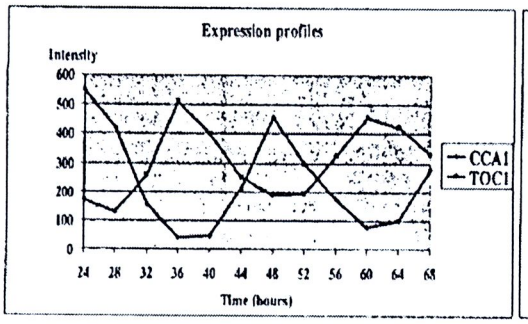
To demonstrate the algorithm, the expression data of fourteen genes were selected from microarray data [20], including seven known core-circadian-clock genes and seven non-circadian genes (*i.e.* genes in glycolysis and flowering pathways). The selected genes and their molecular functions are shown in Table 1.

Table 1. List of genes and functions used in this study [18, 24]

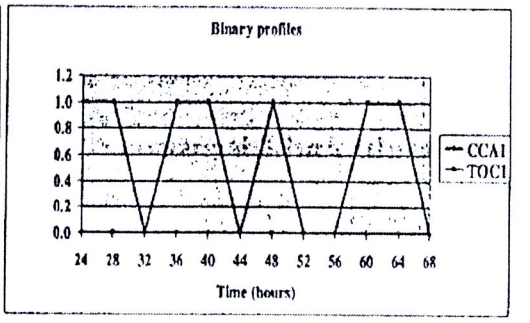
Gene	Function
CCA1	Single Myb domain Transcription factor
ELF4	Transcription factor
GI	Unknown
LHY	Single Myb domain transcription factor
PRR5	Pseudo-response regulator
PRR7	Pseudo-response regulator
TOC1	Pseudo-response regulator
CO	CONSTANTS (CO) promotes flowering under long days
FT	Flowering locus promotes flowering
SOC1	Suppressor of overexpression of CO1
PGI1	Phospho-glucose (Glc) isomerase
TPI	Triosephosphate isomerase
PGK	Phosphoglycerate kinase
PGM	Phosphoglycerate mutase

Max-30%max was used as a threshold for the data discretization in this study. An example of the characteristics of discretized data is shown in Fig. 3. Fig. 3A shows the expression data of two selected circadian clock-genes, *CCA1* and *TOC1*. Based on the discretization method, the expression data of *CCA1* and *TOC1* across time points were discretized into 0 or 1. So, each gene consists of a series of binary values, called binary profile. Fig. 3B shows the binary profiles of such genes. The selection of the discretization method may affect the final result; however the employed discretization method was proven to be the most appropriate one for the system under studied (unpublished data). The matrix of binary values representing binary profiles of a set of genes was an input for the algorithm, see Fig. 3C. The matrix of binary values is delimited text format. The first column is the gene name. The following columns are binary values of each gene across time points. This is a standard format that is convenient for users in the data preparation process.

A



B



C

CCA1	1	0	0	0	0	0	0	0	0	0	0	0
ELF4	0	0	0	0	1	0	0	0	0	0	0	0
GI	0	0	1	1	0	0	0	0	0	0	0	0
LHY	1	0	0	0	0	0	0	0	0	0	0	0
PRR5	0	0	1	1	0	0	0	0	1	1	0	0
PRR7	0	0	1	0	0	0	0	1	1	0	0	0
TOC1	0	0	0	1	0	0	0	0	0	0	0	0
CO	0	0	0	0	0	1	0	0	0	0	0	0
FT	1	0	0	0	0	0	0	0	0	0	0	0
SOC1	0	0	0	0	0	0	0	0	1	1	0	0
PGI1	1	1	1	0	1	0	0	1	1	1	1	1
TPI	0	0	0	1	1	1	0	1	0	0	0	1
PGK	1	1	0	0	1	1	0	0	0	0	1	0
PGM	1	1	1	0	0	0	1	1	0	0	0	0

Fig. 3. Characteristics of data expression profiles (A), example of discretized data (B), and the matrix of binary values (C) of genes in circadian clock, glycolysis, and flowering mechanism

3.2 Genetic Network of Circadian Rhythms

The genetic network of circadian clock was inferred by using our method, the extended constraint-based algorithm with taking consideration of both conceptual and specific constraints into account. The network result by our method was compared with those by the classical Boolean without any biological constraint and the constraint-based Boolean with only conceptual constraints. All these three algorithms can

