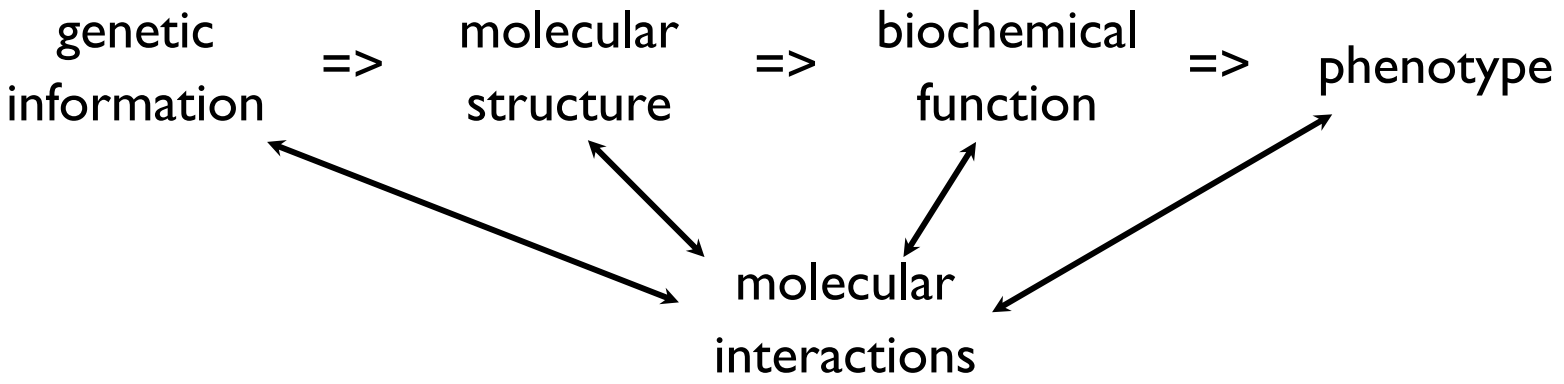
Bioinformatics III:
Network View of Cell Biology

Molecular Systems Biology: considers molecules and their interactions



→ highly connected network of various interactions, dependencies

=> study networks

Lecture – Table of contents - Chapters

Lecture – type of mathematics

Mathematical concept	Object of Investigation	Analysis of Complexity	Time-dependent	Treated in Chapter #
Mathematical graphs	protein-protein networks; protein complexes; gene regulatory networks	Yes	no	**5, 6, 9, 10**
Stoichiometric analysis; matrix algebra	metabolic networks*	yes (count # of possible paths that connect two metabolites)	no	**12**
Differential Equations	signal transduction, energy transduction, gene regulatory networks	No	yes	**9,13**
Equations of motion	individual proteins, protein complexes		yes	**14, 15**
Correlation functions, Fourier transformation	reconstruction of two- and three-dimensional structures of cellular structures and individual molecules	No	yes, when applied on time-dependent data	**2**
Statistical tests	Differential expression and methylation; enriched network motifs	No	yes, when applied on time-dependent data	**8, 9, 10**
Machine learning (linear regression, hidden Markov model)	Predict gene expression, classify chromatin states	No	no	**8, 11**

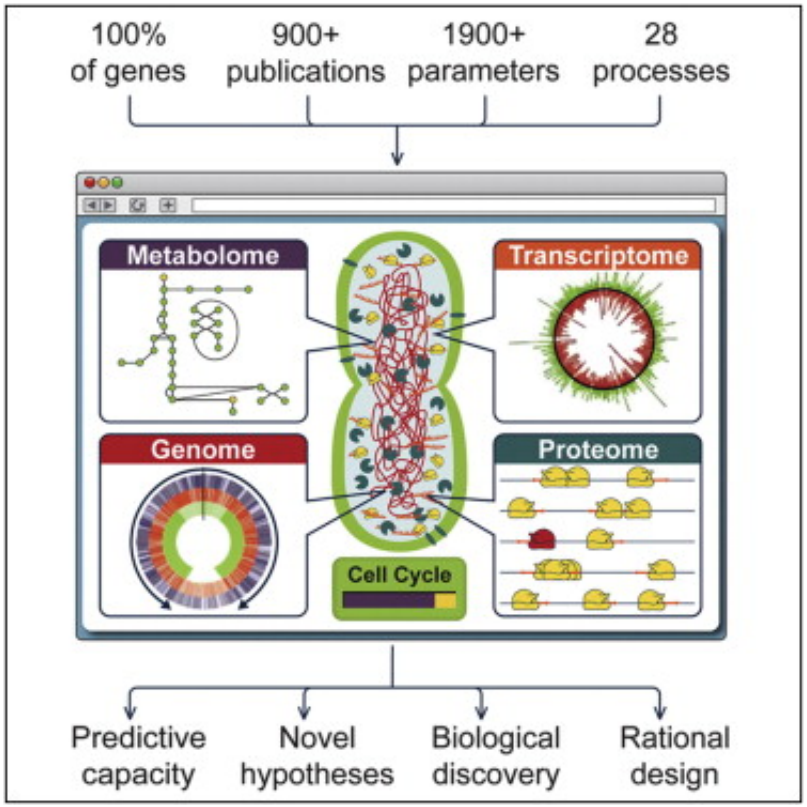
Appetizer: A whole-cell model for the life cycle of the human pathogen *Mycoplasma genitalium* (15.2)

Theory

Cell

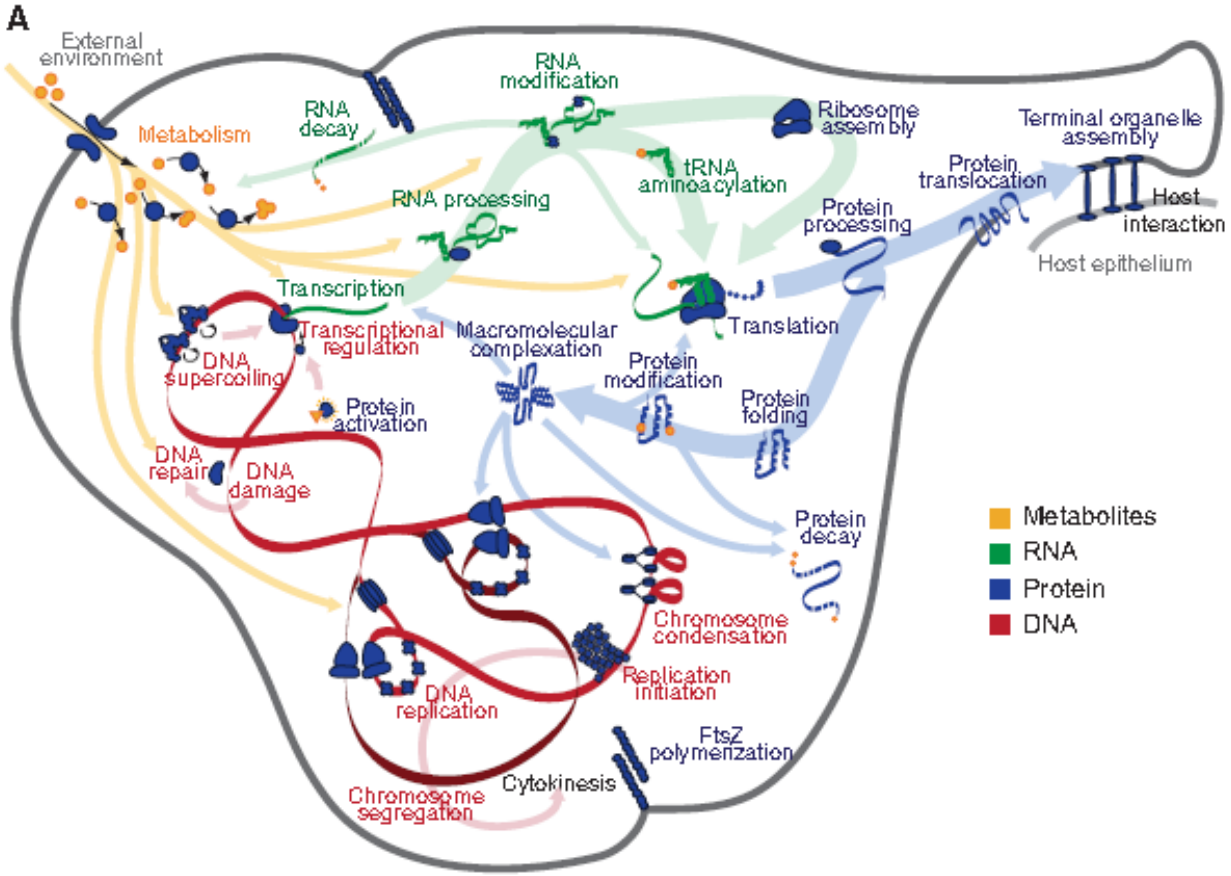
A Whole-Cell Computational Model Predicts Phenotype from Genotype

