Intégration de données hétérogènes
Bases de données non structurées

Master 2

Bioinformatique et Biologie des Systèmes

Programme de la journée “marathon sprinté”

- Matin
  - Apperçu sur
    - Big data, NoSQL, base de données de graphes, neo4j
  - Prise en main de neo4j
    - Construction d'une base de connaissances (graphe) sur les gènes/proéines d'*Escherichia coli*
      - données d'expression
      - données d'interaction protéine-protéine
      - données phylogénomique
      - annotations Gene Ontology
- Après midi
  - Concepts
    - caractérisation d'un ensemble
  - Application
    - Annotations GO sur-représentées parmi les ensembles de gènes/protéines présentant à la fois
      - coexpression,
      - interaction protéine-protéine,
      - conservation du voisinage génomique

Tendances et observations

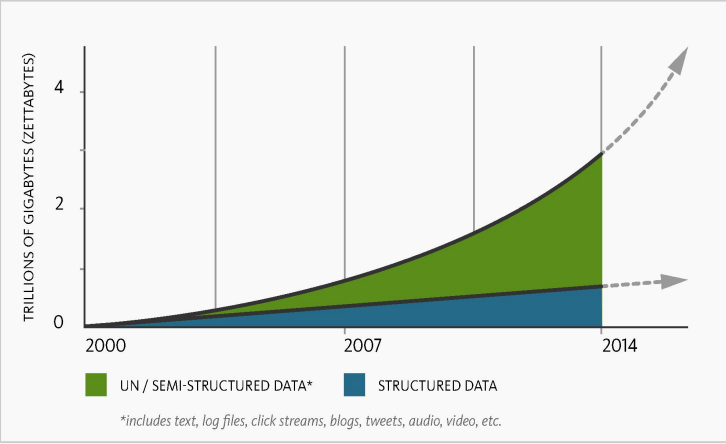
Big data

- “Big data is high volume, high velocity, and/or high variety information assets that require new forms of processing to enable enhanced decision making, insight discovery and process optimization.” (Gartner, 2012)

Données non structurées

- Pareja-Tobes *et al.* 2015 : "Sometimes analysis can be scheduled (annotating a genome for a research project), while in other cases it is inherently impossible to even predict when you will need to run them (think of analyzing data coming from an outbreak). The resources needed or their characteristics vary wildly depending on the specific analysis. We are thus in a context where in general **we do not know in advance what we need to access**, how fast, when, and how."

NoSQL Databases



source: http://disa.fi.muni.cz/david-novak/teaching/nosql-databases-2017/lectures/

NoSQL databases

What is “NoSQL”?

- term used in late 90s for a different type of technology:
  Carlo Strozzi: http://www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/NoSQL/
- “Not Only SQL”? (but many RDBMS are also “not just SQL”)
- “NoSQL is an accidental term with no precise definition”. first used at an informal meetup in 2009 in San Francisco (presentations from Voldemort, Cassandra, Dynomite, HBase, Hypertable, CouchDB, and MongoDB)

NoSQL: Database technologies that are (mostly)

- Not using the relational model (nor the SQL language)
- Designed to run on large clusters (horizontally scalable)
- No schema - fields can be freely added to any record
- Open source
  - Based on the needs of 21st century web estates

Other characteristics (often true):

- easy replication support (fault-tolerance, query efficiency)
- simple API
- eventually consistent (not ACID)

source: David Novak, http://disa.fi.muni.cz/david-novak/teaching/nosql-databases-2017/lectures/

NoSQL Properties in Detail

Flexible scalability
- horizontal scalability instead of vertical

Dynamic schema of data
- different levels of flexibility for different types of DB

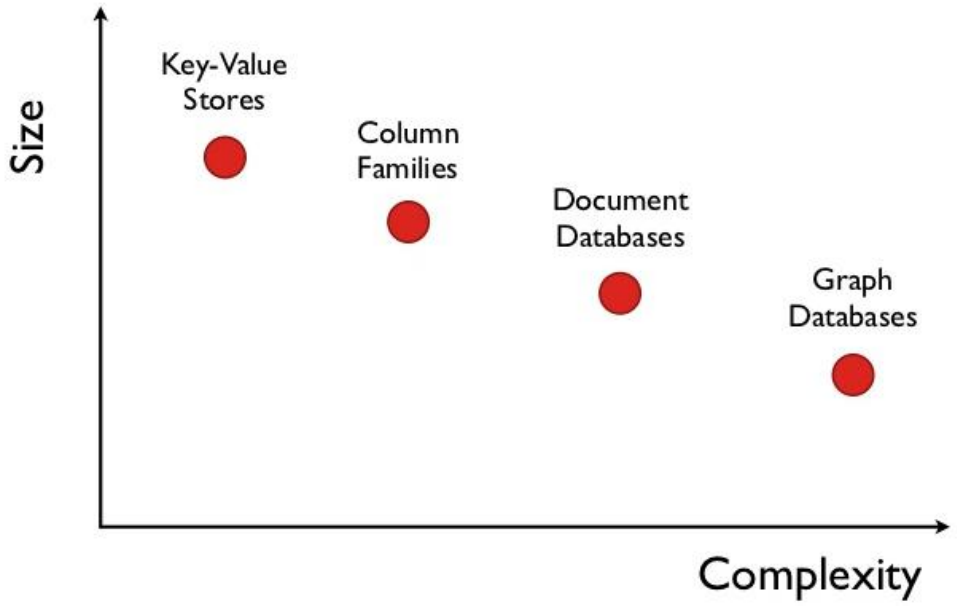
Efficient reading
- spend more time storing the data, but read fast
- keep relevant information together

