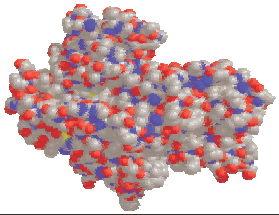
Prof. Yves Deville
Computing & Engineering Department
Université catholique de Louvain

CLP(BioNet) : Towards a CLP framework for the analysis of Biochemical Networks

Yves Deville

SweConsNet 2004



Linköping 15 January 2004

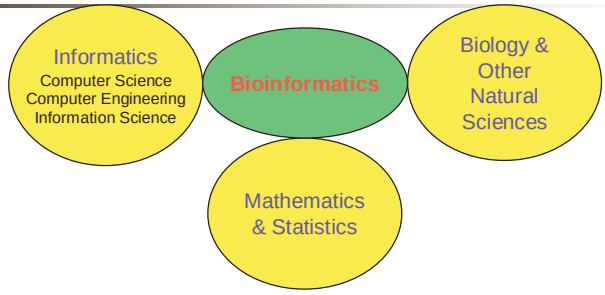
Overview

- Bioinformatics
- Biological networks
- CLP(BioNet)
  - Constraints for bio.net analysis
  - Constraints for bio.net matching
- Perspectives

What is Bioinformatics ?

- An intersection of AI and genetics
  - Two very popular (most wanted) sciences
- An opportunity
  - To use some of the most interesting computational techniques to solve some of the most important and rewarding questions
- Where Frankenstein meets the Terminator

What is Bioinformatics?

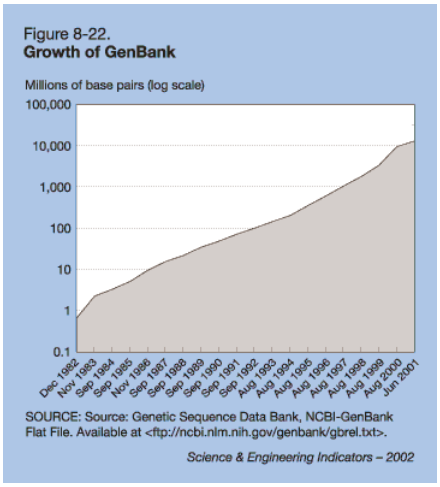


- Bioinformatics : the study of the application of
  - molecular biology, computer science, artificial intelligence, statistics and mathematics
  - to model, organize, understand and discover interesting information associated with the large scale molecular biology databases
  - to guide essays for biological experiments

Why is bioinformatics Important ?

- Genome sequencing, microarrays, …
  lead to large amounts of data to be analyzed
- Leads to important discoveries
- SmartMoney ranks Bioinformatics as #1 among next HotJobs
- Exciting research potential

Database Growth



Challenges in Bioinformatics

- Many **NP-hard problems**: multiple alignment, distant homology, motif finding, protein folding, phylogeny, gene relationship in expression data, mining and learning, …
- From whole genome to functioning system of a biological organism
- Predicting interactions between genes and molecules

Can CP be helpful in some of these challenges ?

Topics in Bioinformatics

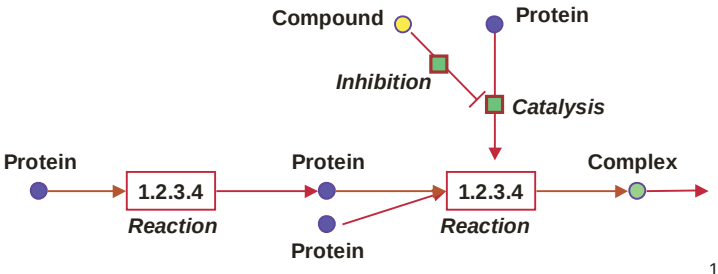
- Sequencing genome
- Sequence alignment
- Searching databases
- Machine learning
- Hidden Markov Model
- Phylogenetic trees
- Functional genomics
- Simulation
- Structure prediction
- Microarrays (DNA chips)
- Biochemical databases
- ***Biochemical network analysis***
- Ethical, legal & social issues
- …

Overview

- Bioinformatics
- Biological networks
- CLP(BioNet)
  - Constraints for bio.net analysis
  - Constraints for bio.net matching
  - Experimental results
- Perspectives

Biological Networks

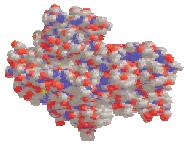
- Networks of **interactions** between **biological entities** within the cell
- Bioentities and Interactions observed in experiments
- Stored in databases



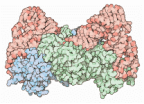
Bioentities

- Gene (part of the DNA)

- Polypeptide (Protein)



- Complex (formed by several polypeptides)



- Compound (ATP, ADP, Water,Proline, …)

Interactions

- An interaction may be a **transformation** of bioentities into other bioentities
  - Reaction : chemical reaction occurring within the cell
  - Expression : Gene $\rightarrow$ polypeptide
  - Assembly : polypeptide forming a complex
- An interaction may be a **control** of a transformation
  - Catalysis of a reaction by some enzyme (protein)
  - Regulation of the expression of a gene

Biochemical networks

One usually distinguish different types of networks :

- **Metabolic** network
  - Series of reactions, possible controlled by enzymes, leading to some specific product
- **Regulatory** network
  - Focus on the regulation of the enzyme activity, or on the stimulation of the enzyme expression
- **Signal transduction** network
  - Transport of information (from membrane to gene)

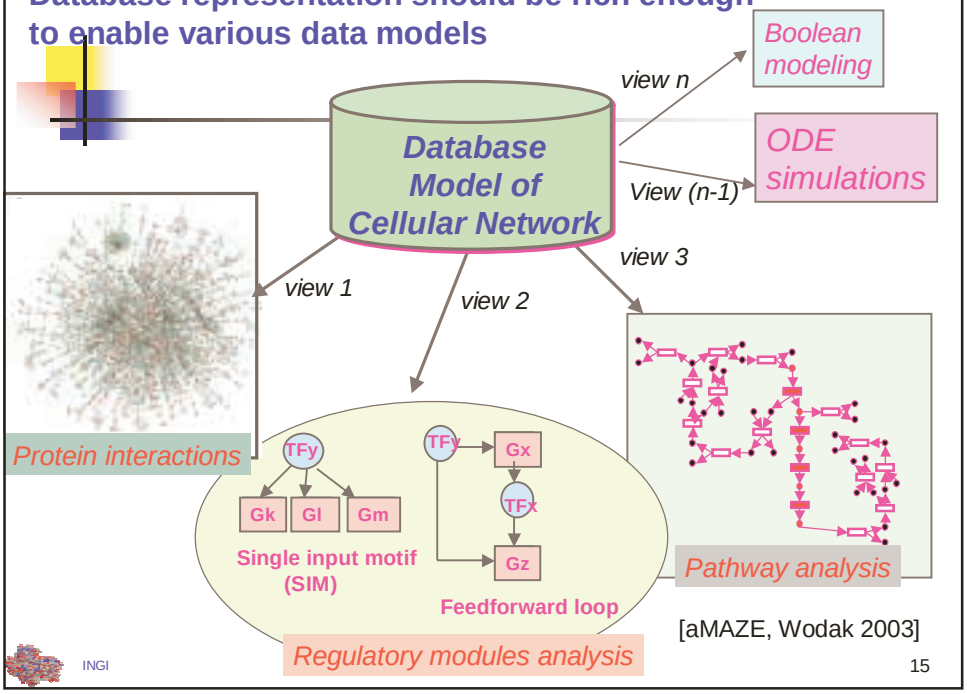
These networks are usually represented using
different models**, and stored in **different databases