Jonathan R. Karr,[1,4] Jayodita C. Sanghvi,[2,4] Derek N. Macklin,[2] Miriam V. Gutschow,[2] Jared M. Jacobs,[2] Benjamin Bolival, Jr.,[2] Nacyra Assad-Garcia,[3] John I. Glass,[3] and Markus W. Covert[2,*]
[1]Graduate Program in Biophysics
[2]Department of Bioengineering
Stanford University, Stanford, CA 94305, USA
[3]J. Craig Venter Institute, Rockville, MD 20850, USA
[4]These authors contributed equally to this work
*Correspondence: mcovert@stanford.edu
http://dx.doi.org/10.1016/j.cell.2012.05.044



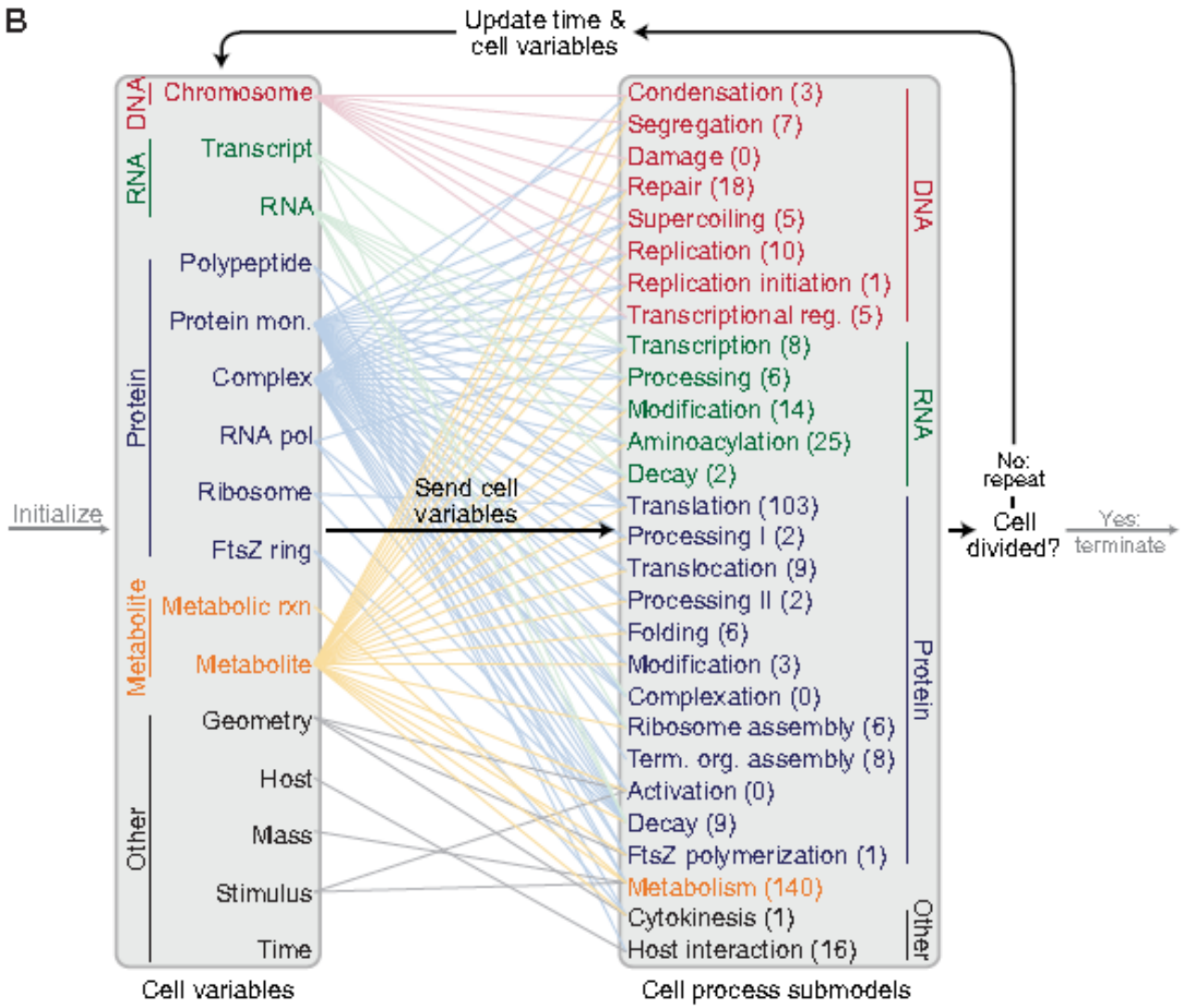
Cell 150, 389-401 (2012)

Divide and conquer approach (Caesar): split whole-cell model into 28 independent submodels



28 submodels are built / parametrized / iterated independently

Cell variables



System state is described by 16 cell variables

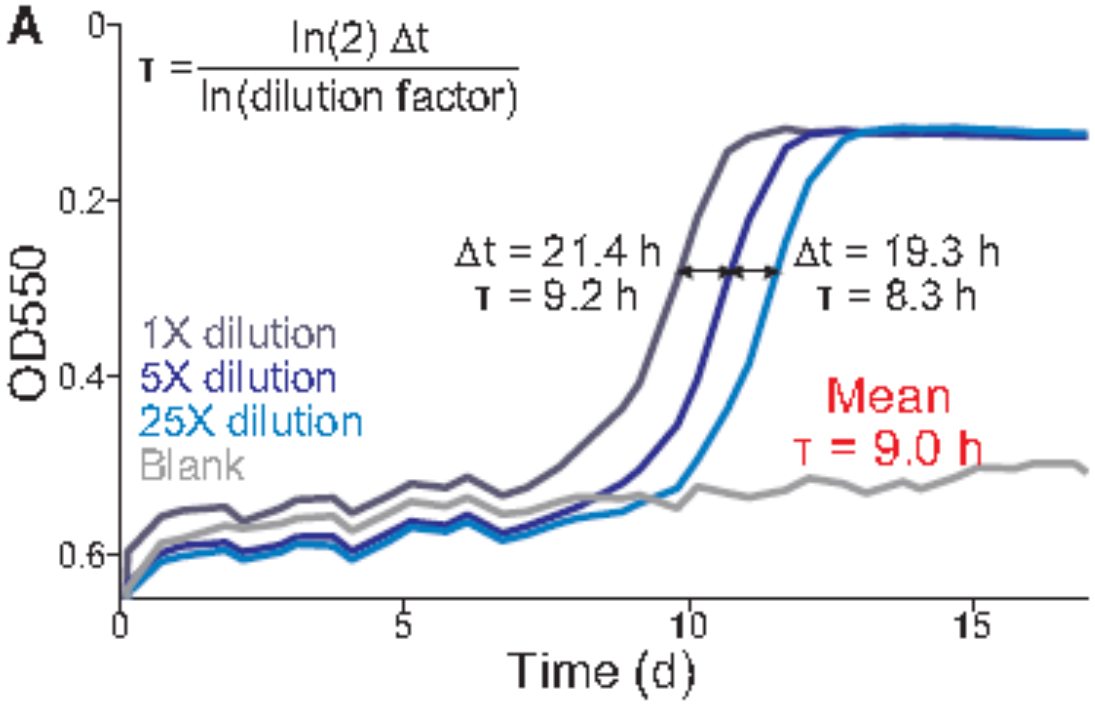
Colored lines: cell variables affected by individual submodels

Mathematical tools:
-Differential equations
-Stochastic simulations
-Flux balance analysis

List S1. Primary sources of the *M. genitalium* reconstruction.