Cost saving
- designed to run on commodity hardware
- typically open-source (with a support from a company)

source: David Novak, http://disa.fi.muni.cz/david-novak/teaching/nosql-databases-2017/lectures/

- MapReduce programming model
  - running over a distributed file system

- Key-value stores
- Document databases
- Column-family stores
- **Graph databases**

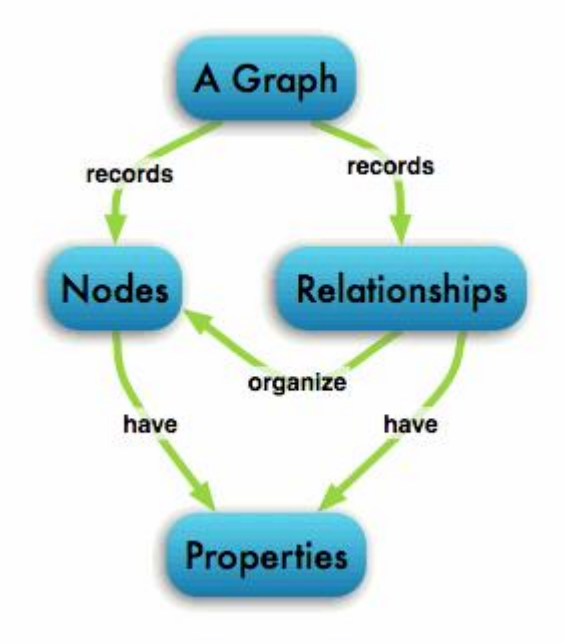


source: http://www.slideshare.net/emileifrem/nosql-east-a-nosql-overview-and-the-benefits-of-graph-databases

source: David Novak, http://disa.fi.muni.cz/david-novak/teaching/nosql-databases-2017/lectures/

Neo4j: Basic Info

- Open source graph database
  - The most popular
- Initial release: 2007
- Written in: Java
- OS: cross-platform
- Stores data as nodes connected by directed, typed relationships
  - With properties on both
  - Called the “property graph”



source: David Novak, http://disa.fi.muni.cz/david-novak/teaching/nosql-databases-2017/lectures/

Neo4j: Basic Features

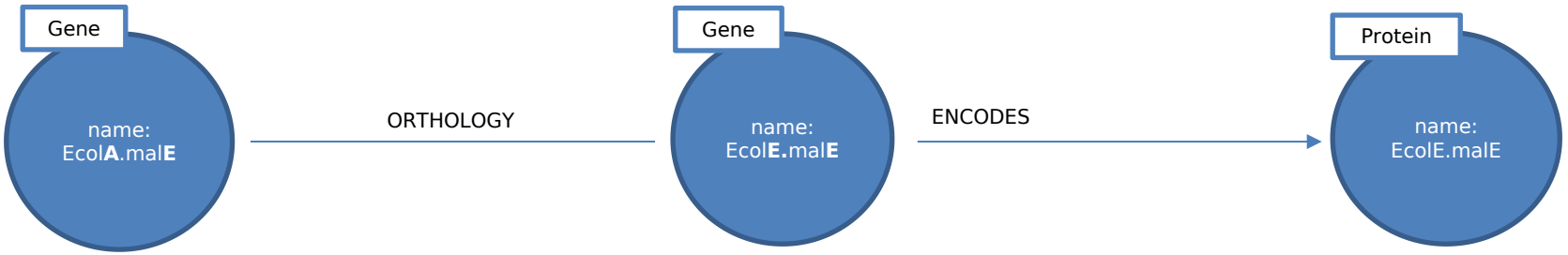
- reliable – with full ACID transactions
- durable and fast – disk-based, native storage engine
- scalable – up to several billion nodes/relationships/properties
- highly-available – when distributed (replicated)
- **expressive** – powerful, human readable graph query language
- fast – powerful traversal framework
- embeddable - in Java program
- simple – accessible by REST interface & Java API

source: David Novak, http://disa.fi.muni.cz/david-novak/teaching/nosql-databases-2017/lectures/

Base de données orientée graphe (*Graph databases*)

- Principes
  - Représenter les données en tant qu'objets reliés par des relations
  - Chaque objet ou relation peut avoir des attributs qui lui sont propres
  - Développement d'un langage de manipulation et de requête