The aMAZE project

- **Database for biological networks**
- Prof. Shoshana Wodak, Molecular Biology & Bioinformatics, ULB, Belgium
- Based on a rich Object Oriented model:
- Integration of different types of networks
  - metabolism
  - regulation
  - signal transduction, etc
- Extendable model

www.amaze.ulb.ac.be

Database representation should be rich enough to enable various data models



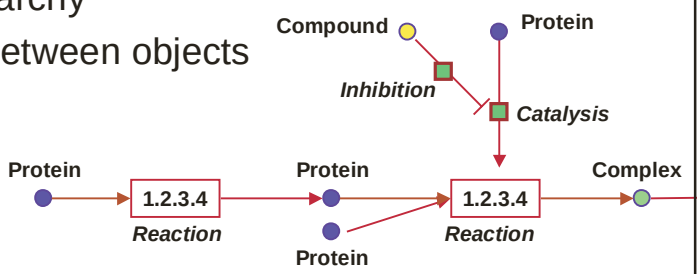
[aMAZE, Wodak 2003]

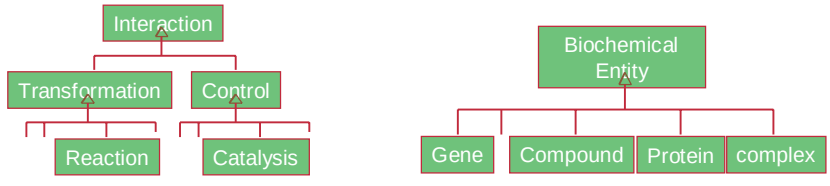
INGI

15

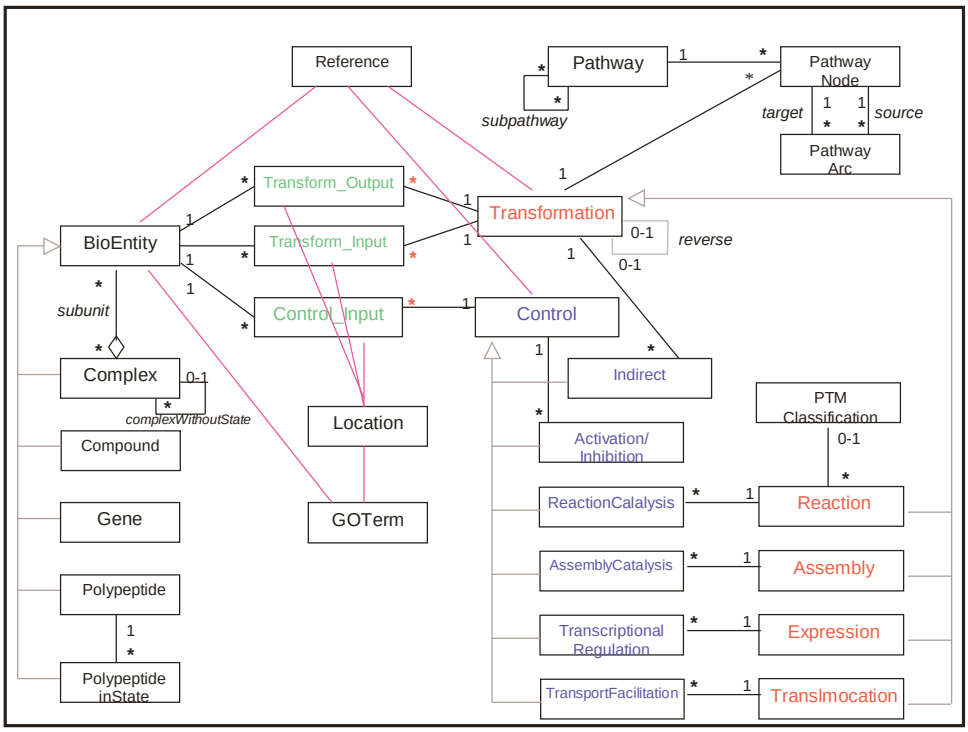
Bio.Net as Object Graphs

- Bioentities and interactions are objects
- Object hierarchy
- Relations between objects



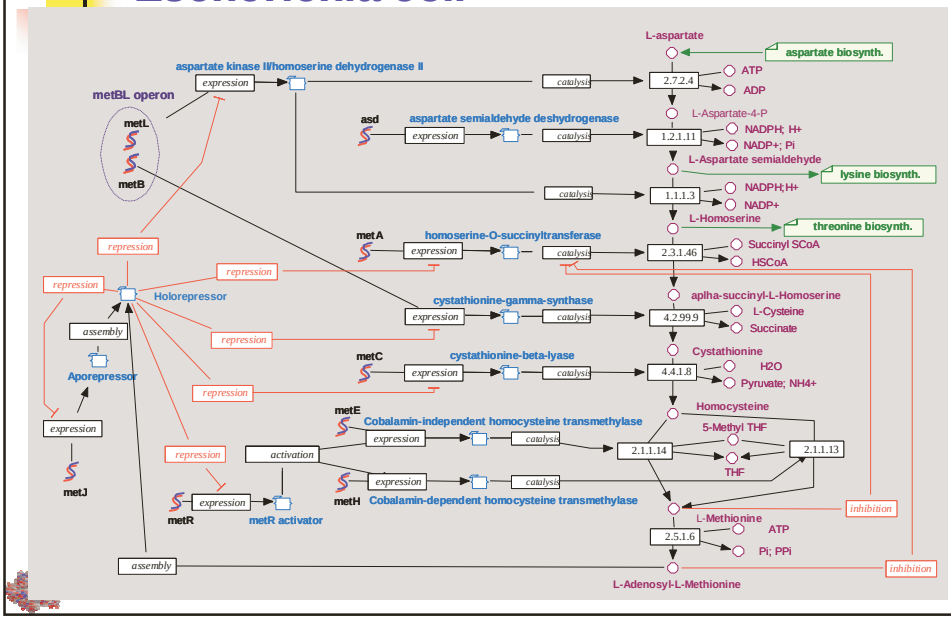


INGI



Methionine Biosynthesis in Escherichia coli

[aMAZE, van Helden 2003]



The BioMaze Project

- **Analysis & Visualisation of biochemical networks**
- Closely related to the aMAZE project
- Interdisciplinary and interuniversity project
  - Y. Deville, CS, UCL, Louvain-la-Neuve
  - S. Wodak, Bio, ULB, Brussels
  - J.L. Hainaut, CS, FUNDP, Namur
  - E. Zimany, CS, ULB. Brussels
- Funded by the Walloon Region

The BioMaze-UCL Project

- **Analysis of biochemical networks**
  - Application of CP and AI techniques
- People
  - Y. Deville
  - P. Dupont
  - G. Dooms
  - S. Zampelli

Why analysing networks ?

Examples of biological questions

- Give all pathways traversing a set of specified compounds or reactions (e.g. given a set of co-regulated genes, find a pathway that could be formed with the catalyzed reactions).
- Find all genes whose expression is directly or indirectly affected by a given compound.
- Show which paths or pathways may be affected when one or more gene/proteins are turned off or missing.
- Compare biochemical pathways from different organisms and tissues, or at different stages of annotation; highlight common features and differences; predict missing elements.