Data source	Content
Bernstein *et al.*, 2002[602]	mRNA half-lives
BioCyc[6]	Genome annotation, metabolic reactions
BRENDA[570]	Reaction kinetics
CMR[168]	Genome annotation
Deuerling *et al.*, 2003[388]	Chaperone substrates
DrugBank[847]	Antibiotics
Eisen *et al.*, 1999[891]	DNA repair
Endo *et al.*, 2007[391]	Chaperone substrates
Feist *et al.*, 2007[558]	Metabolic reactions
Glass *et al.*, 2006[193]	Gene essentiality
Güell *et al.*, 2009[418]	Transcription unit structure
Gupta *et al.*, 2007[280]	N-terminal methionine cleavage
KEGG[113]	Genome annotation, orthology
Kerner *et al.*, 2005[389]	Chaperone substrates
Krause *et al.*, 2004[409]	Terminal organelle assembly
Lindahl *et al.*, 2000[462]	DNA damage
Morowitz *et al.*, 1962[870]	Cell chemical composition
NCBI Gene[61,777]	Genome annotation
Neidhardt *et al.*, 1990[393]	Cell chemical composition
Peil, 2009[105]	RNA modification
PubChem[587]	Metabolite structures
SABIO-RK[100]	Reaction kinetics
Solabia[754–759]	Media chemical composition
Suthers *et al.*, 2009[610]	Metabolic reactions
UniProt[96]	Genome annotation
Weiner *et al.*, 2000[411]	Promoters
Weiner *et al.*, 2003[569]	mRNA expression

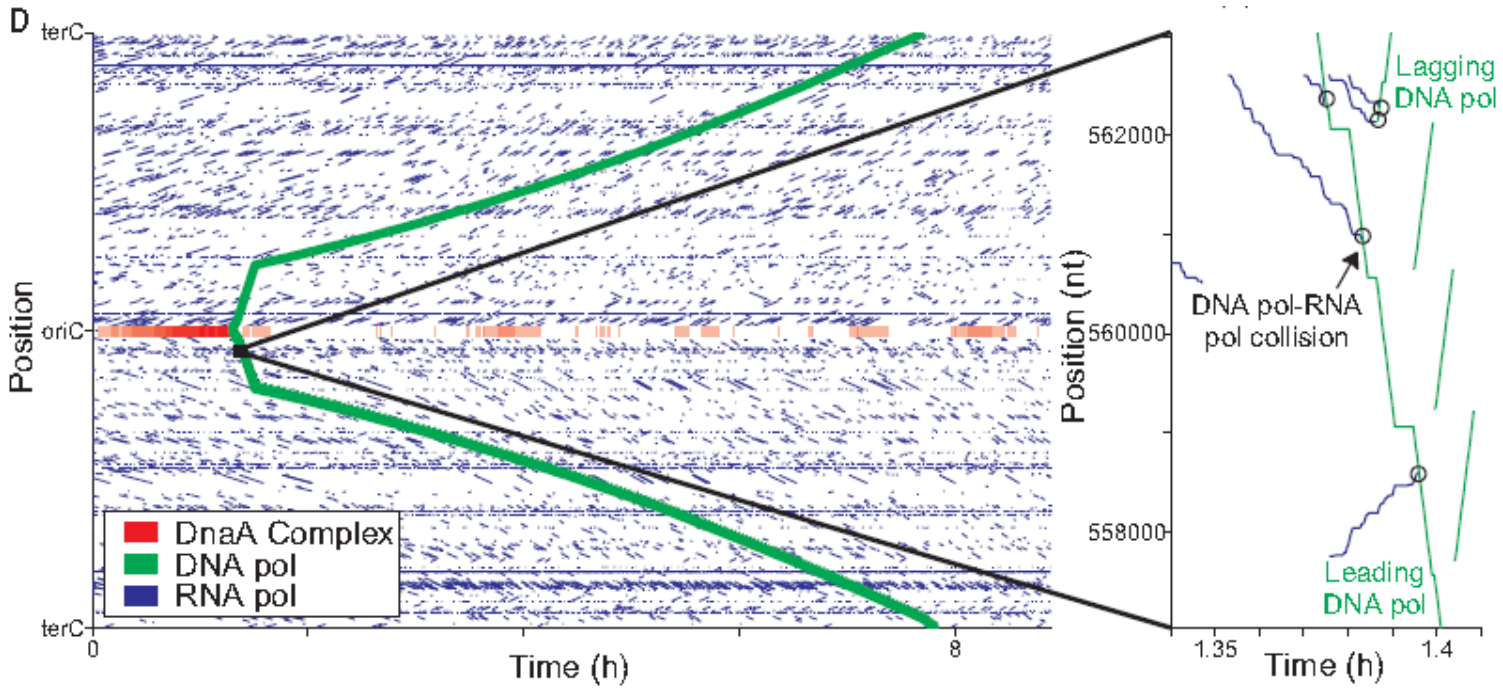
Growth of virtual cell culture



Growth of three cultures (dilutions indicated by shade of blue) and a blank control measured by OD550 of the pH indicator phenol red. The doubling time, t, was calculated using the equation at the top left from the additional time required by more dilute cultures to reach the same OD550 (black lines).

The model calculations were consistent with the observed doubling time!

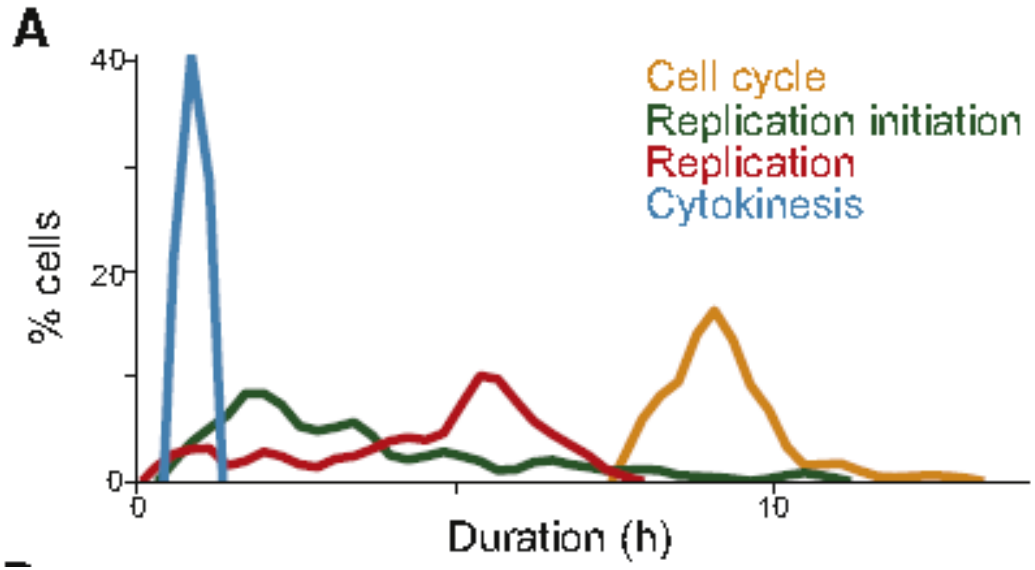
DNA-binding and dissociation dynamics



DNA-binding and dissociation dynamics of the oriC DnaA complex (red) and of RNA (blue) and DNA (green) polymerases for one in silico cell. The oriC DnaA complex recruits DNA polymerase to the oriC to initiate replication, which in turn dissolves the oriC DnaA complex. RNA polymerase traces (blue line segments) indicate individual transcription events. The height, length, and slope of each trace represent the transcript length, transcription duration, and transcript elongation rate, respectively.

