Analysis of big bio-molecular networks through semi-supervised graph-based learning methods

Giorgio Valentini
DI – Dipartimento Informatica
Università degli Studi di Milano

Computer Science Department

UNIVERSITÀ DEGLI STUDI DI MILANO

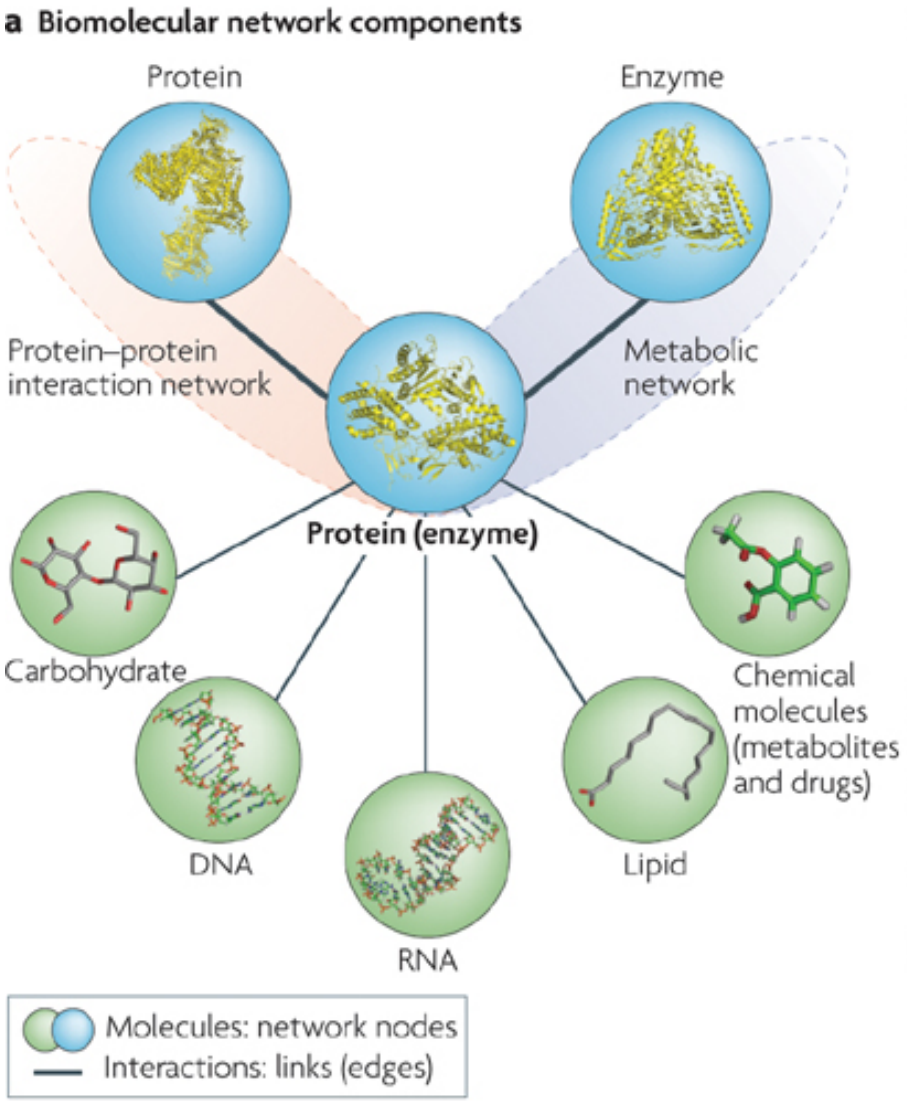
Anacleto Lab

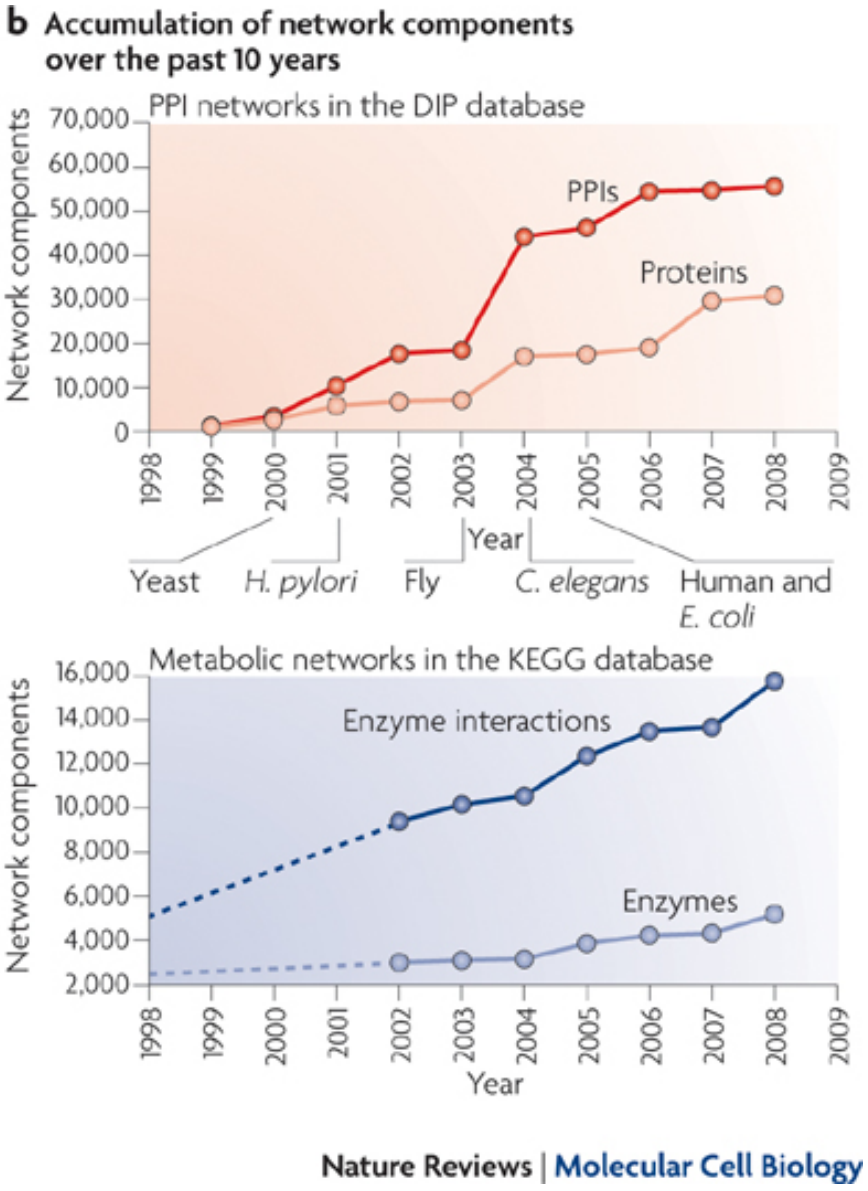
Computational Biology and Bioinformatics

Outline

- Relevant problems in molecular biology and medicine can be modeled through graphs