Examples of analysis

- Structure of the network
- Path finding
- Distance between paths
- Pathway synthesis
- Pathway prediction
- Patterns discovery
- Functionally related enzyme clusters
- …

Existing Approaches

- Most analysis use a simplistic model of biochemical networks
  - E.g. compound graphs where nodes are bioentities and arcs are the reactions
  - Many useful analysis are meaningless in such models
- Most analysis use specialized graph algorithms
- Analysis cannot easily be combined

Overview

- Bioinformatics
- Biological networks
- CLP(BioNet)
  - Constraints for bio.net analysis
  - Constraints for bio.net matching
- Perspectives

Why CP ?

Why using CP for the analysis of biochemical networks ?

- Existence of efficient ad hoc graph algorithm for specific analysis
- Difficult to combine such algorithms
- Difficult to extend these algorithms to extended analysis
- New analysis usually need lot of programming effort
- Richness of the underlying model can be exploited through constraints

Objectives of CLP(BioNet)

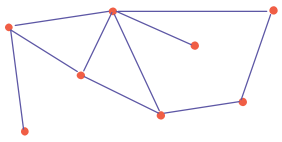
- Introduction of graph domain variable
  - Values are biochemical networks
  - Domains are sets of biochemical networks
  - Exploiting the nature of graph
- Definition of constraints
  - Dealing with graph domain variables
  - Useful for the analysis of biochemical networks
  - Extendable framework

Related Work in CP

- Set constraints (e.g. [Gervet, 1997])
  - Implementation techniques
  - Propagation techniques
- Global constraints (e.g. [Beldiceanu, 2000])
  - Graph algorithms
  - Implementation techniques

Graph

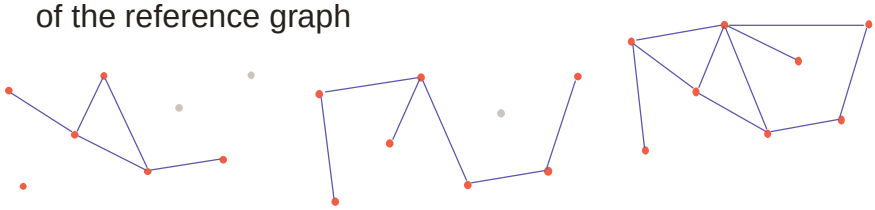
- A bio.net is represented by a graph
- A graph *g=(N,A)* is defined by
  - *N* : set of nodes
  - *A* : set of arcs ($A \subseteq N \times N$)



For simplicity of presentation, no types for the nodes

Graph domain variable

- A **graph domain variable** is declared with an initial domain, called the **reference graph** of *G*
- The (initial) domain a gd-variable *G* is the set of all subsets of the reference graph



- We assume here that the gd-variables have the same reference graph
  - Analysis of a single biological network

Representation of a gd-variable

- A graph domain variable *G* is represented by
  - its reference graph *g* = (*Nref*,*Aref*)
  - a finite set domain variable *N* over *Nref*
  - a finite set domain variable *A* over *Aref*
  - The constraint $A \subseteq N \times N$
- We denote
  - *N* = *node(G)*
  - *A* = *arc(G)*

Implementation

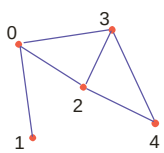
Choice of a constraint programming environemnt

- Choice of **Oz**
- Facility to develop new propagators
- Local expertise at UCL
  - Peter Van Roy and his research team

Implementation of gd-variable

- We do not explicitly use existing finite set variables; not suitable for implementation of specific graph propagators
- Reference graph *g*=(*Nref*,*Aref*)
  - *Nref* : nodes labeled from 0 to *n* -1
  - *Aref* : represented as an adjacency matrix (n^2 Boolean value)

A more elaborated representation could be used

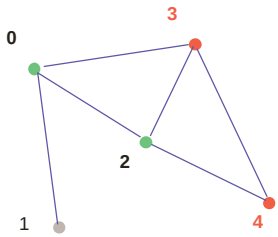


Aref

	0	1	2	3	4
0	0	1	1	1	0
1	1	0	0	0	0
2	1	0	0	1	1
3	1	0	1	0	1
4	0	0	1	1	0

Implementation of gd-variable

- **node(G)** : fs-domain variable over *Nref*
  - Vector of *n* Boolean domain variables
  - State the presence/absence of the node in *G*
  - Denoted *nodeBV(G)*

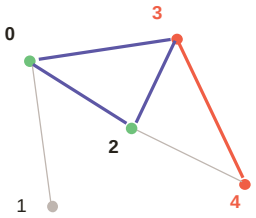


nodeBV(G)

0	1	2	3	4
0 -1	**0**	0 -1	1	1

Implementation of gd-variable

- **arc(G)** : fs-domain variable over *Aref*
  - Adjacency matrix of n^2 Boolean domain variables
  - State the presence/absence of the arcs in *G*
  - Denoted *arcBV(G)*
  - $arcBV(G)_{ij} = 0$ *when* $Aref_{ij} = 0$



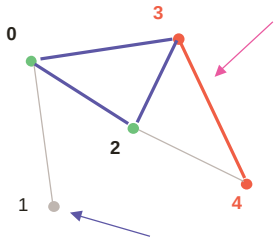
arcBV(G)

	0	1	2	3	4
0	0	**0**	0-1	0-1	0
1		0	0	0	0
2			0	0-1	0-1
3				0	1
4					0

Implementation of gd-variable

Internal constraint : *arc(G) $\subseteq$ node(G) x node(G)*

- Represented by n^2 propagators
  $arcBV(G)_{ij} \Rightarrow nodeBV(G)_i \wedge nodeBV(G)_j$



nodeBV(G)

0	1	2	3	4
0 -1	**0**	0 -1	1	1

arcBV(G)

	0	1	2	3	4
0	0	**0**	0-1	0-1	0
1		0	0	0	0
2			0	0-1	0-1
3				0	1
4					0

Constraints on gd-variables

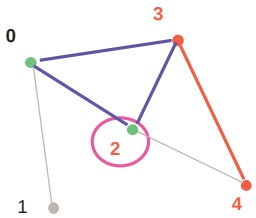
- *NodeInGraph(n,G)*
- *ArcInGraph(a,G)*
- *SubGraph(S,G)*
- *Path(P,ns,ne,max)*
- *EveryArc(G)*
- *ExistPath(ns,ne,max,G)*
- *Connex(G)*

NodeInGraph(n,G)

- *G* : gd-variable
- Constraint : $n \in node(G)$

Implementation

- $nodeBV(G)_n$



NodeInGraph(2)

nodeBV(G)

0	1	2	3	4
0 -1	**0**	1	1	1

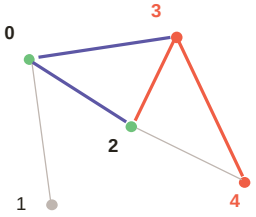
Basic constraints that can be negated

ArcInGraph(a,G)

- *G* : gd-variable
- Constraint : $a \in arc(G)$

Implementation

- $arcBV(G)_{ij}$ with $a = (i,j)$



ArcInGraph((2,3))

arcBV(G)

	0	1	2	3	4
0	0	**0**	0-1	0-1	0
1		0	0	0	0
2			0	1	0-1
3				0	1
4					0

Basic constraints that can be negated

SubGraph(S,G)

- *S, G* : gd-variables
- Constraint : *S* is a subgraph of *G*
  $node(S) \subseteq node(G)$ and $arc(S) \subseteq arc(G)$

Implementation

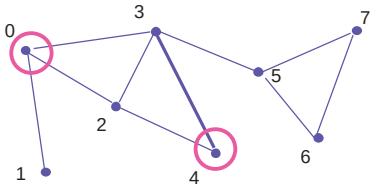
- $n^2 + n$ propagators
  $arcBV(S)_{ij} \Rightarrow arcBV(G)_{ij}$
  $nodeBV(G)_i \Rightarrow nodeBV(G)_j$