Inset : several predicted collisions between DNA and RNA polymerases that lead to the displacement of RNA polymerases and incomplete transcripts.

Predictions for cell-cycle regulation

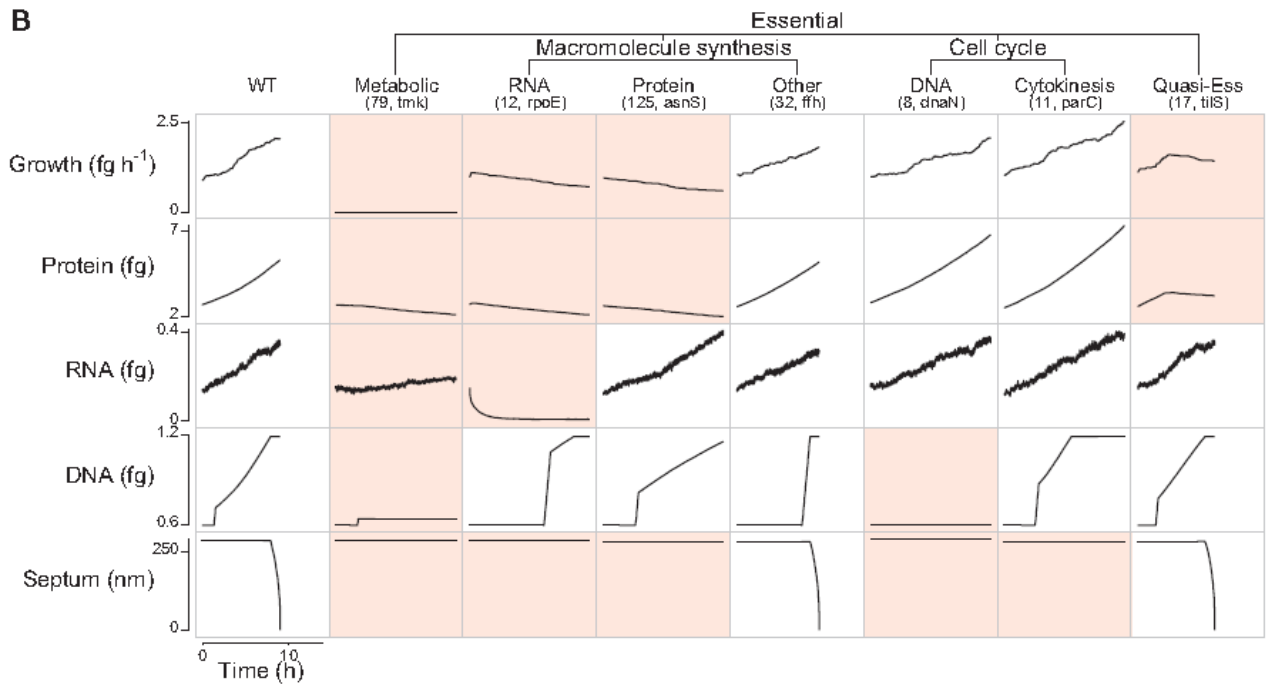


Distributions of the duration of three cell-cycle phases, as well as that of the total cell-cycle length, across 128 simulations.

There was relatively more cell-to-cell variation in the durations of the replication initiation (64.3%) and replication (38.5%) stages than in cytokinesis (4.4%) or the overall cell cycle (9.4%).

This data raised two questions:
(1) what is the source of duration variability in the initiation and replication phases; and
(2) why is the overall cell-cycle duration less varied than either of these phases?

Single-gene knockouts : essential vs. non-essential genes



Single-gene disruption strains grouped into phenotypic classes (columns) according to their capacity to grow, synthesize protein, RNA, and DNA, and divide (indicated by septum length).

Each column depicts the temporal dynamics of one representative in silico cell of each essential disruption strain class.

Dynamics significantly different from wild-type are highlighted in red.

The identity of the representative cell and the number of disruption strains in each category are indicated in parenthesis.

Literature

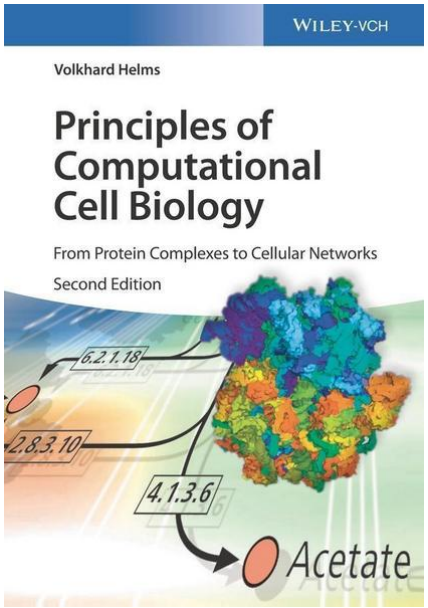
Lecture **slides** — available before the lecture

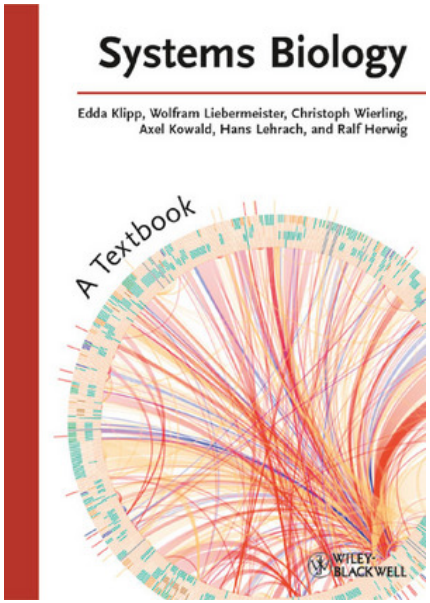
Suggested **reading**

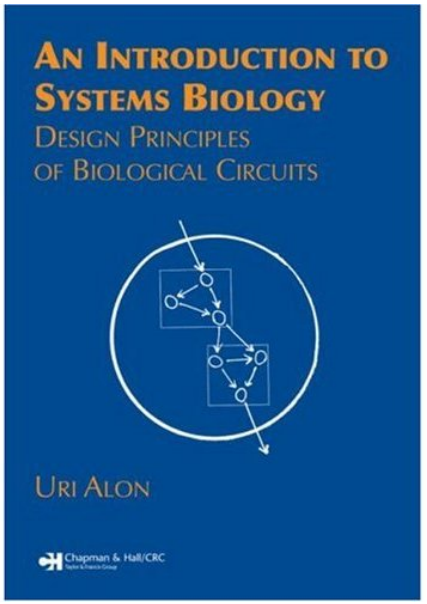
=> check our web page

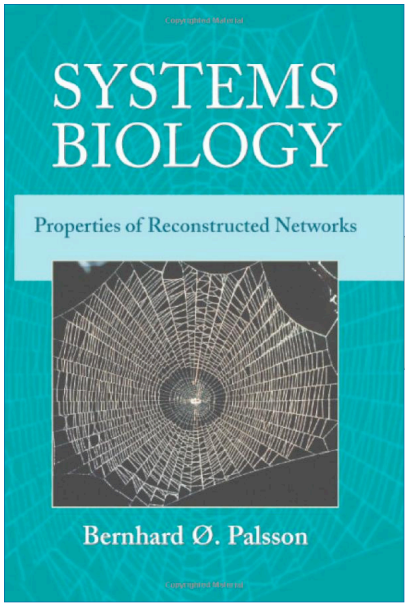
http://gepard.bioinformatik.uni-saarland.de/teaching/…

Textbooks









⇒ check computer science library. For Helms book:

https://ebookcentral.proquest.com/lib/sulb/detail.action?docID=5613486

How to pass this course

Schein = you need to qualify for the **final exam** and pass it

Final exam:

written test of 180 min length about selected parts of the lecture
(slides will be defined 2 weeks before exam) AND about selected assignments

requirements for participation in final exam:

- 50% of the points from the **assignments**
- one assignment task presented @ blackboard in tutorial

Final exam will take place at the end of the semester.