- Most network-based algorithms show serious scaling problems with big networks
- Strategies to learn big bio-molecular networks
- Analysis of huge biological networks with off-the-shelf machines through secondary memory-based computation techniques

Biomolecular networks

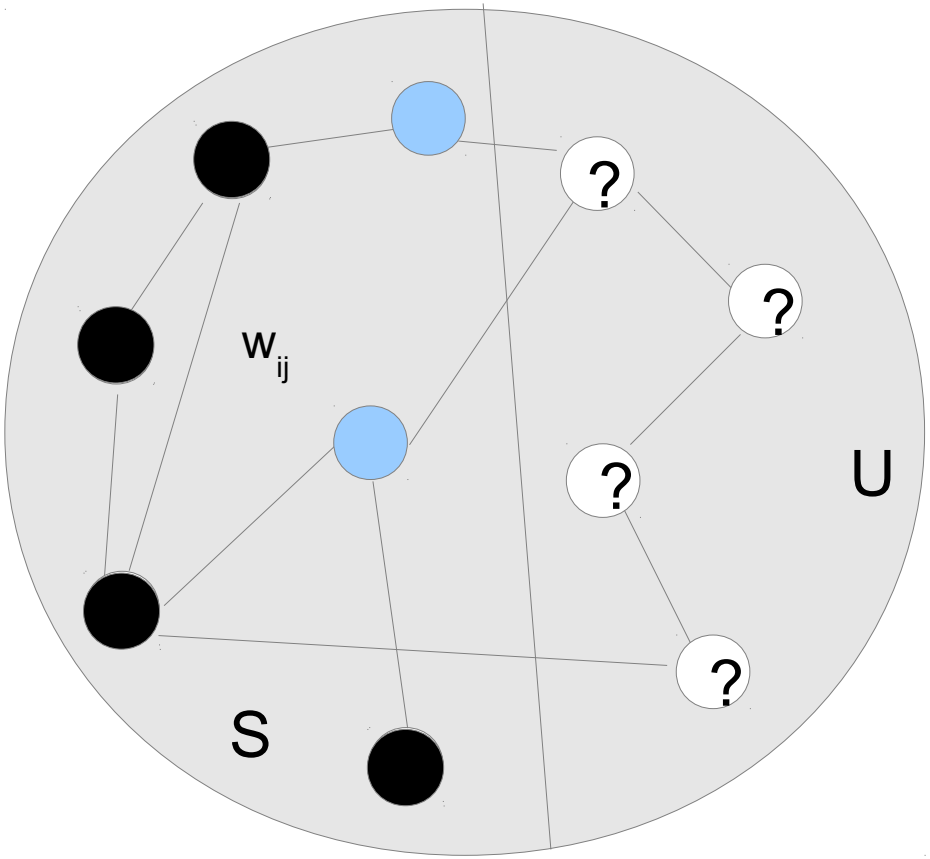




Nature Reviews | Molecular Cell Biology

Graph Semi-Supervised Learning (GSSL) problem

$G = \langle V, E \rangle$



V : proteins,genes,drugs,...