Path(P,ns,ne,max)

- *P* : gd-variable
- *ns, ne* ∈ *Nref*
- *max* : integer
- Constraint : *P* is a path from *ns* to *ne*, length ≤ *max*

$ns=n_0 \wedge ne=n_k \wedge node(P) = \{n_0,\ldots, n_k\} \wedge k \leq max$

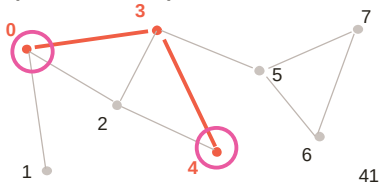
$\wedge\ arc(P) = \{ (n_i, n_{i+1}) \mid 0 \leq i < j \}$



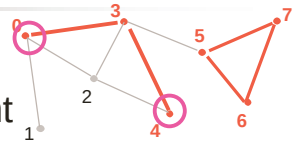
Path(P,0,4,3)

Path implementation

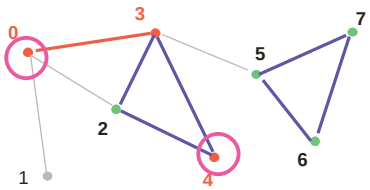
- *ns* and *ne* must be in the path *P*
  NodeInGraph(ns,P) $\wedge$ *NodeInGraph(ne,P)*
- Degree of the nodes in P (i.e. number of neighbors)
  - degree(ne)=degree(ns)=1
  - Other nodes : degree(n)=2
- Implemented by *n* propagators
  $nodeBV(P)_i \Leftrightarrow \Sigma\, arcBV(P)_{ij} = 2 \quad (ne \neq i \neq ns)$
  $\Sigma\, arcBV(P)_{ns\,j} = 1$
  $\Sigma\, arcBV(P)_{ne\,j} = 1$

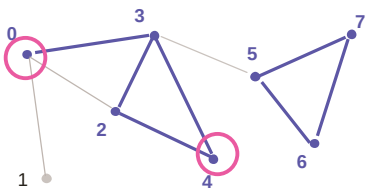


Path implementation



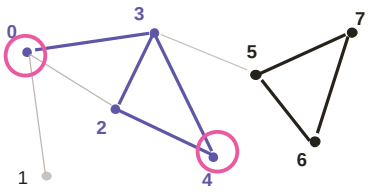
- nodes in *P* must be constrained
  to be a single connected component
- Implemented by a stateful propagator
- Data structure *ConGraph*
  - lub of the possible graphs in the **current domain** of *P*
    (i.e. drop the nodes and arcs *not* in G)





ConGraph

- If the nodes of *P* with $nodeBV(P)_i$ =1 (e.g. *ns*, *ne*) are not in the **same connected component** of *ConGraph*
  Then failure
  Else prune the nodes and arc in the other components



- Propagator reawaken when *arc(P)* is reduced

Connected components

- Search of connected components
  - Standard breadth-first, depth-limited (*max*) search, starting from *ns*
  - Limited to the main connected component (i.e. containing *ns*)
- Exploiting the connected component
  - Check whether the connected component is a tree (no cycle)
  - In that case, assign the unique path to *P*

Complexity of Path constraint

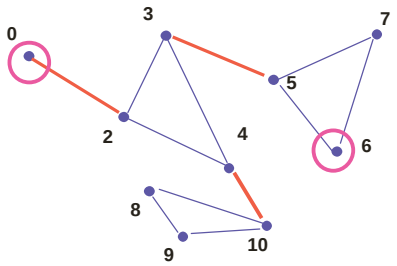
- Sum constraints
  - At most |*Nref* | constraints
  - Amortized complexity of one constraints $O(|Aref|)$
  - Hence complexity $O(|Nref|.|Aref|)$
- Connected components
  - Construction of *ConGraph* : $O(|Aref|)$
  - Search of the connected components : $O(|Aref|)$
  - Constraint executed when an arc is removed from *P* : $O(|Aref|)$
  - Hence complexity $O(|Aref|^2)$
- Hence a global complexity of $O(|Aref|^2)$ for the constraint

Enhancement (complexity)

- Dynamic graph connectivity algorithms [Holn & al., 1998]
  - $O(|Aref| .\log((|Nref|)^2)$
  - Find also the edge-connectivity of the graph
  - Complex implementation

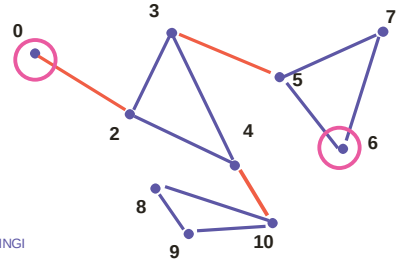
Enhancement (pruning)

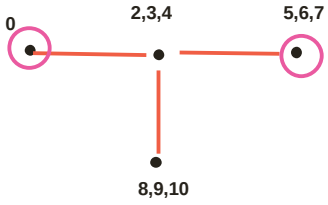
- In the main connected component (containing *ne* + *ns*) of *ConGraph* determine
  - 2-edge-connected components (*an arc can be removed without loosing the connected property*)
  - Bridges between these components



Enhancement (pruning)

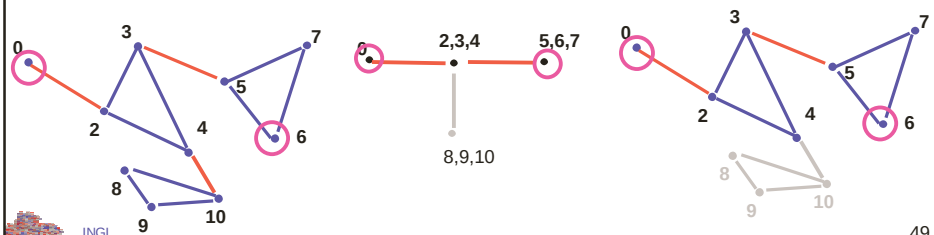
- From *ConGraph* and its main connected component (containing *ne* + *ns*), construct a tree
  - 2-edge-connected components are the nodes
  - Bridges are the arcs





Enhancement pruning

- Determine the (unique) path from the node containing *ns* to the node containing *ne*
- All the arcs in this path must be in *arc(P)*
- All the arcs and nodes not in this path can be pruned from G



Experimental results

Objectives

- Preliminary results showing the feasibility of the approach

Overview

- The experimental data
- Path finding
- Combined constraints

The experimental data

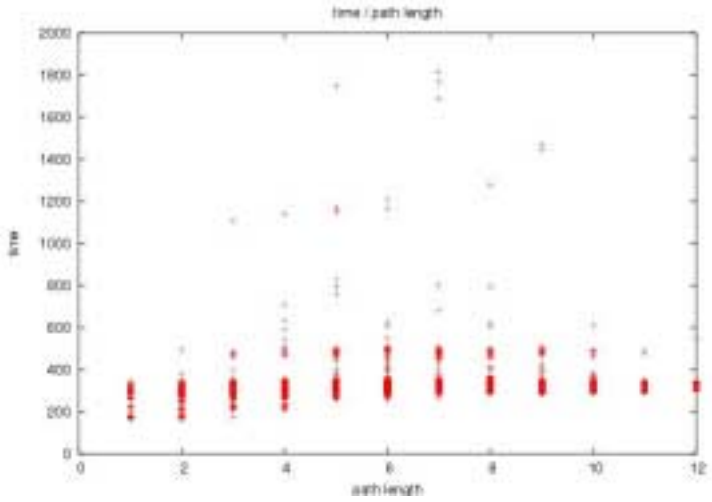
- Graph extracted from aMAZE database
- Biochemical networks mostly Escherichia coli
- 9.773 Edges, 21.755 arcs
- Average arity : 4.45