In case you are sick (final exam) you should submit a medical certificate

to take the written re-exam (then this will be counted as first exam).

Re-exam: will take place in first week of the summer term 2020.
Everybody can take the re-exam (first exam failed or passed).

Assignments

Tutors: Andreas Denger, Markus Hollander, Nicolas Künzel,
Pratiti Bhadra, Thorsten Will

Tutorial: Mon, 12:15–13:45, E2 1, room 007

10 assignments with 100 points each

Assignments are part of the course material (not everything is covered in lecture)

=> **one** solution for **two** students (or one)

=> hand-written or one **printable** PDF/PS file per email

=> content: data analysis + interpretation — **think!**

=> no 100% solutions required!!!

=> attach the source code of the programs for checking (no suppl. data)

=> present one task at the **blackboard**

Hand in at the following Fri electronically until 13:15 or
printed at the start of the lecture.

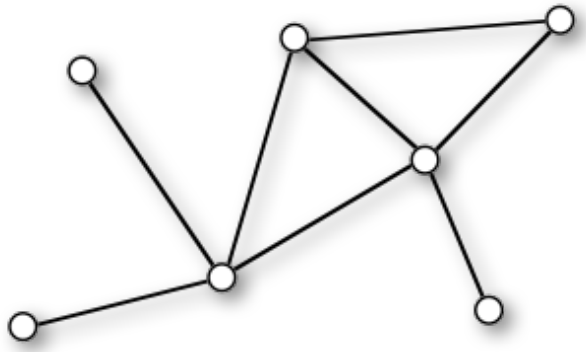
Some Graph Basics

Network <=> Graph

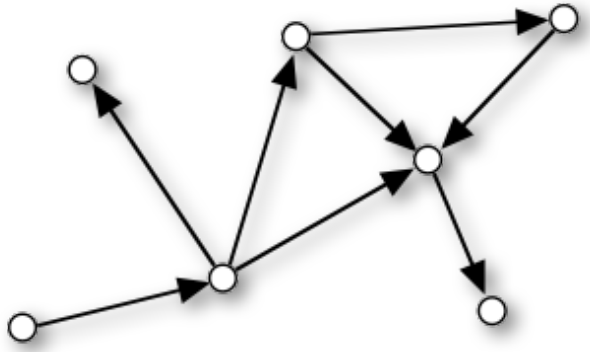
Formal **definition**:

A **graph** *G* is an ordered pair (*V*, *E*) of a set *V* of **vertices** and a set *E* of **edges**.

$$G = (V, E)$$



undirected graph



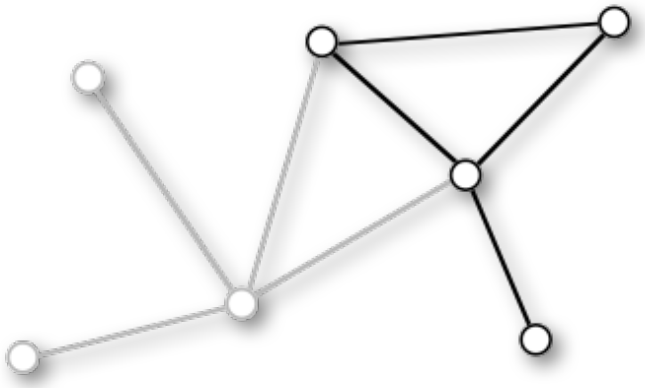
directed graph

If $E = V^{(2)}$ => fully connected graph

Graph Basics II

Subgraph:

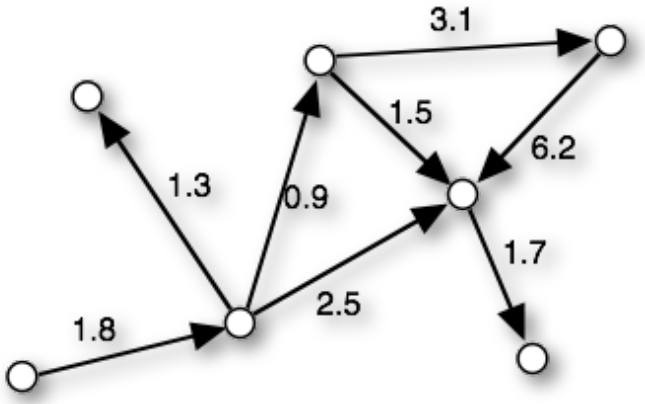
$G' = (V', E')$ is a subset of $G = (V, E)$



Practical question: how to define useful subgraphs?

Weighted graph:

Weights assigned to the edges



Note: no weights for vertices

Walk the Graph

Path = sequence of connected vertices

start vertex => internal vertices => end vertex

Two paths are **independent** (internally vertex-disjoint),

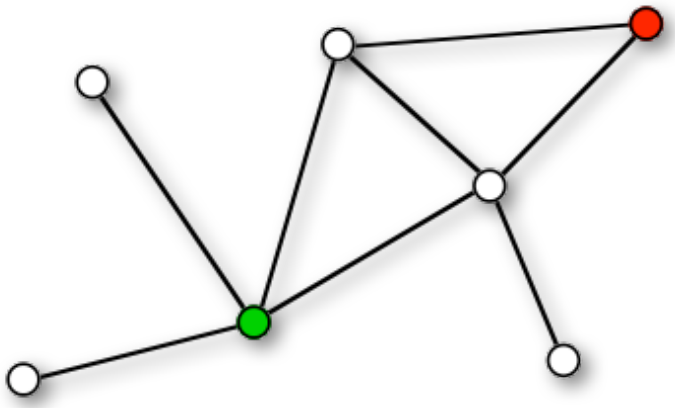
if they have no internal vertices in common.

Vertices *u* and *v* are **connected**, if there exists a path from *u* to *v*.

otherwise: disconnected

Trail = path, in which all edges are distinct

Length of a path = number of vertices || sum of the edge weights



How many paths connect the green to the red vertex?

How long are the shortest paths?

Find the four trails from the green to the red vertex.

How many of them are independent?