- Neo4j : Labeled property graph

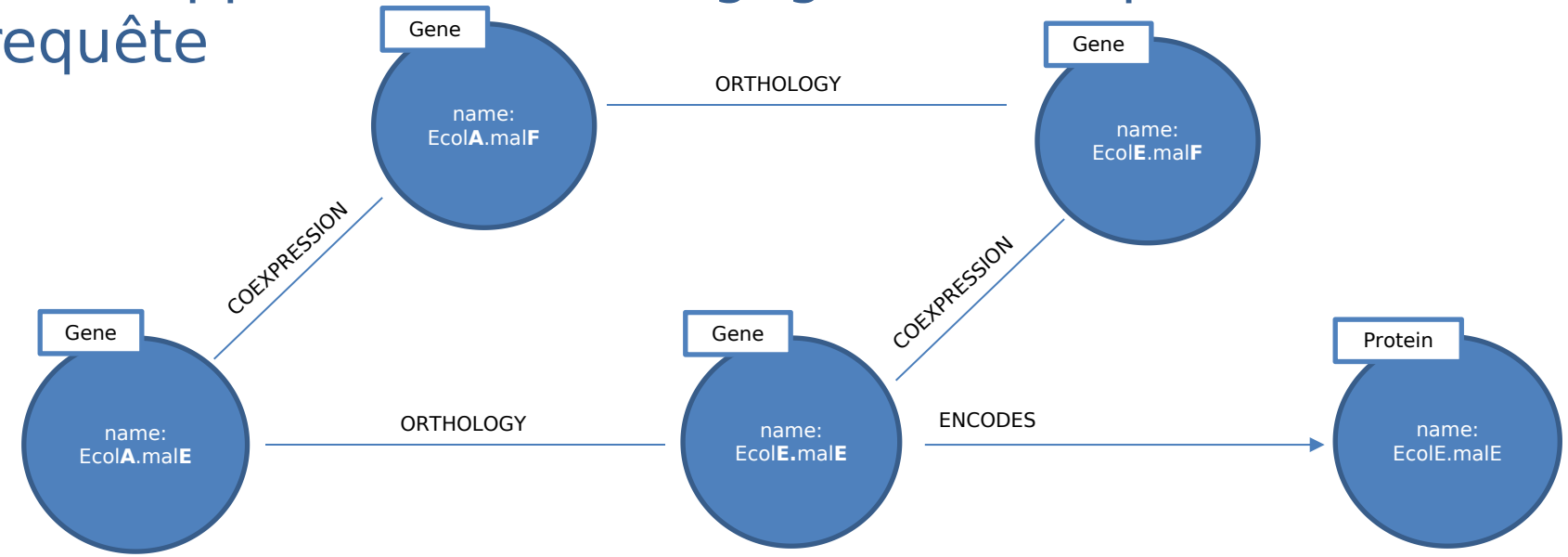


Labeled Property Graph

```
(EcolA.malE:Gene)-[:ORTHOLOGY]-(EcolE.malE:Gene)-[:ENCODES]->(EolE.malE:Protein)
```

Base de données orientée graphe (*Graph databases*)

- Principes
  - Représenter les données en tant qu'objets reliés par des relations
  - Chaque objet ou relation peut avoir des attributs qui lui sont propres
  - Développement d'un langage de manipulation et de requête

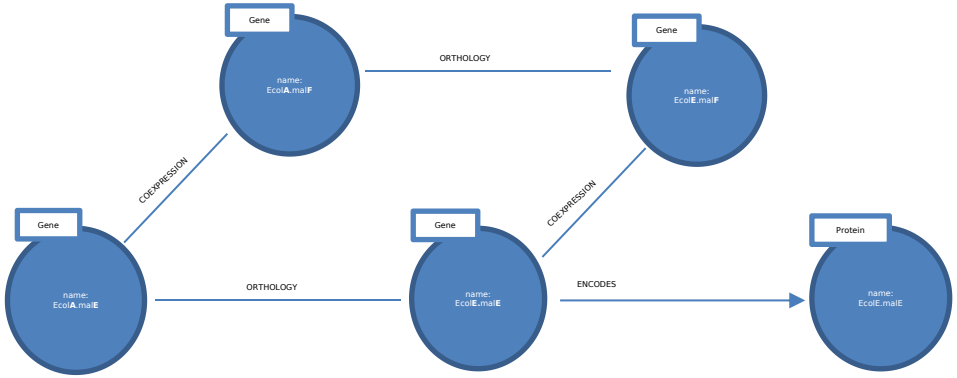


Labeled Property Graph

```
(EcolE.malE:Gene)-[:COEXPRESSION]-(EcolE.malF)-[:ORTHOLOGY]-
(EcolA.malF:Gene)-[:COEXPRESSION]-(EcolA.malE:Gene)-[:ORTHOLOGY]-(EcolE.malE:Gene)
-[:ENCODES]->(EolE.malE:Protein)
```

Labeled property graph

Un graphe avec propriétés étiquetées est constitué de sommets, relations, propriétés et étiquettes :