Analysis is done on a part of such a graph

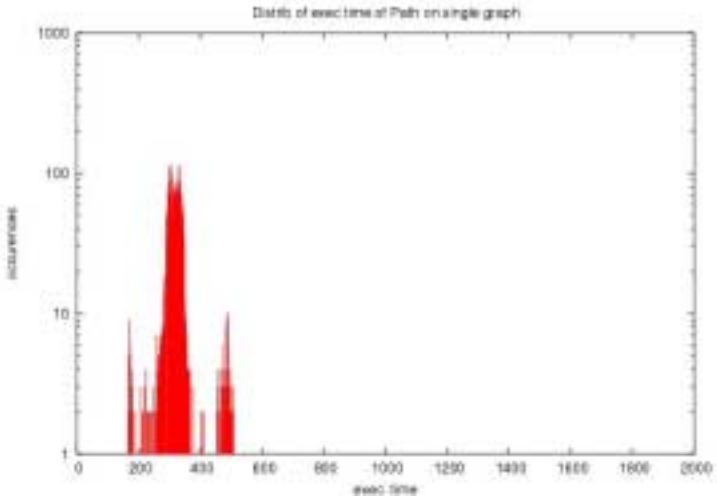
Experiment 1

- Graph with 100 nodes in a single connected component, average degree 3.1
- Search of a path for each pair of nodes
  4.950 paths
  - Path length Average: 7.3 Std dev.: 2.3
  - Num of vars : 15 148.
  - Num of propag Average: 20 519 Std dev.: 325
  - Num of invoked propag
    Average: 75 273 Std dev.: 7 402
  - Run time Average: 324ms Std dev.: 73.7

Experiment 1



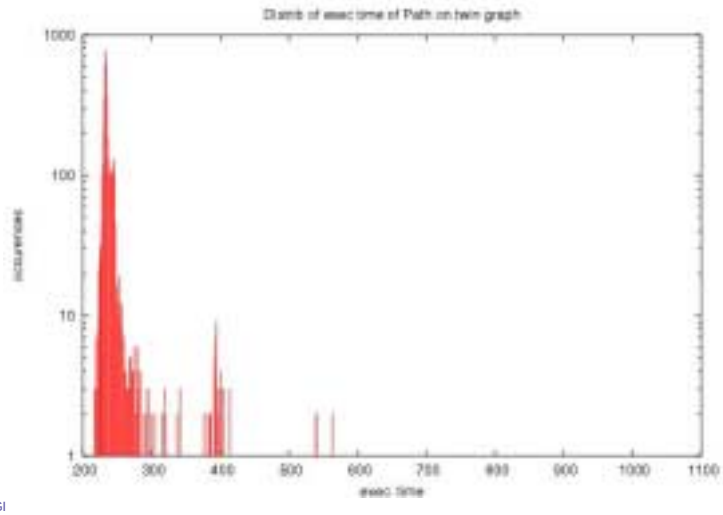
Experiment 1



Experiment 2

- Graph with 200 nodes in two single connected components, average degree 3.1
- Search of a path for each pair of nodes from the two components (5 000 failures)
  - Path length Average: 0
  - Num of vars : 60 298
  - Num of propag : 80 400
  - Num of invoked prop
    Average: 257 628 Std dev.: 8 158
  - Run time Average: 240ms Std dev.: 38

Experiment 2



Experiment 3

- Graph with 100 nodes in a single connected component, average degree 3.1
- Search of a path between two specific nodes, with two constrained intermediate nodes (100 pairs of intermediate nodes)
  - Path length Average: 5.88 Std dev.: 7.9
  - Num of vars Average: 60193 Std dev.: 73
  - Num of propag Average: 80 924 Std dev.: 3139
  - Num of invoked prop
    Average: 378297 Std dev.: 555663
  - Run time Average: 2261 ms

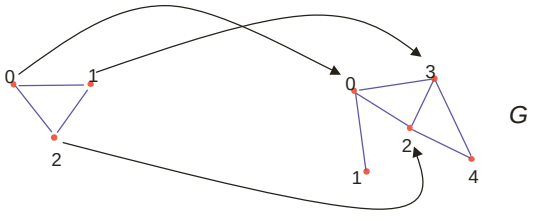
Overview

- Bioinformatics
- Biological networks
- CLP(BioNet)
  - Constraints for bio.net analysis
  - **Constraints for bio.net matching**
- Perspectives

Subgraph isomorphism

- *Gp=(Np,Ap)* : **pattern graph**
- *G=(N,A)* with $|Np| \leq |N|$
- Find a function $f : Np \rightarrow N$ such that
  - *f* is injective
  - $\forall n1, n2 : (n1,n2) \in Ap \Rightarrow (f(n1),f(n2)) \in A$

Pattern graph *Gp*



- Subgraph isomorphism is NP-complete

Family of problems

- Graph vs subgraph isomorphism
- Exact vs inexact matching
  - Nodes with attributes, matching between two node is a *distance*
  - Some arcs from the pattern graph are not considered. Distance from the initial pattern and the chosen subpattern

Analysis of Biochemical Networks

- Subgraph isomorphism is a basic operation for graph pattern matching

Applications

- Compare biochemical networks
  - from different organisms and tissues
  - at different stages of annotation;
  - highlight common features and differences;
  - predict missing elements ('reconstruction')
- Compile repertoires of recurrent network motifs (topological patterns) at different resolution levels

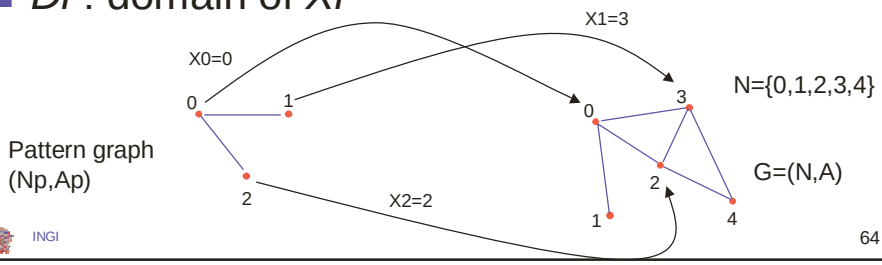
Existing algorithms

Many existing algorithms for standard subgraph isomorphism based on various techniques

- Cliques
- Fuzzy set theory
- Elastic graph matching
- Multiple graph matching
- Error correction
- Genetic algorithms
- Decision tree
- Neural networks
- Clustering
- Connected components
- **Constraint programming** [Rudolf, 1998] [Valiente, 2000]
- …

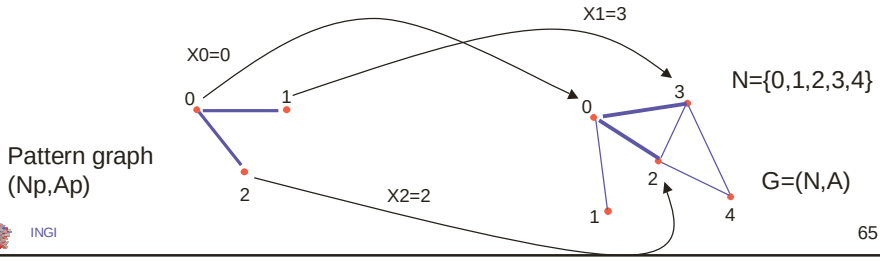
SI as a CSP

- *Gp=(Np,Ap)* : **pattern graph**
- *G=(N,A)* with $n = |Np| \leq |N| = d$
- We use *Np=(X1,…,Xn)*
- **Domain variables** $Xi \in N$
- *Di* : domain of *Xi*



Subgraph Isomorphism as a CSP

- Let *i* and *j* be distinct nodes from pattern graph
- C1 : $Xi \neq Xj$ for all $i \neq j$
- C2 : $(Xi,Xj) \in A$ for all $(i,j) \in Ap$
- [Rudolf, 1998], [Valiente, 2000]
- How to achieve pruning ?