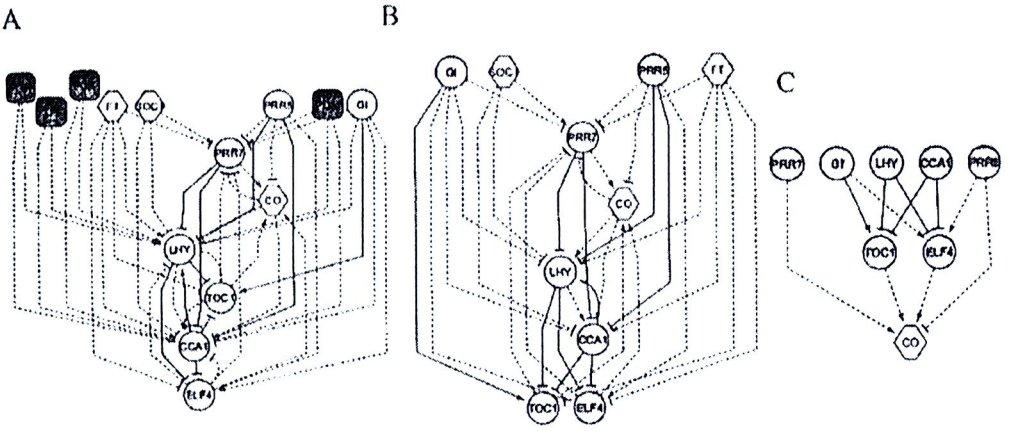


Fig. 4. Genetic networks of 14 genes in circadian rhythm and flowering mechanism using classical Boolean, without consideration of biological constraint *in priori* (A); constraint-based Boolean with conceptual constraints (B); constraint-based Boolean with both conceptual and specific constraints (C). A blue rectangular represents an enzymatic gene; a yellow hexagon represents a flowering gene; a pink circle represents a circadian gene. Black solid lines represent predicted relationship corresponding to known biological knowledge, while broken lines represent predicted relationship that exceeds the current knowledge.

infer directed networks describing the types of gene regulatory relationship, either activation or inhibition (Fig. 4). The node is a gene and the edge is the relationship between genes. The types of the relationships are represented by an arrow and a T-shape arrow for activation and inhibition, respectively.

Figs. 4A-C show the networks inferred by using classical Boolean, constraint-based Boolean with conceptual constraints, and constraint-based Boolean with both conceptual and specific constraints, respectively. All these inferred genetic networks can describe regulations between *TOC1/CCA1/LHY* and *(PRR5/PRR7/PRR9) /CCA1/LHY* loops. All three inferred network show that *CCA1* and *LHY* are inferred as negative regulators of *TOC1* and *ELF4* which corresponds to known biological knowledge [15, 18], while *PRR5* and *PRR7* are inferred as positive regulators of *CCA1* and *LHY* [16] in the inferred network by using classical Boolean and constraint-based Boolean with conceptual constraints. However, among three inferred networks, the two networks from the previously developed methods show significantly higher in complexity and false predictions than the one from our method, indicated by the number of solid and broken line edges. Adding conceptual constraints before generating Boolean function can greatly reduce the complexity of the network (Fig. 4B). That means it can reduce false predicted relationships that are caused by regulations by products of enzyme-encoding genes as shown in Fig. 4A. Fig. 4C shows the inferred genetic network by using our method with adding conceptual and specific constraints before generating Boolean function. The algorithm can identify regulations in the core oscillator of circadian mechanism and also substantially reduce the false predicted relationships. This resulted in less complex inferred network that is reasonable for further analysis and making a biological sense.

3.3 Network Evaluation

The inferred genetic networks were evaluated through a set of coefficients: ACC, SPC, PPV, and FDR. These coefficients allow us to assess the performance of our algorithm in comparison with those of the previously developed methods which are the classical Boolean and the constraint-based Boolean algorithms with conceptual constraints.