- Propriétés des sommets : de type clé/valeur

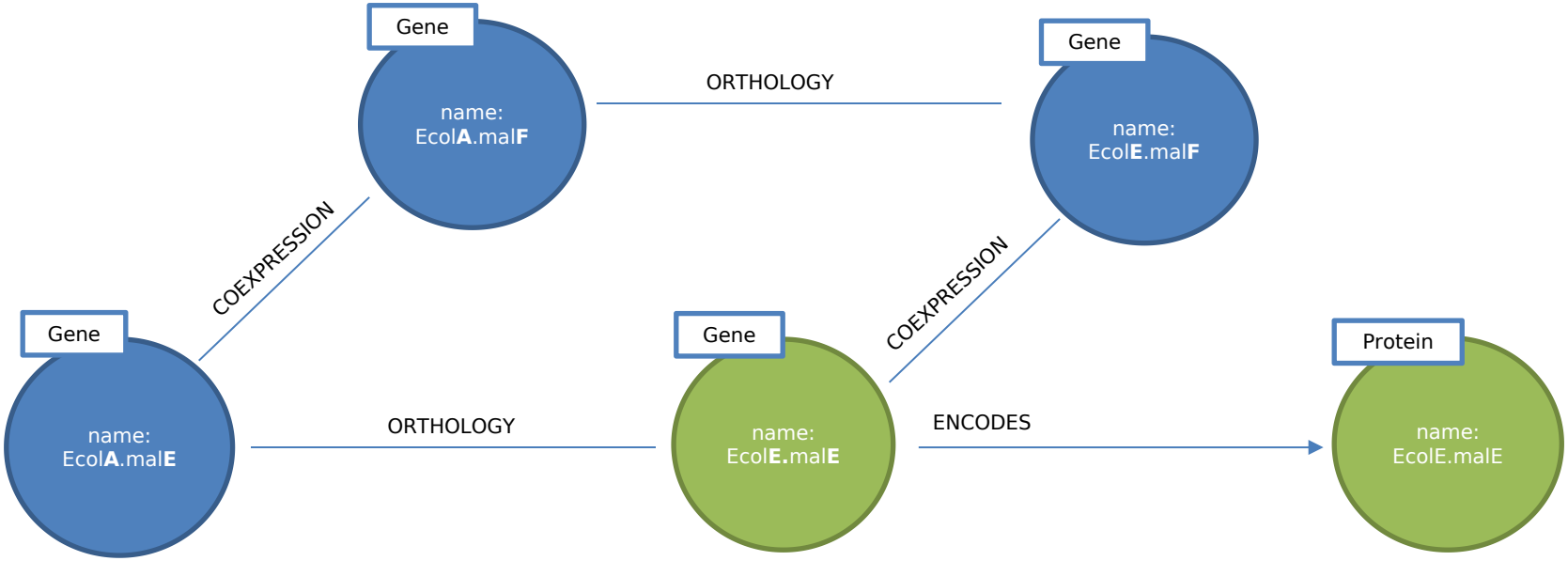
- Étiquettes des sommets : une ou plusieurs afin de les regrouper (`Gene`, `Protein`)

- Relations : orientées, peuvent avoir des propriétés comme les sommets.

Langage de requête, exemple : Cypher

```
MATCH (g:Gene)-[:ENCODES]->(p:Protein)
WHERE g.name='EcolE.malE'
RETURN g,p

ou
MATCH (g:Gene {name: 'EcolE.malE'})-[:ENCODES]->(p:Protein)
RETURN g,p
```



Cypher Language

- Neo4j graph query language
  - For querying and updating
- Declarative – we say what we want
  - Not how to get it
  - Not necessary to express traversals
- Human-readable
- Inspired by SQL and SPARQL
- Still growing = syntax changes are often

http://neo4j.com/docs/stable/cypher-query-lang.html

source: David Novak, http://disa.fi.muni.cz/david-novak/teaching/nosql-databases-2017/lectures/

Cypher: Clauses

- **MATCH**: The graph pattern to match
- **WHERE**: Filtering criteria
- **RETURN**: What to return
- **CREATE/MERGE**: Creates nodes and relationships.
- **DELETE**: Remove nodes, relationships, properties
- **SET**: Set values to properties
- **WITH**: Divides a query into multiple parts

source: David Novak, http://disa.fi.muni.cz/david-novak/teaching/nosql-databases-2017/lectures/