E : functional similarities/relationships

W : similarity matrix

S : labeled nodes

U : unlabeled nodes

GOAL: predict labels for unlabeled nodes (*labeling problem*) or rank nodes with respect to the class to be predicted (*ranking problem*)

State-of-the-art node labeling/ranking methods in computational biology

- Guilt by association (*Marcotte* et al., 1999, *Oliver* et al. 2000)
- Evaluation of functional flow in graphs (*Vazquez* et al. 2003)
- Hopfield network-based methods (*Karaoz* et al. 2004, *Bertoni et al*. 2011)
- Local learning and weighed integration (*Chua* et al 2007)
- Label propagation based on Markov fields (*Deng* et al. 2004)
- Kernel methods for semi-supervised learning and integration of networks (*Tsuda* et al. 2005, *Borgwardt et al.* 2011)
- Label propagation based on Gaussian random fields and ridge regression (*Mostafavi* et al. 2008)
- Random walk-based algorithms (*Kohler et al*., 2008, *Bogdanov* and *Singh*, 2010)
- Kernelized score functions (RANKS, *Valentini et al*. 2016)

State-of-the-art node labeling/ranking methods in computational biology

- Guilt by association (*Marcotte* et al., 1999, *Oliver* et al. 2000)
- Evaluation of functional flow in graphs (*Vazquez* et al. 2003)
- Hopfield network-based methods (*Karaoz* et al. 2004, *Bertoni et al*. 2011)
- Local learning and weighed integration (*Chua* et al 2007)
- Label propagation based on Markov fields (*Deng* et al. 2004)
- Kernel methods for semi-supervised learning and integration of networks (*Tsuda* et al. 2005, *Borgwardt et al.* 2011)
- Label propagation based on Gaussian random fields and ridge regression (*Mostafavi* et al. 2008)
- Random walk-based algorithms (*Kohler et al*., 2008, *Bogdanov* and *Singh*, 2010)
- Kernelized score functions (RANKS, *Valentini et al*. 2016)

Limitations of methods applied in biological networks analysis

1) For many interesting biological problems **labeled data can be rare/expensive**. Moreover wet lab experiments are time consuming. Unlabeled data is much cheaper.

2) Methods classically applied in biological networks analysis suffer of **serious scalability limitations**. It is fairly common to see in literature biological networks learning experiments involving tens of thousands of genes but network learning experiments based on networks with hundreds of thousands of nodes are rare.

At today there are no attempts to perform learning tasks with biological networks composed by millions of nodes (i.e. multi-species genes/proteins network).

A relevant computational biology problem: Multi-species protein function prediction

Can we predict the functions of proteins belonging to different species, by using graph based learning methods?

Can exisiting network-based learning algorithms scale with big protein networks?

How to construct multi-species functional networks?

UniprotKB/TrEMBL (2015)

~550.000 manually annotated proteins
~90 millions of sequences

Approaches to the scalability problem

1) Graph compression
- Compression by node ordering (*Chierichetti* et al 2009)
- Compression by matrix factorization (*Nourbakhsh, Bulò and Pelillo* 2014)

2) Parallel distributed computation
- MapReduce framework (*Dean* and *Ghemawat*, 2004)
- Distributed graph parallel learning (*Gonzalez* et al. 2012)

Issues:
- Partitioning graphs across cluster nodes is hard (*Leskovec* et al 2009)
- Debugging and optimization is difficult
- Requires cluster / cloud systems

3) Secondary memory-based computation
- Graph Database technologies (*Webber* et al. 2012)
- Secondary memory-based systems for the analysis of big graphs (*Kyrola* et al. 2014)

Issues:
- Design of **novel data structures** to store graphs on disks
- **Efficient I/O operations** and **graph processing** on disk

A novel approach to big biological network analysis

1) Local implementation of GSSL algorithms

- Vertex-centric computation: *think globally and solve locally*.
- "*Local*" implementation of "*global*" algorithms:
  - classical : Random walk
  - novel : RANKS (*Re* et al, 2012)

2) Secondary memory-based computation

(a) Graph DB technologies (*Webber* et al. 2012)

(b) Disk-based systems for the analysis of big graphs (*Kyrola* et al. 2012)

- No assumptions on the structure of the graph
- Algorithms efficiently computed with and without approximation

A secondary memory-based approach to biological network analysis

M. Mesiti, M. Re, G. Valentini Think globally and solve locally: secondary memory-based network learning for automated multi-species function prediction, GigaScience, 3:5, 2014