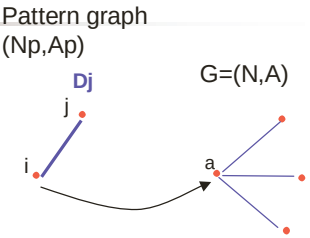


C1 : Pruning

C1 : $Xi \neq Xj$ for all $i \neq j$

- allDiff(X1,..,Xn) constraint
- More pruning than arc consistency on binary constraints
- $O(n^2d^2)$ with $n = |Np|$ and $d = |N|$
- [Regin, 1994]

C2 : Pruning



C2 : *(Xi,Xj) ∈ A* for *(i,j) ∈ Ap*

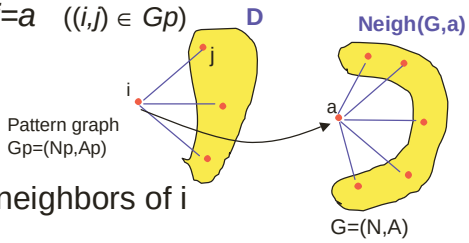
- When can we prune value *a ∈ Di* ?
- If there is no value *b ∈ Dj* s.t. *(a,b) ∈ A*
- Classical arc consistency

Implementation

- Constraint can be reexpressed as
  C2 : *Dj ∩ neigh(G,a) = ∅*
- Independent from *i*; same pruning criteria for all neighbors of j in Ap
- *S(j,a)* = | *Dj ∩ neigh(G,a)* |
- *S(j,a)* : # neigbhors of *a* in *Dj*
- When *S(j,a)=0*, prune *a* from all neighbors of *j* in pattern graph
- **Space complexity** : *O(nd)*
- **Time complexity** : $O(nd^2)$ (instead of $O(n^2d^2)$ for classical AC algorithm)

C3 : Redundant constraint

- Candidate values for *Xj* if *Xi=a* (*(i,j) ∈ Gp*)
  (*Dj ∩ neigh(G,a)*)



- Let $D = \bigcup_{j \in neigh(Gp,i)} Dj$
- Candidate values for all the neighbors of i

  $D \cap neigh(G,a)$
- There must be more candidates than neighbors :

  C3 : $| D \cap neigh(G,a) | \geq | neigh(Gp,i) |$
- **We can prune value *a ∈ Di* if there are more neighbors than candidate values for these neighbors**
- [Valiente, 2000]

C3 : Redundant constraint

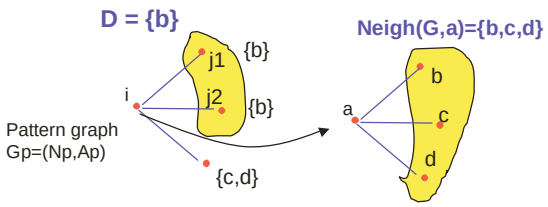
- Can be implemented with the following data structure
- $R(i,b) = |\{ j \in neigh(Gp,i) \text{ s.t. } b \in Dj \}|$
- $R(i,b)$ = # of b in the domain of the neighbors of i (D)
- $CT(i,a) = |\{ b \in neigh(G,a) \text{ s.t. } R(i,b)>0 \}|$

Prune a from Di when $|neigh(Gp,i)| > CT(i,a)$

- Space complexity $O(nd)$
- Time complexity $O(nd^2)$

Redundant constraint

- **C2** only considers one neighbor of node i
- **C3** considers globally all the neighbors node i
- Possible to consider a subset of the neighbors
- More pruning if the neighbors have similar domains

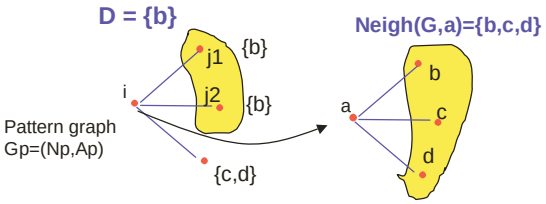


A new constraint

Simple case

- Neighbors *j1* and *j2* of node *i* have similar domain (e.g. they have the same type)

$$C4 : |\ (Dj1 \cup Dj2) \cap neigh(G,a)\ | \geq 2$$



C4 : Implementation

- Using D = Dj1 ∪ Dj2, we get

$$C4 : |\ D \cap neigh(G,a)\ | \geq 2$$

- Can be implemented as **C3**

C4 : general case

- Neighbors *j1, …, jk* of node *i* have similar domain (e.g. they are of the same type)
- With $D = Dj1 \cup \ldots \cup Djk$

$$\textbf{C4} : |\, D \cap neigh(G,a) \,| \geq k$$

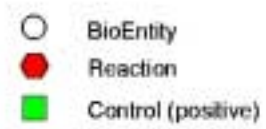
- For $k=|neigh(G,i)|$, **C4** ≡ C3
- For $k=1$, **C4** ≡ C2

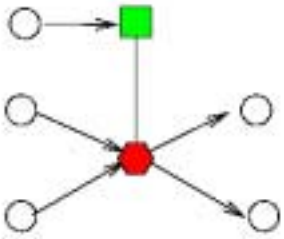
C4 : complexity

- Same complexity than C2, but more constraints
  - Let p = # additional constraints per node
  - Space complexity $O(npd)$
  - Time complexity $O(npd^2)$
- Problem is the potential huge number of such constraints !
- p must be small : only for discriminant properties of nodes such as types

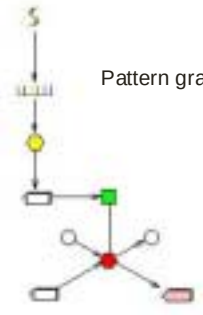
C4 : Example

- Vary useful in biochemical networks

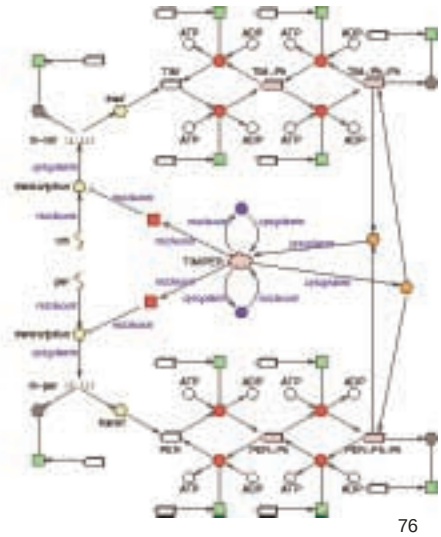




Example



Pattern graph



Extensions

- Introducing properties in nodes and in arcs (e.g. types, attributes)
- Including properties in patterns (e.g. type hierarchy, string patterns)
- Inexact pattern matching (distance between nodes and between arcs)
- Constraints on the pattern
  - This reaction node should have between two and four substrates
- Other pattern matching features
  - Generic arc in pattern representing a path
  - Generic node in pattern representing a (sub)graph