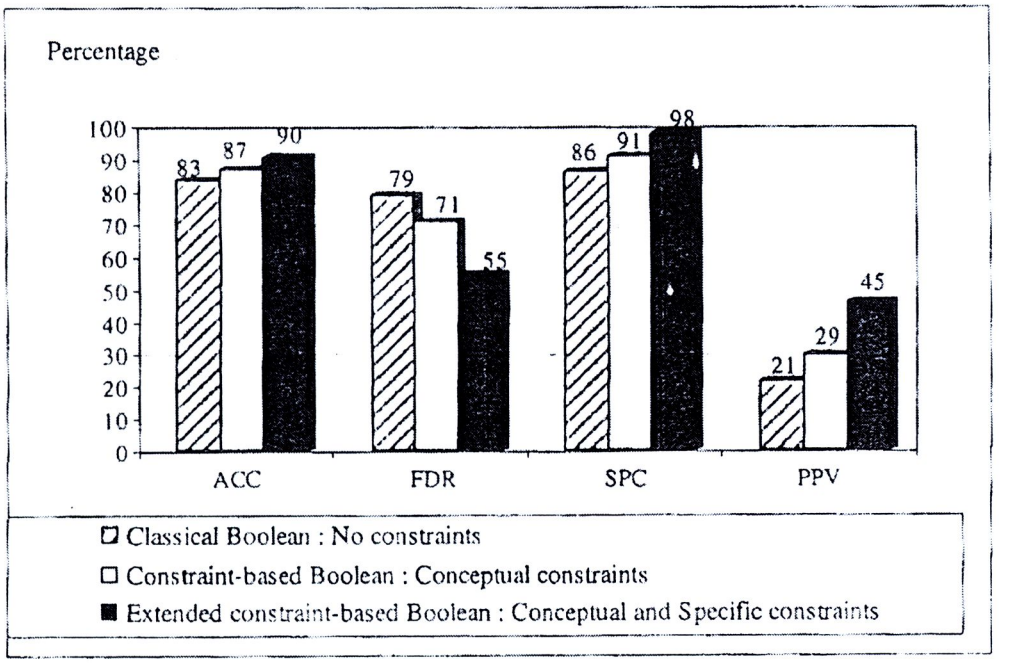


Fig. 5. Comparing the performances of the extended constraint-based Boolean, constraint-based Boolean, and classical Boolean algorithms

Fig. 5 shows that the Boolean network taking into consideration of biological constraints gives better accuracy, specificity, and precision. In comparison with the classical Boolean network, the extended constraint-based Boolean algorithm provides 90% accuracy, 98% specificity, and 45% precision, which are 8%, 13% and 114% improvement, respectively. Moreover, the false discovery rate (FDR) is decreased from 79% to 55% (31% improvement). When considering the Boolean network taking only the conceptual constraints into account, the percent improvement over the classical Boolean network are 4%, 5%, and 38% for accuracy, specificity, and precision, respectively. These results clearly show that taking more consideration of biological constraints in priori can provide better accuracy, specificity, and precision. Besides the improve accuracy, the extended constraint-based Boolean algorithm provides a result with a low level of false prediction. The extension of the constraint-based method by incorporating the specific constraint is thus not only advantage in term of reduction in, but also great decrease in computational burden due to Boolean functions calculation. Not only genetic network inference of circadian clock, the

algorithm was also applied to infer genetic network of galactose pathway using microarray data [25]. The results show that our algorithm provides both high (>70%) accuracy and (>80%) specificity (unpublished data). Therefore, the extended constraint-based Boolean algorithm might be an alternative strategy for genetic network inference. Also, this method might be employed to infer a large-scale genetic network, whose result might be used as seed information for further network analysis or hypothesis development. Although the incorporation of prior knowledge into Boolean network is not yet systematic, this can help scientists to understand simpler genetic network inferred by using this method. However, it will be great to develop it as more systematic approach.

In this work, we have shown the advantages and successes of incorporation prior knowledge (in terms of specific constraint) into the Boolean network though implantation of the constraint in not yet systematic. For the next step, computational technique including systematic incorporation of the constraint will be improved to have the capability of the algorithm to support the large-scale data analysis.

4 Conclusion

The regulation of gene expression lies on a huge number of components that comprise a genetic network. Understanding of this regulation system is often studied by inference of genetic networks from microarray data. We have proposed an algorithm so-called extended constraint-based Boolean algorithm to infer genetic network. The algorithm considers both conceptual constraints of a typical genetic circuit and specific constraints of a particular system before generating Boolean functions. The method was demonstrated in inference of a genetic network underlying circadian rhythms in Arabidopsis thaliana from microarray time series data. The inferred circadian network was validated with literature and the performance of the novel algorithm was evaluated. The resulted network showed that prior knowledge is an useful bias for modeling genetic network. Moreover, the results showed that the proposed method provides good accuracy, specificity, and precision under the trade-off of computational efforts. The proposed method is therefore a promising alternative approach for inferring genetic network from high-throughput microarray time series data. In the future, this method will be applied to infer genetic network from different conditions of microarray data.