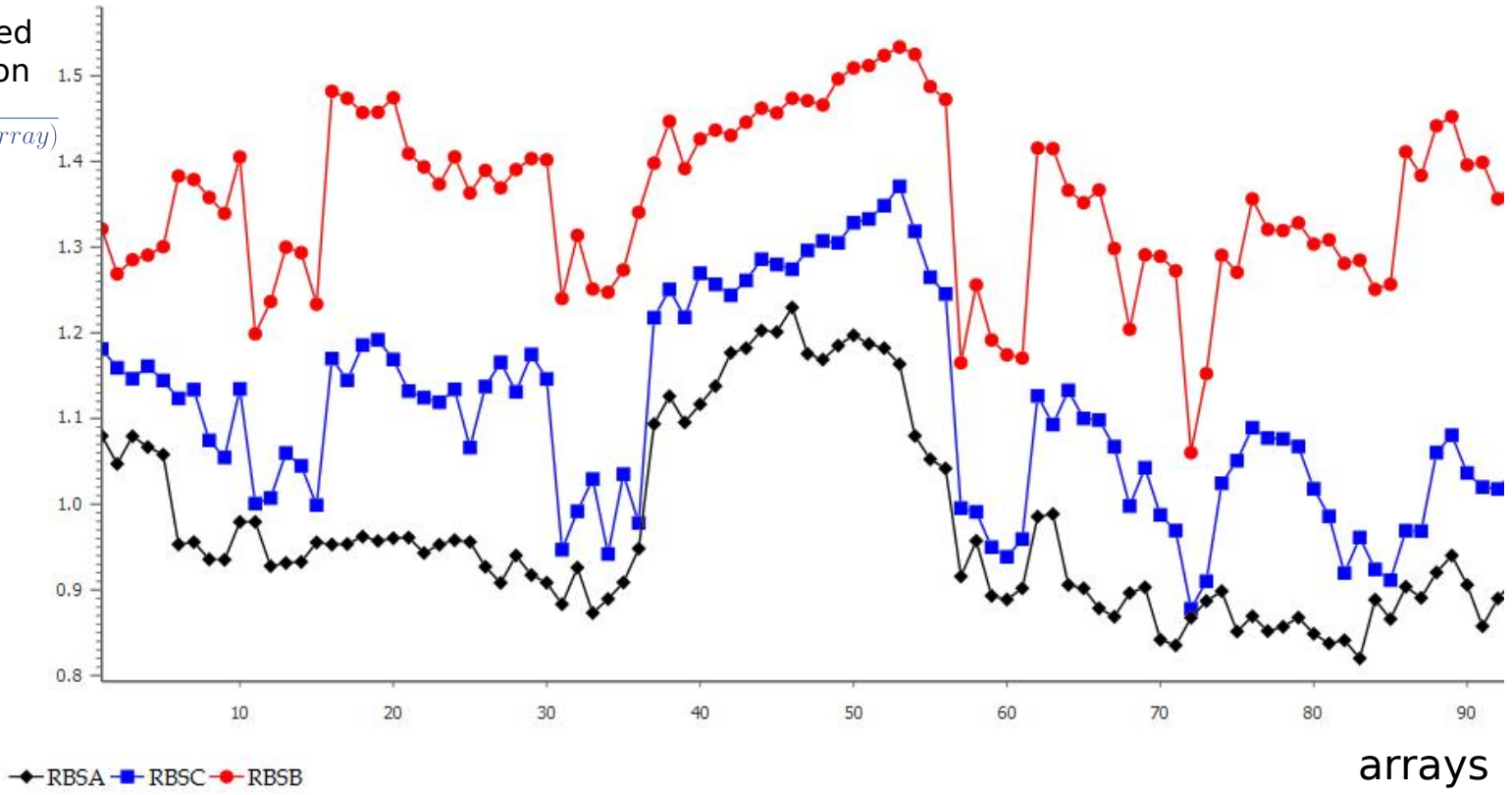
Access to Neo4j

- Embedded database in Java system
- Language-specific connectors
  - Libraries to connect to a running Neo4j server (python, R, ...)
- Cypher query language
  - Standard language to query graph data
- HTTP REST API
- etc.

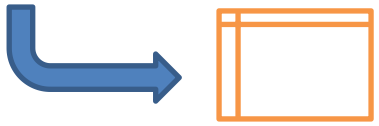
Gene expression

normalized expression

$$y' = \frac{y}{median(array)}$$



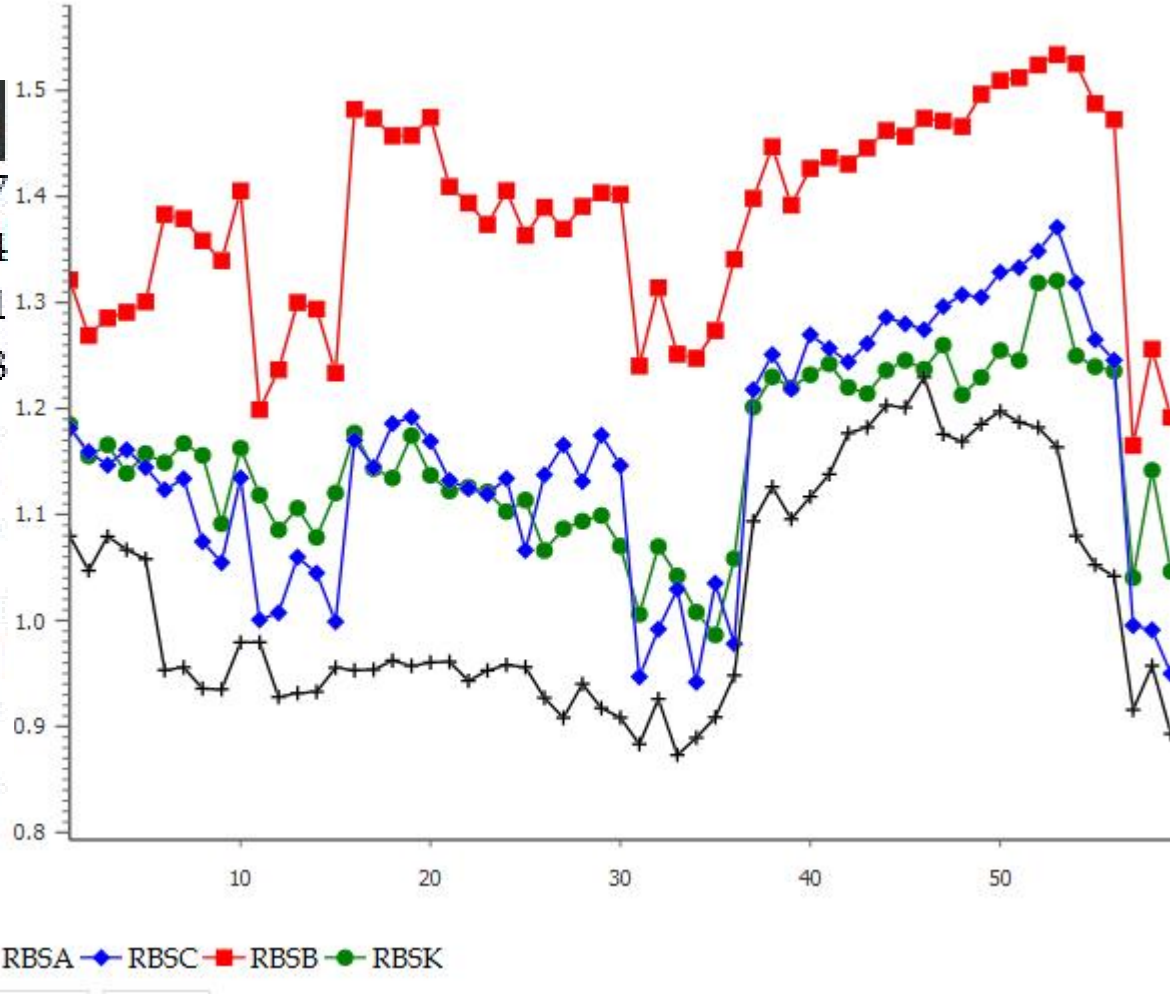
- a gene: set of expression values in various experimental conditions
- a pair of genes: dissimilarity index based on Pearson's correlation coefficient
- score : average dissimilarity