"Local" implementation

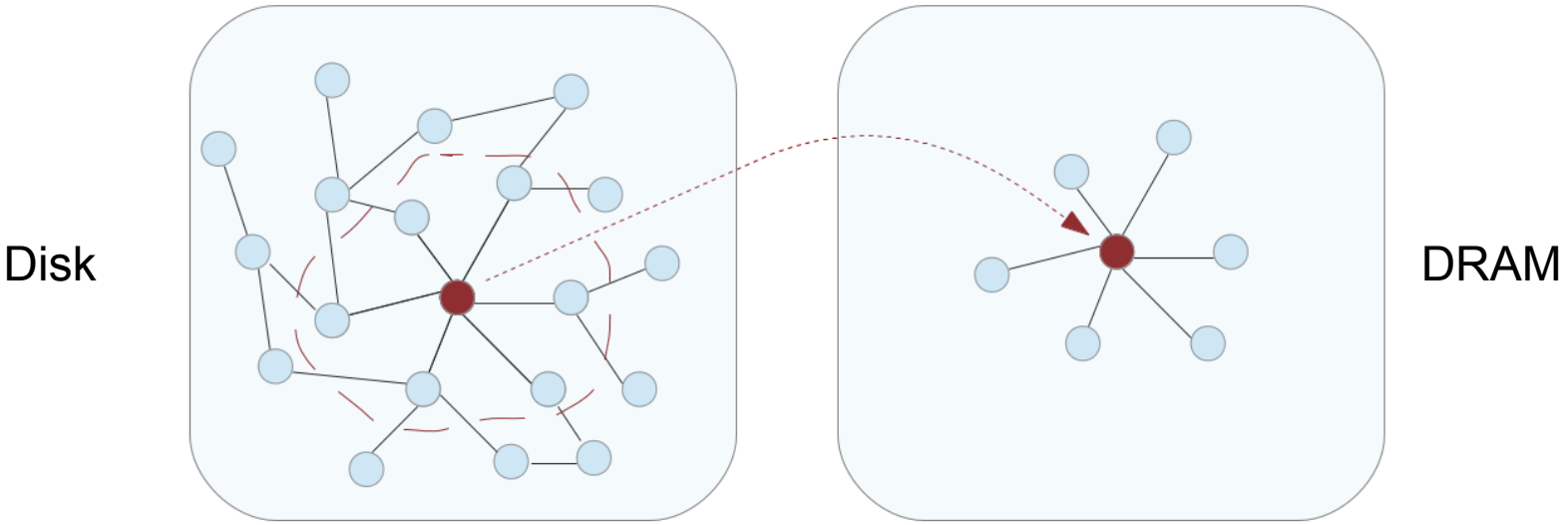
+

"disk-based" computation

=

analysis of big biological graphs **on single PCs**

"local" implementation of network-based algorithms



- We need DRAM to *store <u>only the neighborhood of a single node</u>*
- ***Vertex centric*** *computational model*:

translate "global" network-based methods to "local" implementation

The problem is: can we express a global GSSL algorithm as an iterative computation involving each time **only a single vertex and its neighborhood**?

An example: the classical random walk algorithm

Random walk: the classical algorithm in "global" version:

$\boldsymbol{W}$: weighted adjacency matrix of the graph

$\boldsymbol{D}$: diagonal matrix with $d_{ii} = \sum_j w_{ij}$ $\quad \boldsymbol{Q} = \boldsymbol{D}^{-1}\boldsymbol{W}$: the stochastic matrix

Probability update : $\boldsymbol{p}^{t+1} = \boldsymbol{Q}^{T}\,\boldsymbol{p}^{t}$

Random walk: the "local" vertex-centric implementation:

$$p_i^{t+1} = Q_i\, p^t = D^{-1} W_i\, p^t = \sum_j d_{jj}\, w_{ji}\, p_j^t$$

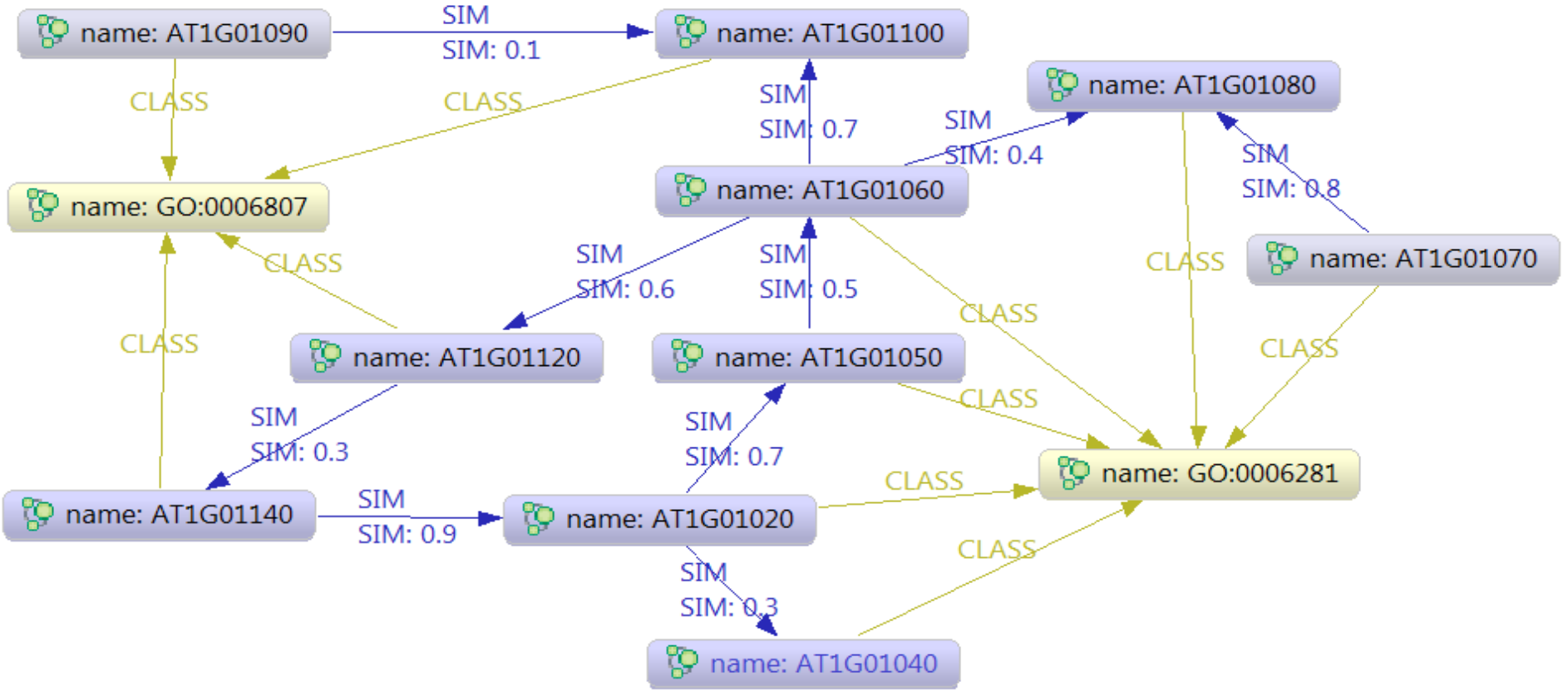
For each vertex i we need only its neighbours (at worst the ith column of $\boldsymbol{W}$, the diagonal of $\boldsymbol{D}^{-1}$ and the probabilities computed at the previous iteration)

But we need fast disk access ...

A first solution to efficient disk-based processing of graphs: graph DB technologies

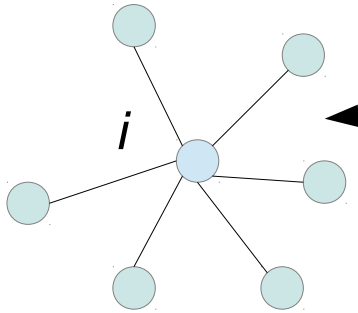
Neo4j (*Webber* et al. 2012):
a DB management system

- graph data model
- graph compute engine



Advantages : fast queries on graphs
Limitations: difficult to manage dynamic systems

RW local Implementation with Neo4J



A db transaction on the neighborhood of node i

```
Update_node (i, GO_term)  {
    Begin TRANSACTION
        p(i) ← 0
        for each edge e outcoming from i {
            j ← getOutNode(e,i)
            if j is annotated for GO_term
                p(i) ← p(i) + p(j) q_ji
        }
    End TRANSACTION
}
```

Synchronous model of computation: we need to maintain two copies of p(*i*)

A second possible approach: GraphChi (Kyrola et al. 2012)

GraphChi:
a disk-based system for the analysis of big graphs on a single PC

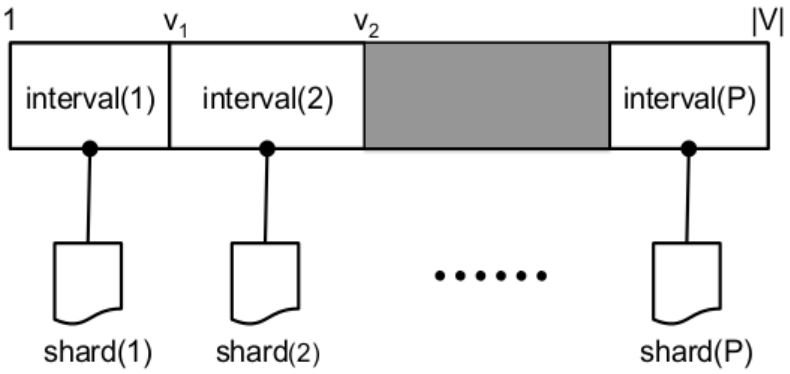
Methods for efficiently breaking large graphs into small parts