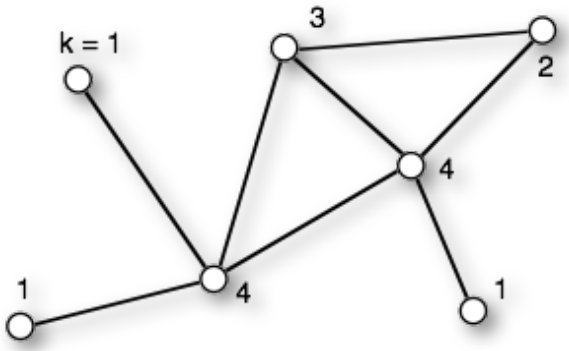
Local Connectivity: Degree/Degree Distribution

Degree *k* of a vertex = number of edges at this vertex

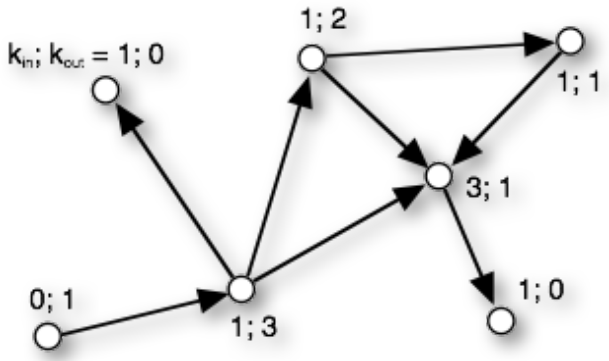
Directed graph => distinguish k_{in} and k_{out}

Degree distribution *P(k)* = fraction of nodes with *k* connections

$$P(k) = \frac{n_k}{N}$$



k	0	1	2	3	4
P(k)	0	3/7	1/7	1/7	2/7



k	0	1	2	3
$P(k_{in})$	1/7	5/7	0	1/7
$P(k_{out})$	2/7	3/7	1/7	1/7

Graph Representation e.g. by adjacency matrix

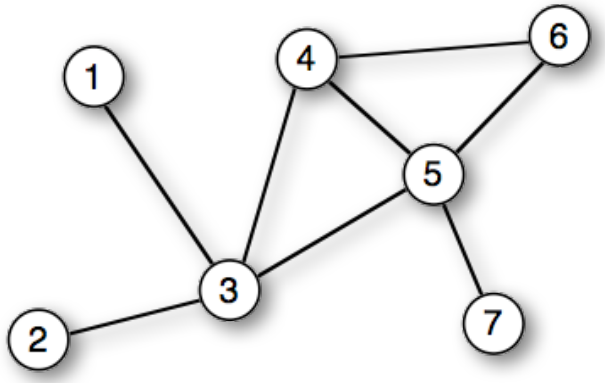
Adjacency matrix is a N x N matrix
with entries M_{uv}

M_{uv} = weight when edge between u and v exists,
0 otherwise

→ symmetric for undirected graphs

+ fast *O(1)* lookup of edges
– large memory requirements
– adding or removing nodes is expensive

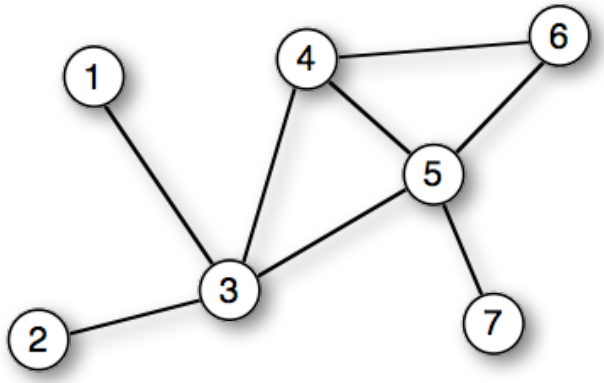
Note: very convenient in programming languages that support sparse multi-dimensional arrays
=> Perl



	1	2	3	4	5	6	7
1	–	0	1	0	0	0	0
2	0	–	1	0	0	0	0
3	1	1	–	1	1	0	0
4	0	0	1	–	1	1	0
5	0	0	1	1	–	1	1
6	0	0	0	1	1	–	0
7	0	0	0	0	1	0	–

Applications of the adjacency matrix

Finding paths between nodes



if a path of length 1 exists from one vertex to another (ie. the 2 vertices are adjacent), there must be an entry of 1 in the corresponding position in the matrix.

e.g. from vertex 5, one can reach vertices 3, 4, 6 and 7 via a path of length 1.

How about paths of length 2?

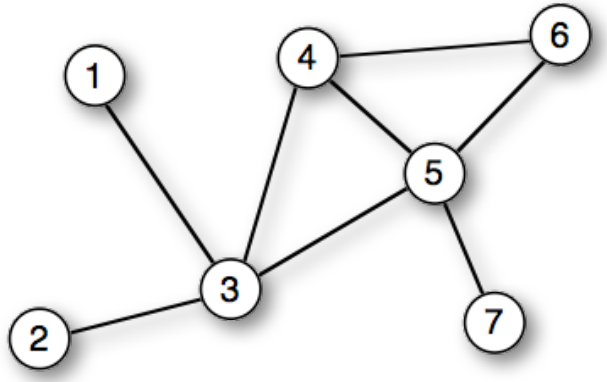
	1	2	3	4	5	6	7
1	–	0	1	0	0	0	0
2	0	–	1	0	0	0	0
3	1	1	–	1	1	0	0
4	0	0	1	–	1	1	0
5	0	0	**1**	**1**	–	**1**	**1**
6	0	0	0	1	1	–	0
7	0	0	0	0	1	0	–

Applications of the adjacency matrix

Fill up the diagonal with 0 values

and multiply the matrix with itself

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$



$$= \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 4 & 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 3 & 1 & 1 & 1 \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{2} & 4 & \mathbf{1} & \mathbf{0} \\ 0 & 0 & 2 & 1 & 1 & 2 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}$$

From the matrix product, you can read off whether 2 nodes are connected by a **path of length 2.**

E.g. from node 5, one can reach all other nodes except node 7.

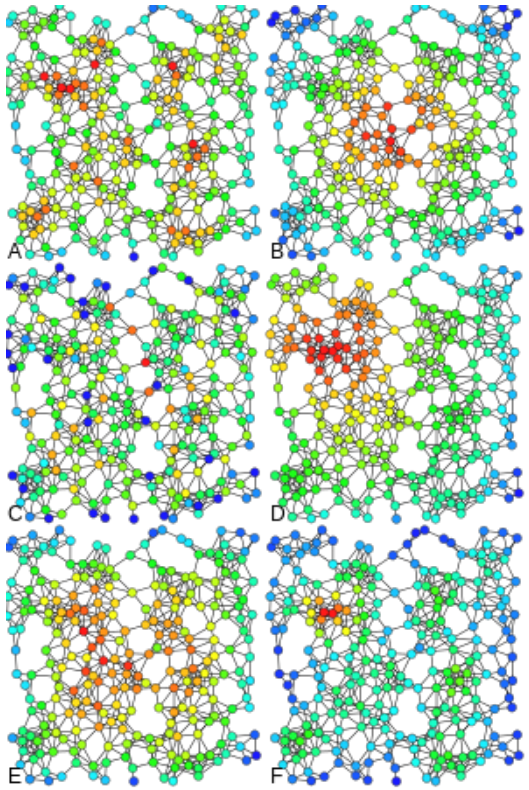
The entries give the multiplicity. E.g. there are two alternative paths to reach node 4 from node 5.

What are the diagonal entries?

Measures and Metrics

"Which are the most important or central vertices in a network?"

Examples of
A) Degree centrality,



B) Closeness centrality,

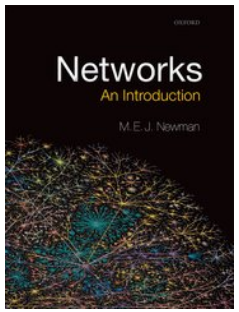
C) Betweenness centrality,

D) Eigenvector centrality,

E) Katz centrality,

F) Alpha centrality of the same graph.

www.wikipedia.org



book by Mark Newman / Oxford Univ Press

- Chapter 7: measures and metrics
- Chapter 11: matrix algorithms and graph partitioning

Degree centrality

Perhaps the simplest centrality measure in a network is the **degree centrality** that is simply equal to the **degree** of each vertex.

E.g. in a **social network**, individuals that have many connections to others might have

- more **influence**,
- more **access to information**,
- or more **prestige** than those individuals who have fewer connections.

Towards Eigenvector Centrality

Let us start by defining the **centrality** of vertex x_i as the sum of the centralities of all its neighbors:

$$x_i' = \sum_j A_{ij} x_j$$

where A_{ij} is an element of the adjacency matrix.
(This equation system must be solved recursively until convergence.)