gene pairwise dissimilarity matrix

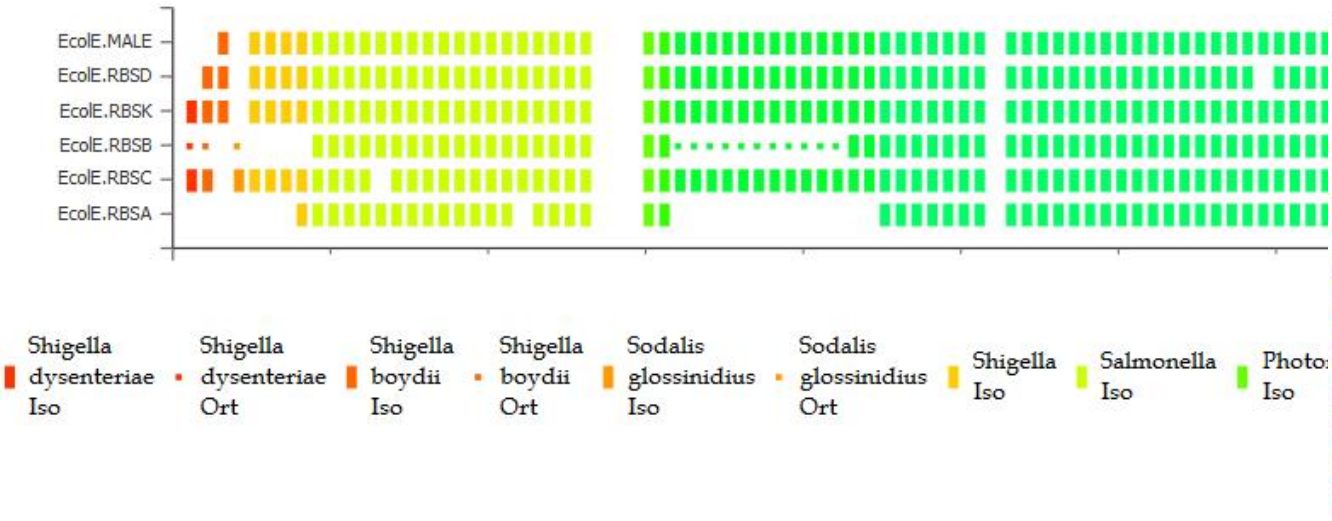
- training: rbsA, rbsB, rbsC in *E. coli* K-12

candidate	score	rank	rank ratio
RBSK	0.1870	1	0.0002467
RBSD	0.2695	2	0.0004934
FDOI	0.3288	3	0.0007401
MALE	0.3514	4	0.0009868
MALK	0.3537	5	0.001234
FDOG	0.3551	6	0.001480
FDOH	0.3670	7	0.001727
TREB	0.3679	8	0.001974
NUPG	0.3841	9	0.002220
LAMB	0.3850	10	0.002467
MALF	0.3933	11	0.002714



Phylogenetic profiles

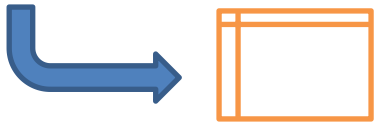
training: rbsA, rbsB, rbsC



candidate	score	rank	rank ratio
RBSD	0.6304	1	0.0002369
MGSA	0.7274	2	0.0004739
CDAR	0.7280	3	0.0007108
CYTR	0.7285	4	0.0009478
GLPT	0.7416	5	0.001185
PTSG	0.7474	6	0.001422
MALG	0.7475	7	0.001659
RBSK	0.7486	8	0.001896
CPDB	0.7533	9	0.002132
POTB	0.7536	10	0.002369
FLIY	0.7560	11	0.002606

sinia

- a gene: presence/absence of orthologs 1:1 in other genomes
- pair of genes: dissimilarity index based on the Jaccard index
- score: average dissimilarity



gene pairwise dissimilarity matrix

Phylogenetic data

- Phylogenetic profiles

- gene pairwise distance matrix computation
  - Hypothesis: genes located near each other in a set of genomes are likely to be functionaly related
  - g_1' and g_2' orthologs 1:1 of $gene_1$ and $gene_2$ in another genome *i*
  - Probability that the distance D_i is smaller than the observed distance d_i

$$p_i = Pr(D_i \leq d_i) = \frac{2d_i}{N_i - 1}$$

  - For a set of *M* other genomes

$$d = Pr(D_1 \leq d_1, \ldots, D_M \leq d_M) = \prod_{i=1}^{M} p_i$$

  - *M* depends on the pair of genes considered
  - *d* not comparable between genes (*e.g.* $0.1^6=10^{-6}$ *vs.* $0.5^{20}=9.5\ 10^{-7}$)
  - normalization: log transformation, z-score, average of distance matrix and its transpose

Phylogenetic data: genome selection

- Reference genomes should not be too evolutionary close to the genome of interest