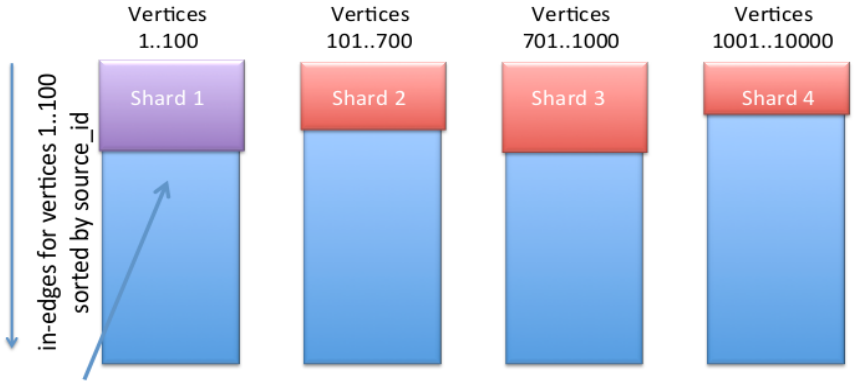
Efficient disk I/O. Small number of non sequential accesses to disk: PSW system

Efficient management of evolving graphs

Asynchronous model of computation

GraphChi: Parallel Sliding Windows (PSW)





Vertices split in P intervals. For each interval: in-edges stored in a shard, sorted by out-edges	⇨	To read each interval at most P non sequential reads (PSW method)	READ
Multi-thread asynchronous computation in main mem.	⇨	Parallel update of vertices and edges in the memory shards	EXEC
Blocks written back to disk	⇨	At most P^2 non sequential reads/writes on disk/full pass on the graph	WRITE

Experiments:

- **13** organisms
- **202,442** proteins
- **25,132,538** edges
- **50** classes

M. Mesiti, M. Re, G. Valentini Think globally and solve locally: secondary memory-based network learning for automated multi-species function prediction, GigaScience, 3:5, 2014

5 folds CV. Learning method: classical random walk. Implementations: GraphChi, Neo4j (a graph database)

Empirical time complexity :

***Eukarya-net*: Average per-term empirical time complexity between*Neo4j* and*GraphChi* implementations**

	16 Gb RAM machine server		4 Gb RAM machine notebook	
Algorithm	***Neo4j***	***GraphChi***	***Neo4j***	***GraphChi***
RW - 1 step	189.60s	20.44s	2520.00s	21.46s
RW - 2 steps	367.82s	31.68s	4919.35s	33.19s
RW - 3 steps	549.84s	45.73s	7333.10s	46.69s

Experiments: Comparison of multi-species and single species approaches

-

Table 9 Comparison of the average AUC, precision at 20% recall (P20R) and precision at 40% recall between multi-species and single-species approaches with 301 species of bacteria

Multi-species approach			
Algorithm	**AUC**	**P20R**	**P40R**
RW - 1 step	0.8744	0.2264	0.1673
RW - 2 steps	0.8590	0.1318	0.0893
RW - 3 steps	0.8419	0.1064	0.0713
Single-species approach			
Algorithm	**AUC**	**P20R**	**P40R**
RW - 1 step	0.8263	0.1801	0.1176
RW - 2 steps	0.8146	0.1059	0.0647
RW - 3 steps	0.8179	0.1009	0.0563

On going work on multi-species protein function prediction (MAFP) with kernelized score function

1. GraphChi vertex-centric implementation of the kernelized score functions

2. Construction of a big network including all the core proteins of the STRING database:
- more than 400 organisms
- 1.5 millions of proteins
- hundreds of millions of edges
- thousands of GO functional classes to be predicted

Main goals:
- Showing that MAFP can be exploited on off-the-shelf computers
- Showing that multi-species functional prediction significantly improves on single species functional prediction.

On going work on multi-species protein function prediction (MAFP) with kernelized score function

1. GraphChi vertex-centric implementation of the kernelized score functions

2. Construction of a big network including all the core proteins of the STRING database:
- more than 400 organisms
- 1.5 millions of proteins
- hundreds of millions of edges (intra and inter-species)
- thousands of GO functional classes to be predicted

A vertex centric implementation is not straightforward

The construction of Inter-species edges is not straightforward

Main goals:
- Showing that MAFP can be exploited on off-the-shelf computers
- Showing that multi-species functional prediction significantly improves on single species functional prediction.

Vertex centric implementation of the Average Score Random Walk Kernel (Lin et. al. 2017)

Proposition 1. The vector $S_{AV}(V_C)$ of average scores for the whole graph $G =< V, E >$ with a p-step random walk kernel starting from nodes of $V_C \subset V$ can be computed as $S_{AV}(V_C) = \boldsymbol{D}^{\frac{1}{2}}\boldsymbol{v}^p$ by the iterative formula

$$\boldsymbol{v}^p = \boldsymbol{M}\boldsymbol{v}^{p-1} \quad \text{where } \boldsymbol{M} = [(a-1)\boldsymbol{I} + \boldsymbol{D}^{-1}\boldsymbol{W}]$$