Remember the multiplication of a matrix with a vector below …

$$\mathbf{A} = \begin{pmatrix} a & b & c \\ p & q & r \\ u & v & w \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} x \\ y \\ z \end{pmatrix},$$

$$\mathbf{AB} = \begin{pmatrix} a & b & c \\ p & q & r \\ u & v & w \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} ax + by + cz \\ px + qy + rz \\ ux + vy + wz \end{pmatrix},$$

en.wikipedia.org

Towards Eigenvector Centrality

Let us start by defining the **centrality** of vertex x_i as the sum of the centralities of all its neighbors:

$$x_i' = \sum_j A_{ij} x_j$$

where A_{ij} is an element of the adjacency matrix.

We can also write this expression in matrix notation as

x' = A x where **x** is the vector with elements x_i .

Repeating this process to make better estimates gives after *t* steps the following vector of centralities:

$$\mathbf{x}(t) = \mathbf{A}^t \mathbf{x}(0)$$

Eigenvector Centrality

Now let us write $\mathbf{x}(0)$ as a linear combination of the eigenvectors $\mathbf{v}_i$ of the (quadratic) adjacency matrix[1]

$$\mathbf{x}(0) = \sum_i c_i \mathbf{v}_i \quad \text{with suitable constants } c_i$$

Then

$$\mathbf{x}(t) = A^t \sum_i c_i \mathbf{v}_i$$

Because $\mathbf{v}_i$ are eigenvectors of A, A $\mathbf{v}_i = k_i \mathbf{v}_i$ with the eigenvalue k_i .

$$\mathbf{x}(t) = A^t \sum_i c_i \mathbf{v}_i = \sum_i c_i \, {k_i}^t \, \mathbf{v}_i = {k_1}^t \sum_i c_i \left[\frac{k_i}{k_1}\right]^t \mathbf{v}_i$$

Here, we divide by the first (and largest) eigenvector k_1 and multiply by k_1 in the front.

Since $k_i \; / \, k_1 \; < 1$ for all $i \neq j$, all terms in the sum decay exponentially as t becomes large. Only the term with $i = j$ remains unchanged.

In the limit $t \rightarrow \infty$, we get for the centrality vector $\mathbf{x}(t) = c_1 \, {k_1}^t \, \mathbf{v}_1$

[1] Remember from linear algebra that a quadratic matrix with full rank can be diagonalized.

Eigenvector Centrality

The limiting vector of the eigenvector centralities is simply proportional to the leading eigenvector of the adjacency matrix.

Equivalently, we could say that the centrality **x** satisfies

$$\mathbf{A}\,\mathbf{x} = k_1\,\mathbf{x}$$

This is the **eigenvector centrality** first proposed by Bonacich (1987).

The centrality x_i of vertex i is proportional to the sum of the centralities of its neighbors:

$$x_i = {k_1}^{-1} \sum_j A_{ij} x_j$$

Divide above eq. by k_1

This has the nice property that the centrality can be large either because a vertex has many neighbors or because it has important neighbors with high centralities (or both).

Problems of the Eigenvector Centrality

The eigenvector centrality works best for undirected networks.

For directed networks, certain complications can arise.

In the figure on the right, vertex A will have eigenvector centrality zero.

Hence, vertex B will also have centrality zero (because A is the only neighbor that points at it).

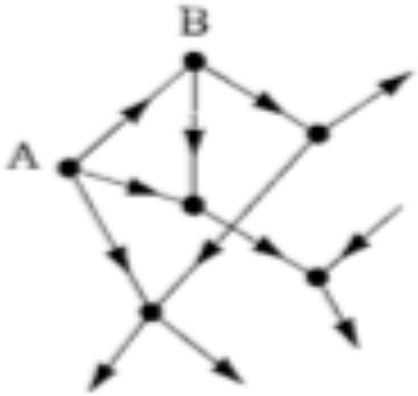


Figure 7.1: A portion of a directed network. Vertex A in this network has only outgoing edges and hence will have eigenvector centrality zero. Vertex B has outgoing edges and one ingoing edge, but the ingoing one originates at A, and hence vertex B will also have centrality zero.

Katz Centrality

One solution to the issues of the Eigenvector Centrality is the following:

We simply give each vertex a small amount of centrality "for free",
regardless of its position in the network or the centrality of its neighbors.

→ we define $x_i = \alpha \sum_j A_{ij} x_j + \beta$ where α and β are positive constants.

In matrix terms, this can be written as $\mathbf{x} = \alpha\mathbf{A}\mathbf{x} + \beta\ \mathbf{1}$

where $\mathbf{1}$ is the vector $(1,1,1,\ldots)^T$. By rearranging for $\mathbf{x}$ we find

$\mathbf{1}\,\mathbf{x} - \alpha\,\mathbf{A}\,\mathbf{x} = \beta\,\mathbf{1}$ (where we used $\mathbf{1}\,\mathbf{x} = \mathbf{x}$)

$(\mathbf{1} - \alpha\,\mathbf{A})\,\mathbf{x} = \beta\,\mathbf{1}$

$(\mathbf{1} - \alpha\,\mathbf{A})^{-1}\,(\mathbf{1} - \alpha\,\mathbf{A})\,\mathbf{x} = (\mathbf{1} - \alpha\,\mathbf{A})^{-1}\,\beta\,\mathbf{1}$ (multiply both sides with $(1 - \alpha A)^{-1}$)

$\mathbf{x} = \beta\,(\mathbf{1} - \alpha\,\mathbf{A})^{-1}\,\mathbf{1}$

When setting $\beta = 1$, we get the **Katz centrality** (1953) $\mathbf{x} = (1 - \alpha\,\mathbf{A})^{-1}\,\mathbf{1}$

Computing the Katz Centrality

The Katz centrality differs from the ordinary eigenvector centrality by having a **free parameter** α, which governs the balance between the eigenvector term and the constant term.

However, inverting a matrix on a computer has a complexity of $O(n^3)$ for a graph with n vertices.

This becomes prohibitively expensive for networks with more than 1000 nodes or so.

It is more efficient to make an initial guess of x and then repeat

$$\mathbf{x'} = \alpha \mathbf{A x} + \beta \mathbf{1}$$

many times. This will converge to a value close to the correct centrality.

A good test for convergence is to make two different initial guesses and run this until the resulting centrality vectors agree within some small threshold.

Towards PageRank

The Katz centrality also has one feature that can be **undesirable**.

If a vertex with high Katz centrality has edges pointing to many other vertices,
then all those vertices also get high centrality.

E.g. if a Wikipedia page points to my webpage,
my webpage will get a centrality comparable to Wikipedia!

But Wikipedia of course also points to many other websites,
so that its contribution to my webpage "should" be relatively small
because my page is only one of millions of others.

-> we will define a variation of the Katz centrality in which the
centrality I derive from my network neighbors is proportional
to their centrality divided by their out-degree.

PageRank

This weighted centrality is defined by

$$x_i = \alpha \sum_j A_{ij} \frac{x_j}{k_j^{out}} + \beta$$

If the network contains vertices with zero outdegree,
this can be fixed by setting $k_j^{out} = 1$ for all such vertices.

In matrix terms, this equation becomes

$$\mathbf{x} = \alpha\, \mathbf{A}\, \mathbf{D}^{-1}\, \mathbf{x} + \beta\, \mathbf{1}$$

where $\mathbf{1}$ is the vector $(1,1,1,\ldots)^T$ and $\mathbf{D}$ the diagonal matrix with $D_{jj} = max(k_j^{out}, 1)$

PageRank

By rearranging we find that

$$\mathbf{x} = \beta\,(\mathbf{I} - \alpha\,\mathbf{A}\,\mathbf{D}^{-1})^{-1}\,\mathbf{I}$$

Because β plays the same unimportant role as before, we will set $\beta = 1$.