Integration

Constraints and pattern matching

- Constraints on the pattern
  - Constraints on the nodes
  - This reaction node should have between two and four substrates
- Other pattern matching features
  - Generic arc in pattern representing a path
  - Pattern as a **constraint pattern variable**

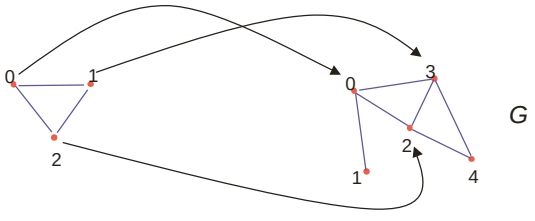
Overview

- Bioinformatics
- Biological networks
- CLP(BioNet)
  - Constraints for bio.net analysis
  - **Constraints for bio.net matching**
- Perspectives

Subgraph isomorphism

- *Gp=(Np,Ap)* : **pattern graph**
- *G=(N,A)* with $|Np| \leq |N|$
- Find a function $f : Np \rightarrow N$ such that
  - *f* is injective
  - $\forall n1, n2 : (n1,n2) \in Ap \Rightarrow (f(n1),f(n2)) \in A$

Pattern graph *Gp*



- Subgraph isomorphism is NP-complete

Family of problems

- Graph vs subgraph isomorphism
- Exact vs inexact matching
  - Nodes with attributes, matching between two node is a *distance*
  - Some arcs from the pattern graph are not considered. Distance from the initial pattern and the chosen subpattern

Analysis of Biochemical Networks

- Subgraph isomorphism is a basic operation for graph pattern matching

Applications

- Compare biochemical networks
  - from different organisms and tissues
  - at different stages of annotation;
  - highlight common features and differences;
  - predict missing elements ('reconstruction')
- Compile repertoires of recurrent network motifs (topological patterns) at different resolution levels

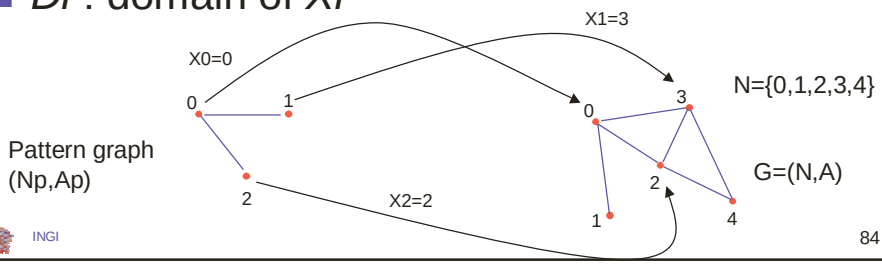
Existing algorithms

Many existing algorithms for standard subgraph isomorphism based on various techniques

- Cliques
- Fuzzy set theory
- Elastic graph matching
- Multiple graph matching
- Error correction
- Genetic algorithms
- Decision tree
- Neural networks
- Clustering
- Connected components
- **Constraint programming** [Rudolf, 1998] [Valiente, 2000]
- …

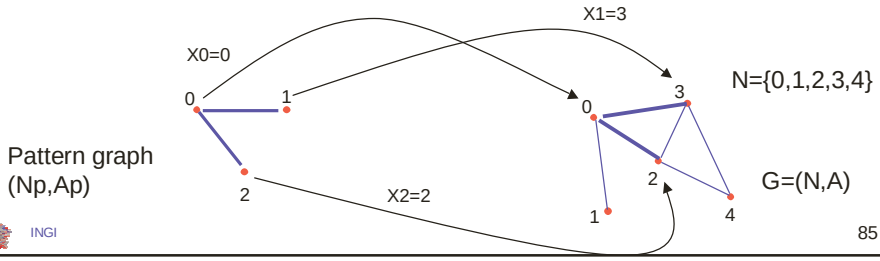
SI as a CSP

- *Gp=(Np,Ap)* : **pattern graph**
- *G=(N,A)* with $n = |Np| \leq |N| = d$
- We use *Np=(X1,…,Xn)*
- **Domain variables** $Xi \in N$
- *Di* : domain of *Xi*



Subgraph Isomorphism as a CSP

- Let *i* and *j* be distinct nodes from pattern graph
- C1 : $Xi \neq Xj$ for all $i \neq j$
- C2 : $(Xi,Xj) \in A$ for all $(i,j) \in Ap$
- [Rudolf, 1998], [Valiente, 2000]
- How to achieve pruning ?

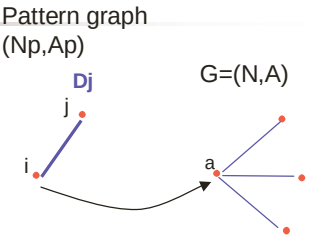


C1 : Pruning

C1 : $Xi \neq Xj$ for all $i \neq j$

- allDiff(X1,..,Xn) constraint
- More pruning than arc consistency on binary constraints
- $O(n^2d^2)$ with $n = |Np|$ and $d = |N|$
- [Regin, 1994]

C2 : Pruning



C2 : *(Xi,Xj) ∈ A* for *(i,j) ∈ Ap*

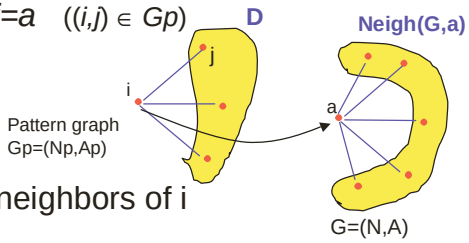
- When can we prune value *a ∈ Di* ?
- If there is no value *b ∈ Dj* s.t. *(a,b) ∈ A*
- Classical arc consistency

Implementation

- Constraint can be reexpressed as
  C2 : *Dj ∩ neigh(G,a) = ∅*
- Independent from *i*; same pruning criteria for all neighbors of j in Ap
- *S(j,a)* = | *Dj ∩ neigh(G,a)* |
- *S(j,a)* : # neigbhors of *a* in *Dj*
- When *S(j,a)=0*, prune *a* from all neighbors of *j* in pattern graph
- **Space complexity** : *O(nd)*
- **Time complexity** : $O(nd^2)$ (instead of $O(n^2d^2)$ for classical AC algorithm)

C3 : Redundant constraint

- Candidate values for *Xj* if *Xi=a* (*(i,j) ∈ Gp*)
  (*Dj ∩ neigh(G,a)*)



- Let $D = \bigcup_{j \in neigh(Gp,i)} Dj$
- Candidate values for all the neighbors of i

  $D \cap neigh(G,a)$
- There must be more candidates than neighbors :

  C3 : $| D \cap neigh(G,a) | \geq | neigh(Gp,i) |$
- **We can prune value *a ∈ Di* if there are more neighbors than candidate values for these neighbors**
- [Valiente, 2000]

C3 : Redundant constraint

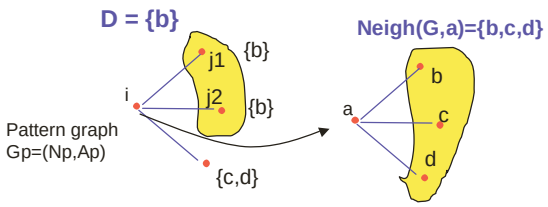
- Can be implemented with the following data structure
- $R(i,b) = |\{ j \in neigh(Gp,i) \text{ s.t. } b \in Dj \}|$
- $R(i,b)$ = # of b in the domain of the neighbors of i (D)
- $CT(i,a) = |\{ b \in neigh(G,a) \text{ s.t. } R(i,b)>0 \}|$