with the initialization vector $\boldsymbol{v}^0$ having element

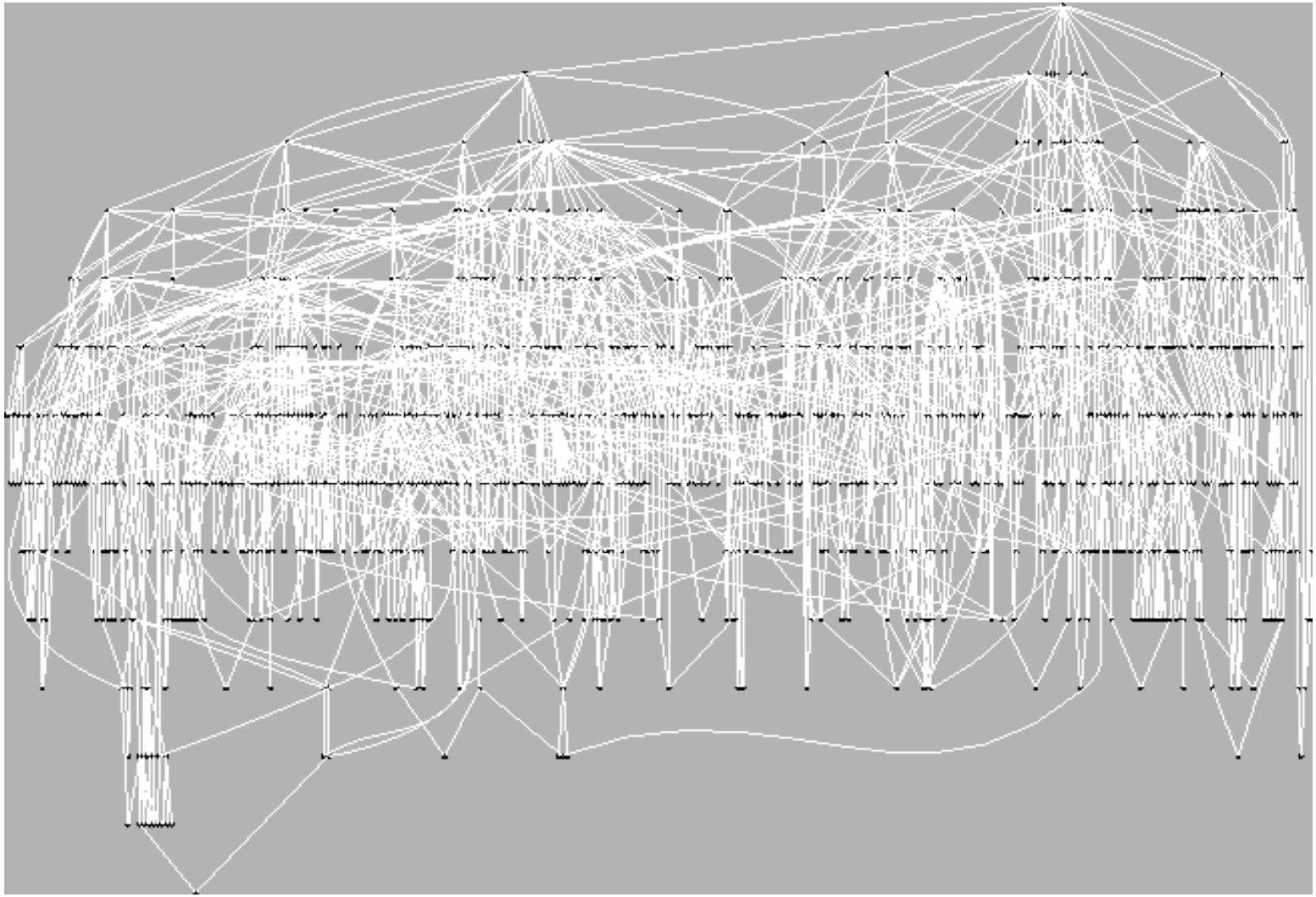
$$v_i^0 = \begin{cases} \frac{1}{|V_C|\sqrt{d_{ii}}} & \text{if } i \in V_C; \\ 0 & \text{otherwise.} \end{cases}$$