Then we get $\mathbf{x} = (\mathbf{I} - \alpha\,\mathbf{A}\,\mathbf{D}^{-1})^{-1}\,\mathbf{I} = \mathbf{D}\,(\mathbf{D} - \alpha\,\mathbf{A})^{-1}\,\mathbf{I}$ expand with D

This centrality measure is commonly known as **PageRank**,
using the term used by Google.

PageRank is one of the ingredients used by Google
to determine the ranking of the answers to your queries.

α is a free parameter and should be chosen less than 1. (Google uses 0.85).

Closeness centrality

An entirely different measure of centrality is provided by the **closeness centrality.**

Suppose d_{ij} is the length of a geodesic path (i.e. the shortest path) from a vertex *i* to another vertex *j*.
Here, length means the number of edges along the path.

Then, the mean **geodesic distance** from *i*, averaged over all vertices *j* in the network is

$$l_i = \frac{1}{n}\sum_j d_{ij}$$

The mean distance l_i is not a centrality measure in the same sense as the other centrality measures.

It gives *low* values for more central vertices and *high* values for less central ones.

Closeness centrality

The inverse of l_i is called the **closeness centrality** C_i

$$C_i = \frac{1}{l_i} = \frac{n}{\sum_j d_{ij}}$$

It has become popular in recent years to rank **film actors**
according to their closeness centrality in the network
of who has appeared in films with who else.

Using data from www.imdb.com the largest component of the network includes
more than 98 % of about half a million actors.

Closeness centrality

The highest closeness centrality of any actor is 0.4143 for Christopher Lee.

The second highest centrality has Donald Pleasence (0.4138).

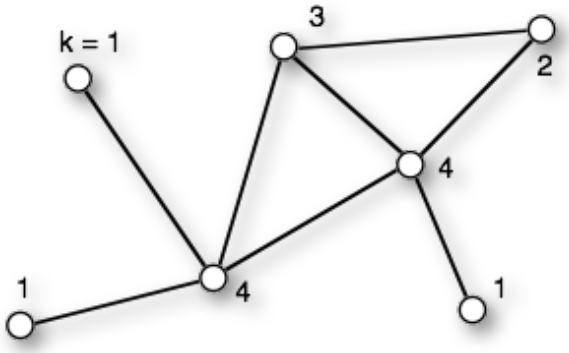
Pictures from wikipedia

Random Graphs

A **graph** *G* is an ordered pair *(V, E)* of a set *V* of **vertices** and a set *E* of **edges**.

Degree distribution *P(k)*

$$P(k) = \frac{n_k}{N}$$



k	0	1	2	3	4
P(k)	0	3/7	1/7	1/7	2/7

Random network:

also called the "Erdös-Renyi model":

- start with set of given nodes
- then add links randomly

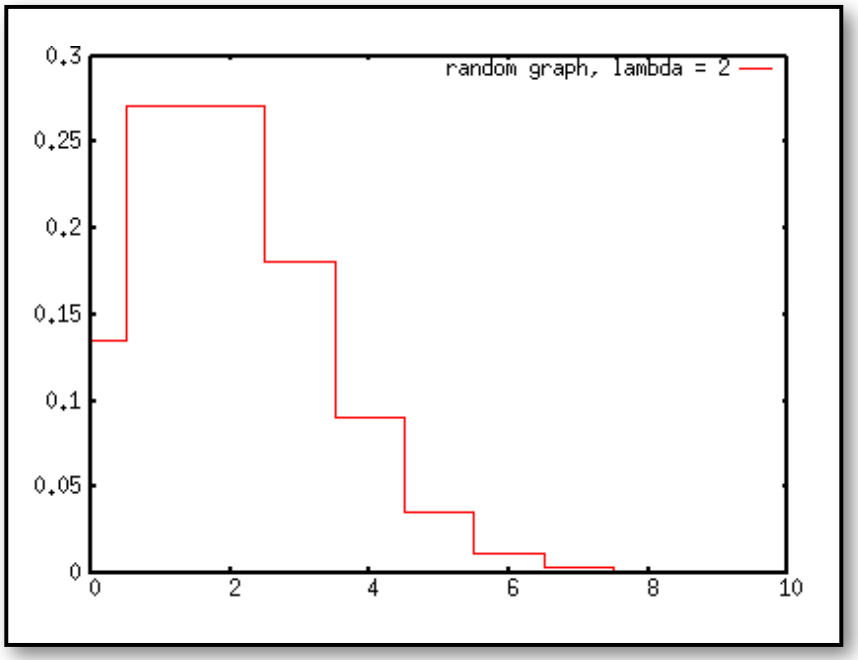
P(k) = "Poisson" (will show this in V2)

$$P(k) = \frac{\lambda^k}{k!} e^{-\lambda}$$

Degree Distribution of Random Network *P(k)*

Many independently placed edges => **Poisson statistics**

$$P(k) = \frac{\lambda^k}{k!} e^{-\lambda}$$



k	*P(k | λ = 2)*
0	0.14
1	0.27
2	0.27
3	0.18
4	0.090
5	0.036
6	0.012
7	0.0034
8	0.00086
9	0.00019
10	3.82e-05

=> Small probability for $k >> \lambda$

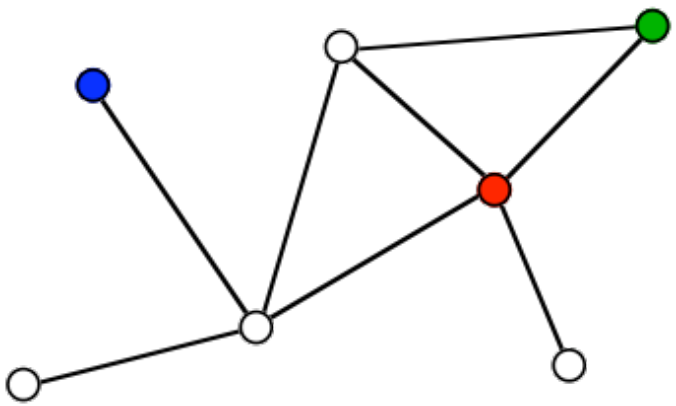
Connectivity of the Neighborhood

How many of the neighboring vertices are themselves neighbors?
=> this is measured by the **clustering coefficient** *C(k)*

Number of possible undirected edges between *k* nodes: $n_{max} = \frac{k\,(k-1)}{2}$

n_k is the actual number of edges between the neighbor nodes.

Fraction of actual edges ≅ **clustering coefficient** $C(k, n_k) = \frac{2n_k}{k\,(k-1)}$



green: $k = 2,\ n_k = 1 \rightarrow C = 1$

red: $k = 4,\ n_k = 2 \rightarrow C = 1/3$

blue: $k = 1,\ n_k = ? \rightarrow$ *C* is not defined

Note: clustering coeff. is sometimes also defined via fraction of possible triangles

Clustering Coefficient of a Graph

Data: C_i for each node i $\rightarrow$ N values

Statistics:

average at **fixed k**

$$\rightarrow \quad C(k) = \frac{1}{n_k} \sum_{k_i = k} C_i$$

average over **all nodes**

$$\rightarrow \quad \langle C \rangle = \frac{1}{N} \sum C_i$$

Note: it is also possible to average the *C(k)*
$\Rightarrow$ This yields a different value for <*C*> !!!
because no weighting is done for different occupancy of k's.

Summary

What you learned **today**:

=> **networks** are everywhere

⇒ how to get the "**Schein**" for BI3

⇒ What is the lecture content

⇒ Adjacency matrix

⇒ How to characterize the centrality of nodes

⇒ How to construct a random graph

Next lectures:

- Random graphs vs. scale-free networks (assignments 1 + 2)
- Structures of protein complexes