Prune a from Di when $|neigh(Gp,i)| > CT(i,a)$

- Space complexity $O(nd)$
- Time complexity $O(nd^2)$

Redundant constraint

- **C2** only considers one neighbor of node i
- **C3** considers globally all the neighbors node i
- Possible to consider a subset of the neighbors
- More pruning if the neighbors have similar domains

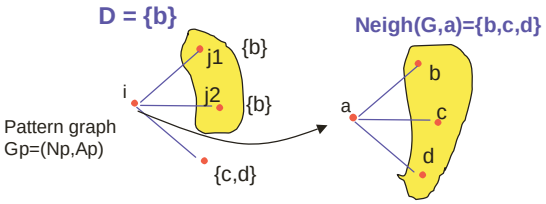


A new constraint

Simple case

- Neighbors *j1* and *j2* of node *i* have similar domain (e.g. they have the same type)

$$C4 : |\ (Dj1 \cup Dj2) \cap neigh(G,a)\ | \geq 2$$



C4 : Implementation

- Using D = Dj1 ∪ Dj2, we get

$$C4 : |\ D \cap neigh(G,a)\ | \geq 2$$

- Can be implemented as **C3**

C4 : general case

- Neighbors *j1, …, jk* of node *i* have similar domain (e.g. they are of the same type)
- With $D = Dj1 \cup \ldots \cup Djk$

$$\textbf{C4} : |\, D \cap neigh(G,a) \,| \geq k$$

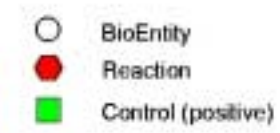
- For $k=|neigh(G,i)|$, **C4** ≡ C3
- For $k=1$, **C4** ≡ C2

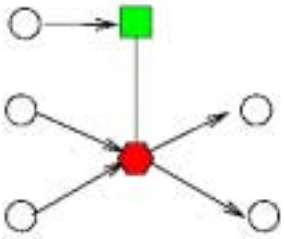
C4 : complexity

- Same complexity than C2, but more constraints
  - Let p = # additional constraints per node
  - Space complexity $O(npd)$
  - Time complexity $O(npd^2)$
- Problem is the potential huge number of such constraints !
- p must be small : only for discriminant properties of nodes such as types

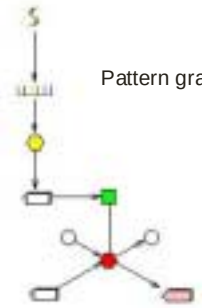
C4 : Example

- Vary useful in biochemical networks

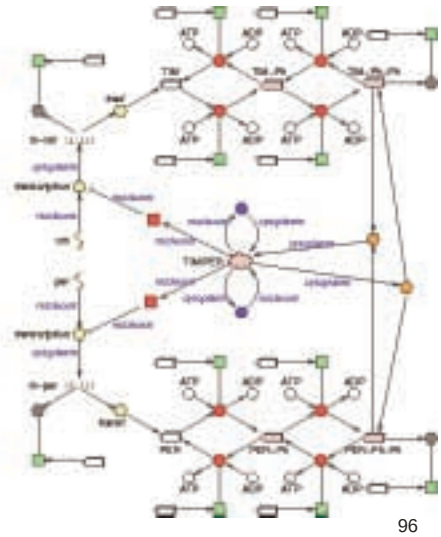




Example



Pattern graph



Extensions

- Introducing properties in nodes and in arcs (e.g. types, attributes)
- Including properties in patterns (e.g. type hierarchy, string patterns)
- Inexact pattern matching (distance between nodes and between arcs)
- Constraints on the pattern
  - This reaction node should have between two and four substrates
- Other pattern matching features
  - Generic arc in pattern representing a path
  - Generic node in pattern representing a (sub)graph

Integration

Constraints and pattern matching

- Constraints on the pattern
  - Constraints on the nodes
  - This reaction node should have between two and four substrates
- Other pattern matching features
  - Generic arc in pattern representing a path
  - Pattern as a **constraint pattern variable**

Overview

- Bioinformatics
- Biochemical networks
- CLP(BioNet)
  - Constraints for bio.net analysis
  - Constraints for bio.net matching
- **Perspectives**

CLP(BioNet)

- Analysis of biochemical networks can be performed on a rich typed graphs representation of the networks
- CLP(BioNet)
  - Introduction of graph domain variable
  - Definition of constraints
  - Analysis of biochemical networks expressed as a combination of constraints
  - Collaboration with biologists
  - Simplicity and versatility of the analysis
  - Extendable framework
  - Preliminary results show the potential of this CP approach

Future work

- Many things remains to be done…
  - Definition of other constraints
  - Integration of constraints and pattern matching
  - Efficient implementation
  - Experimentation in collaboration with biologists
- Using CLP(BioNet) on other domains
  - With complex network representation
  - Analysis of the networks
  - E.g. networking, …

References

- *An Overview of Data Models for the Analysis of Biochemical Pathways* **Yves Deville**, Jacques van Helden, Soshana Wodak David Gilbert *Briefings in Bioinformatics, August 2003*
- *An Object-Oriented Data Model for Signal Transduction* Yves Deville, David Gilbert, Christian Lemer, Jacques van Helden, Shoshana J. Wodak. *ECCB2003.*
- *The aMAZE LightBench: a Web interface to a relational database of cellular processes* Christian Lemer, Erick Antezana, Fabian Couche, Frédéric Fays, Xavier Santolaria, Rekins's Janky, **Yves Deville**, Jean Richelle, Shoshana J. Wodak. *Nucl. Acid Research, 2004*
- *CLP(BioNet).* **Yves Deville,** Pierre Dupont, Grégoire Dooms, Stéphane Zampelli. *Research Report UCL/INGI (to be completed)*

www.info.ucl.ac.be/people/YDE.html

Acknowledgments

aMAZE team

ULB - Belgium
- Erick Antezana
- Fabian Couche
- Fred Fays
- **Olivier Sand**
- **Christian Lemer**
- **Jean Richelle**
- **Jacques van Helden**
- **Soshana Wodak**

(*)Yves Deville (on sabbatical)

BioMaze Project

UCL - Belgium
- **Yves Deville**
- **Pierre Dupont**
- **Stéphane Zampelli**
- **Grégoire Dooms**

FUNDP - Namur
- Jean-Luc Hainaut
- Jean-Marc Hick

ULB - Belgium
- Esteban Zimani
- Sabri

Collaborators

Institut Pasteur - France
- Georges Cohen

Birkbeck College - UK
- Lorenz Wernisch

University of Glasgow - UK
- **David Gilbert**

Univ. Köln - Germany
- Dietmar Schomburg

EBI-EMBL
- Sandra Orchard

External data sources
- Swissprot
- Genbank
- KEGG/LIGAND
- BRENDA
- PUBMED

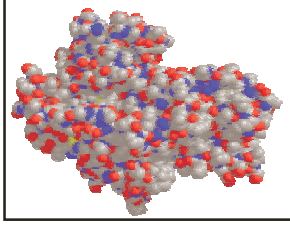
Sponsors
- Astra-Zeneca
- Aventis, Organon
- Roche, (Monsanto)
- EC
- ***Brussels Gov.***
- ***Walloon Region***

Prof. Yves Deville
Computing & Engineering Department
Université catholique de Louvain

CLP(BioNet) : Towards a CLP framework for the analysis of Biochemical Networks

Yves Deville

SweConsNet 2004



Linköping 15 January 2004