$$v_i^p = d_{ii}^{-1} \sum_{j \in I(i)} w_{ij} v_j^{p-1} + (a-1) v_i^{p-1}$$

$$S_{AV}(i, V_C) = d_{ii}^{\frac{1}{2}} v_i^p$$

Vertex-centric implemention of the *RWK score function*

Scalable methods for biological ontologies

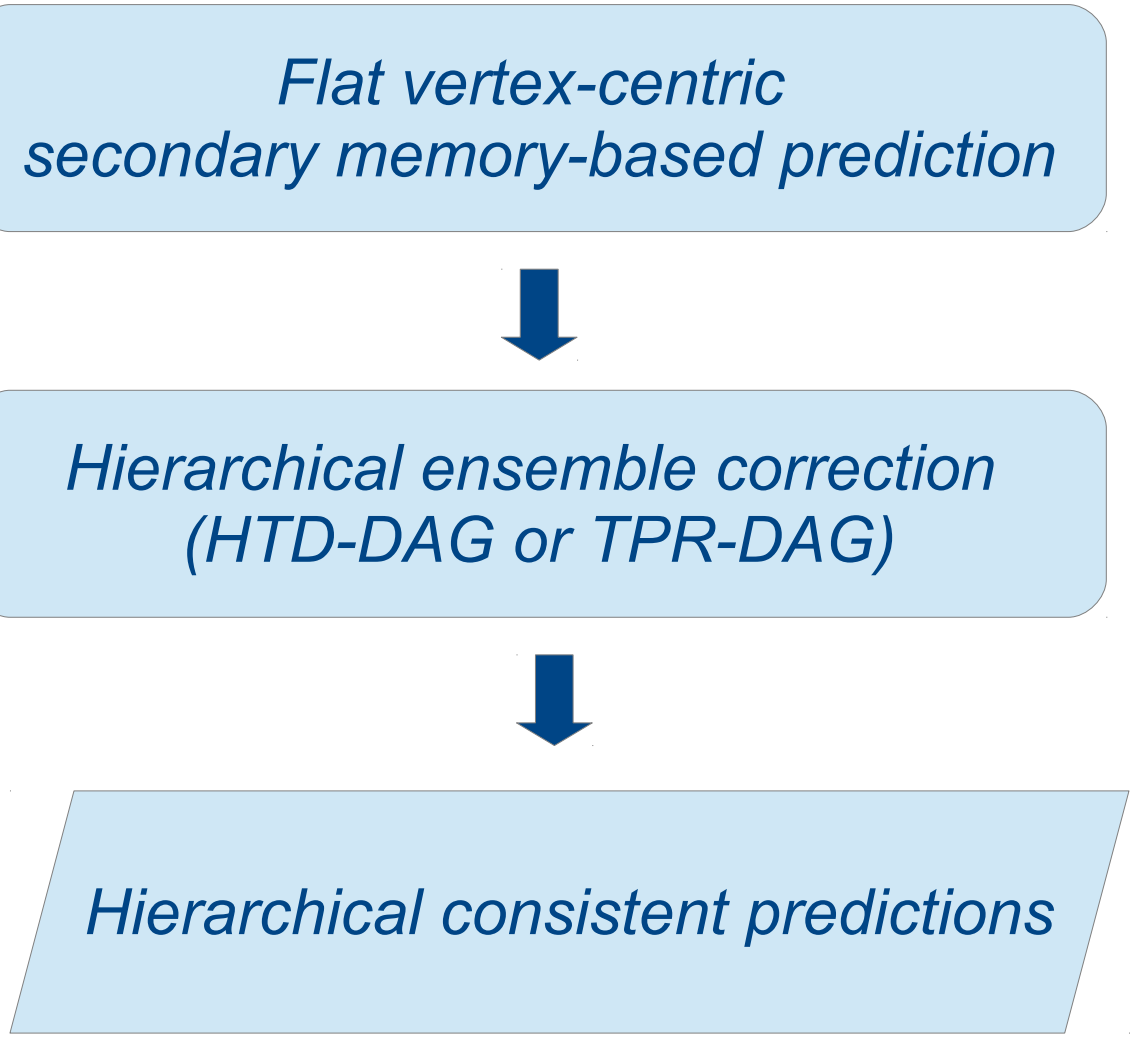


GO DAG of the BP ontology *(S. cerevisiae):*

1074 GO classes (nodes) connected by 1804 edges.

Graph realized through *HCGene* (Valentini and Cesa-Bianchi, *Bioinformatics*, 2008)

In perspective: scalable approaches for hierarchical predictions in huge networks



Summary:

- Semi-supervised graph-based methods are widely applied in several relevant problems in computational biology and medicine
- Kernelized score functions is a flexible algorithmic framework that can be applied in a broad range of interesting bioinformatics problems
- Kernelized score functions and the others state-of-the-art semi-supervised learning methods for biological network analysis are affected by serious scalability problems on big networks
- Local implementation of GSSL methods coupled with the usage of recent secondary memory technologies can make feasible GSSL tasks on very large (and dense) graphs, allowing novel biological insights from the analysis of bio-medical networks.

References:

- J. Lin, M. Mesiti, M. Re and G. Valentini.
  Within network learning on big graphs using secondary memory-based random walk kernels, Complex Networks & Their Applications V: Proceedings of the 5th International Workshop on Complex Networks and their Applications (COMPLEX NETWORKS 2016), *Studies in Computational Intelligence*, Springer, pp. 235-245, 2017
- M. Mesiti, M. Re, G. Valentini.
  Think globally and solve locally: secondary memory-based network learning for automated multi-species function prediction,
  GigaScience, 3:5, 2014
- M. Re, M.Mesiti, G. Valentini, An automated pipeline for multi-species protein function prediction from the UniProt Knowledgebase, Automated Function Prediction SIG 2014 - ISMB, Boston, USA, 2014.
- M. Re, M.Mesiti, G. Valentini, On the Automated Function Prediction of Big Multi-Species Networks, Network Biology SIG 2014 - ISMB, Boston, USA, 2014.