References

1. Karlebach, G., Shamir, R.: Modelling and analysis of gene regulatory networks. Nat. Rev. Mol. Cell. Biol. 9, 770–780 (2008)
2. Zhang, S.-Q., Ching, W.-K., Ng, M.K., Akutsu, T.: Simulation study in Probabilistic Boolean Network models for genetic regulatory networks. Int. J. Data Min. Bioinform. 1, 217–240 (2007)
3. Kwon, A.T., Hoos, H.H., Ng, R.: Inference of transcriptional regulation relationships from gene expression data. Bioinformatics 19, 905–912 (2003)
4. Kervestin, S., Amrani, N.: Translational regulation of gene expression. Genome Biol. 5, 359 (2004)

5. Kauffman, S.A.: Metabolic stability and epigenesis in randomly constructed genetic nets. J. Theor. Biol. 22, 437–467 (1969)
6. Martin, S., Zhang, Z., Martino, A., Faulon, J.L.: Boolean dynamics of genetic regulatory networks inferred from microarray time series data. Bioinformatics 23, 866–874 (2007)
7. Needham, C.J., Manfield, I.W., Bulpitt, A.J., Gilmartin, P.M., Westhead, D.R.: From gene expression to gene regulatory networks in Arabidopsis thaliana. BMC Syst. Biol. 3, 1–18 (2009)
8. Ma, S., Gong, Q., Bohnert, H.J.: An Arabidopsis gene network based on the graphical Gaussian model. Genome Res. 17, 1614–1625 (2007)
9. Spellman, P.T., Sherlock, G., Zhang, M.Q., Iyer, V.R., Anders, K., Eisen, M.B., Brown, P.O., Botstein, D., Futcher, B.: Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization. Mol. Biol. Cell 9, 3273–3297 (1998)
10. Li, P., Zhang, C., Perkins, E.J., Gong, P., Deng, Y.: Comparison of probabilistic Boolean network and dynamic Bayesian network approaches for inferring gene regulatory networks. BMC Bioinformatics 8 (suppl. 7), 13 (2007)
11. Kauffman, S.A.: The Origins of Order: Self-Organization and Selection in Evolution. Oxford University Press, New York (1993)
12. Shmulevich, I., Zhang, W.: Binary analysis and optimization-based normalization of gene expression data. Bioinformatics 18, 555–565 (2002)
13. Shmulevich, I., Dougherty, E.R., Kim, S., Zhang, W.: Probabilistic Boolean Networks: a rule-based uncertainty model for gene regulatory networks. Bioinformatics 18, 261–274 (2002)
14. Munkung, W., Liamwirat, C., Bumee, S., Meechai, A.: A constraint-based Boolean approach to inferring genetic circuits. In: The 13th International Annual Symposium on Computational Science and Engineering, pp. 427–431 (2009)
15. Alabadí, D., Oyama, T., Yanovsky, M.J., Harmon, F.G., Más, P., Kay, S.A.: Reciprocal regulation between TOC1 and LHY/CCA1 within the Arabidopsis circadian clock. Science 293, 880–883 (2001)
16. Farre, E.M., Harmer, S.L., Harmon, F.G., Yanovsky, M.J., Kay, S.A.: Overlapping and distinct roles of PRR7 and PRR9 in the Arabidopsis circadian clock. Curr. Biol. 15, 47–54 (2005)
17. Du, P., Gong, J., Syrkin Wurtele, E., Dickerson, J.A.: Modeling gene expression networks using fuzzy logic. IEEE Trans. Syst. Man Cybern. B Cybern. 35, 1351–1359 (2005)
18. McClung, C.R.: Plant circadian rhythms. Plant Cell 18, 792–803 (2006)
19. Shannon, P., Markiel, A., Ozier, O., Baliga, N.S., Wang, J.T., Ramage, D., Amin, N., Schwikowski, B., Ideker, T.: Cytoscape: a software environment for integrated models of biomolecular interaction networks. Genome Res. 13, 2498–2504 (2003)
20. Covington, M.F., Harmer, S.L.: The circadian clock regulates auxin signaling and responses in Arabidopsis. PLoS Biol. 5, e222 (2007)
21. Workman, C., Jensen, L.J., Jarmer, H., Berka, R., Gautier, L., Nielser, H.B., Saxild, H.H., Nielsen, C., Brunak, S., Knudsen, S.: A new non-linear normalization method for reducing variability in DNA microarray experiments. Genome Biol. 3, research0048 (2002)
22. Li, C., Wong, W.H.: Model-Based Analysis of Oligonucleotide Arrays: Expression Index Computation and Outlier Detection. In: Conference Model-Based Analysis of Oligonucleotide Arrays: Expression Index Computation and Outlier Detection, pp. 31–36 (2001)
23. Blazquez, M., Koornneef, M., Putterill, J.: Flowering on time: genes that regulate the floral transition. EMBO reports 2, 1078–1082 (2001)
24. The Arabidopsis Information Resource, http://www.arabidopsis.org/
25. DeRisi, J.L., Iyer, V.R., Brown, P.O.: Exploring the metabolic and genetic control of gene expression on a genomic scale. Science 278, 680–686 (1997)