- Reference genomes should not be redundant in order not to introduce biases

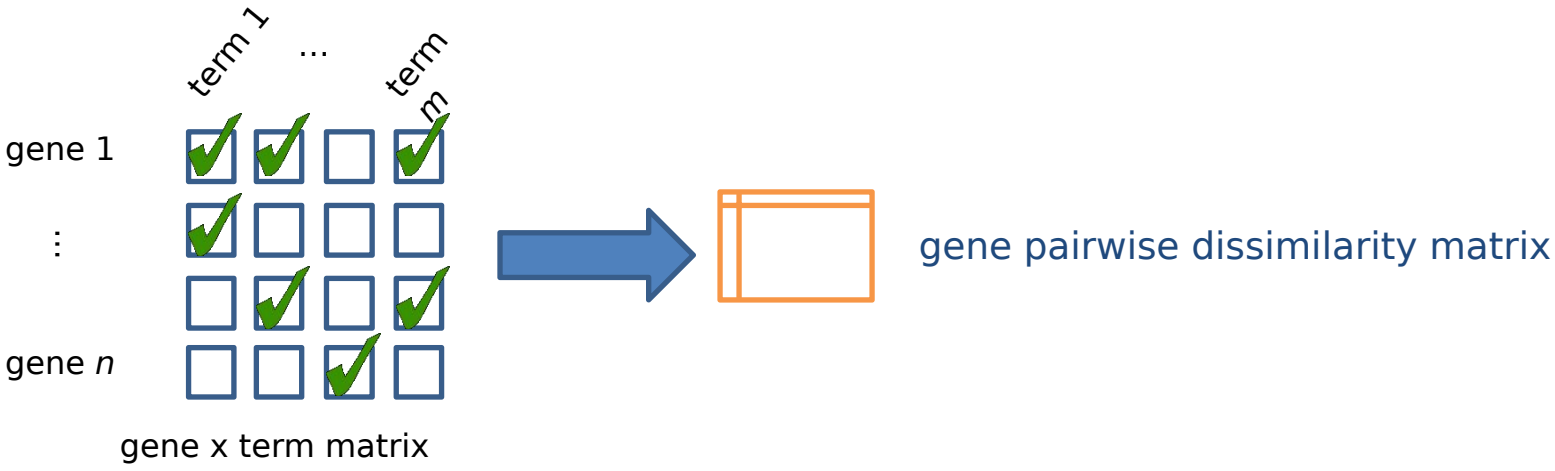
- Need to estimate the relevance of a genome with respect to
  - A genome of interest: is it not too closely related? is it informative?
  - A set of already selected reference genomes: redundancy *vs.* additional signal

- Parameters
  - Rearrangements
    - Significance of genes proximity on the chromosome
  - Core genome size
    - Maximize the coverage of the genome of interest

Approaches:

- gene-term matrix: distance between rows
  - manhattan/euclidean, Jaccard, ...

  but: same weight for each GO-term
- based on GO-term similarity
  - adapt weight to information content



Similarité entre Termes GO

- Node/term information content

$$IC(term) = -\log p(term)$$ with p(term)= freq(term)

- *MICA*(t_1,t_2): Maximum Information Common Ancestor

$$MICA(t_1,t_2) = \arg\max IC(t_i), t_i \in ancestors(t_1,t_2)$$

- $sim_{res}(t_1,t_2) = IC(MICA(t_1,t_2))$ [Resnik, 1995]

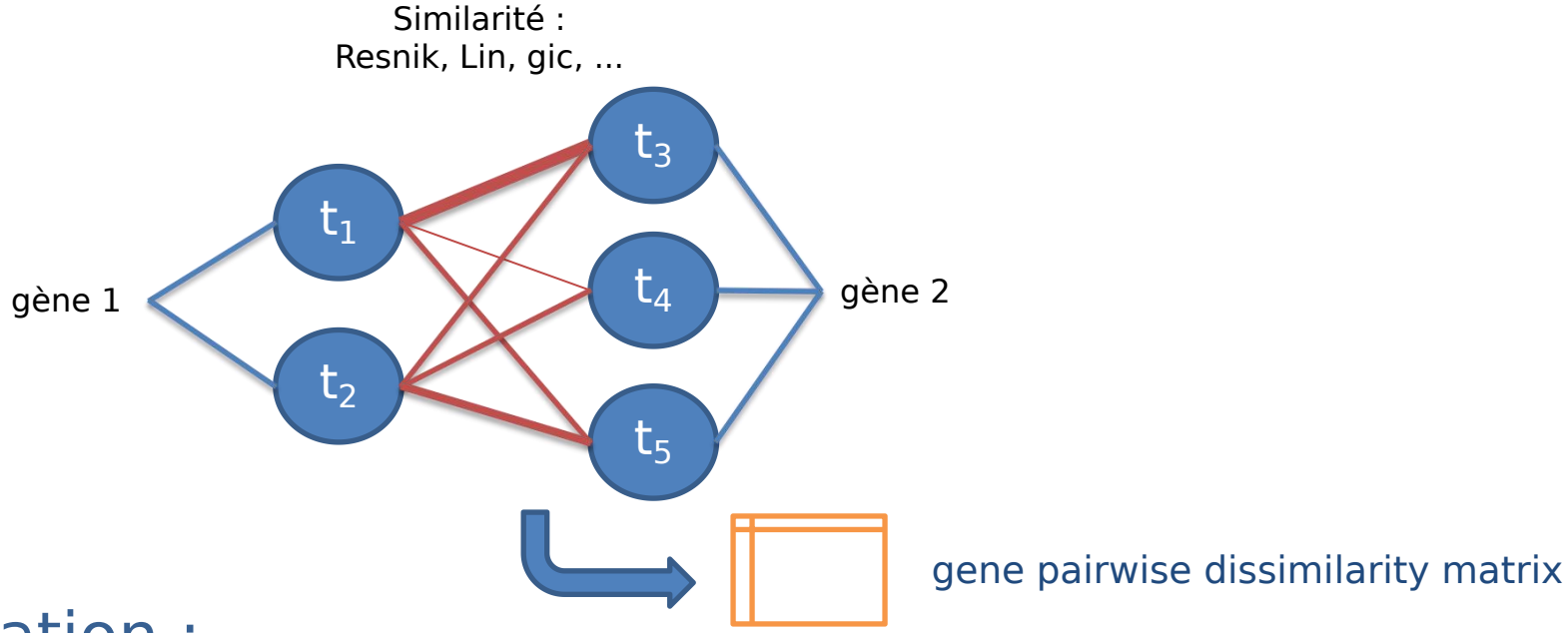
- $sim_{lin}(t_1,t_2) = IC(MICA(t_1,t_2))/(IC(t_1)+IC(t_2))$ [Lin, 1998]

- $$sim_{gic}(t_1,t_2) = \frac{\sum_{t \in \{GO(t_1) \cap GO(t_2)\}} IC(t)}{\sum_{t \in \{GO(t_1) \cup GO(t_2)\}} IC(t)}$$ [Pesquita *et al.*, 2008]

Simimlarité entre gènes basée sur la similarité entre termes GO

Possibilités :

- Similarité moyenne des termes communs au 2 gènes
- Similarité maximale, ex : t_1-t_3
- Best Match Average (bma), ex : ave(t_1-t_3, t_2-t_5)

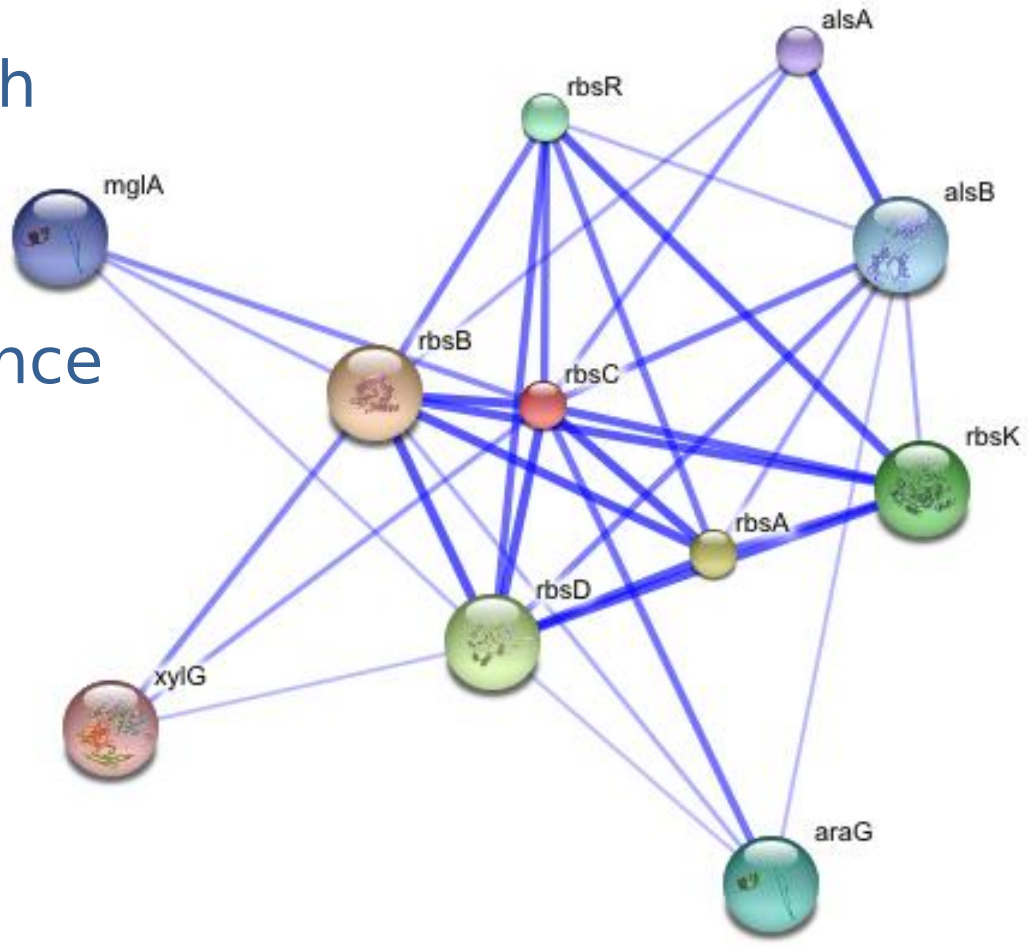


Application :

- Performances légèrement meilleures obtenues que les autres avec la combinaison Resnik + similarité maximale
- à confirmer sur d'autres jeux de données ou d'autres contextes

Interactions

- all pairs shortest path
- a pair of gene:
  shortest path length
- score: average distance



from STRING
http://string-db.org

training: rbsA, rbsB, rbsC

candidate	score	rank	rank ratio
RBSK	1.000	2	0.0005136
RBSD	1.000	2	0.0005136
RBSR	1.000	2	0.0005136
ALSB	1.333	5	0.001284
ALSC	1.333	5	0.001284
YPHD	1.333	5	0.001284
MGLC	1.667	10.5	0.002696
XYLG	1.667	10.5	0.002696
ALSA	1.667	10.5	0.002696
YTFT	1.667	10.5	0.002696