4-16 Recitation

EF Lecture #15 and DL Lecture #16

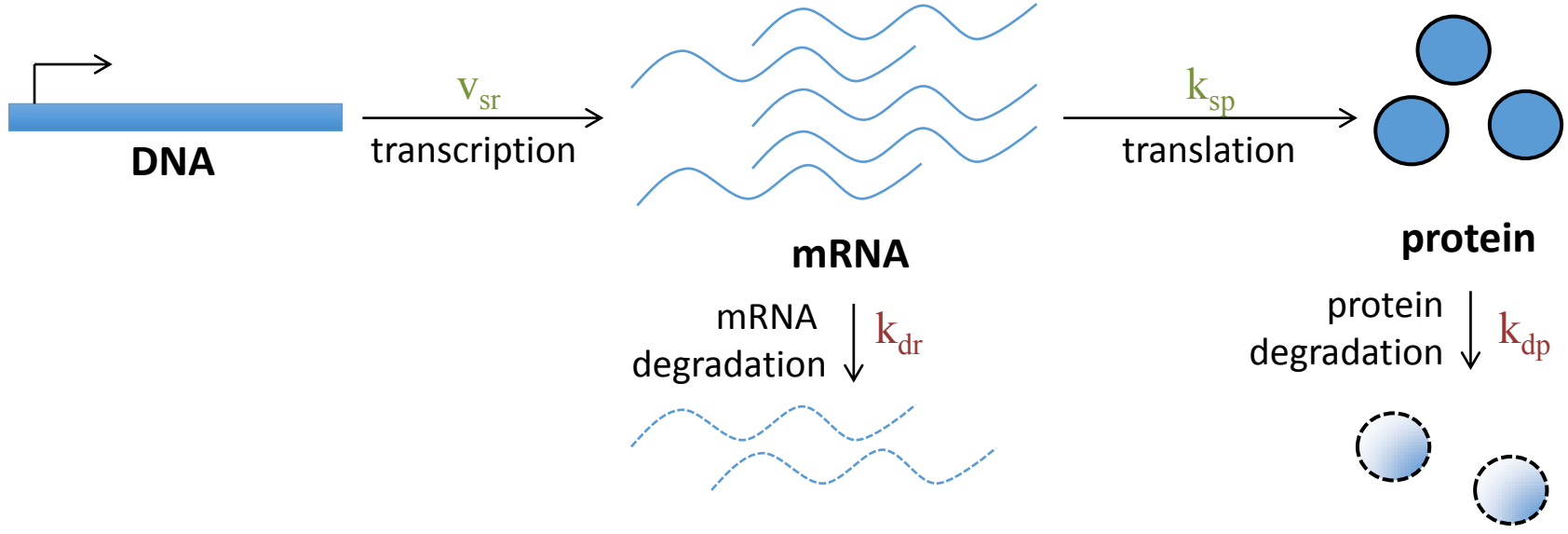
Protein Interaction Networks & Computable Network Models

Reminders

- Project due next Tuesday at midnight!
  - writeup description posted, email us if any questions
- Problem Set #5 (last one!) will be out next Tuesday also
  - due Thurs, May 1st
  - a bit shorter, not a lot of programming

Predicting Protein Levels

Recall the central dogma:

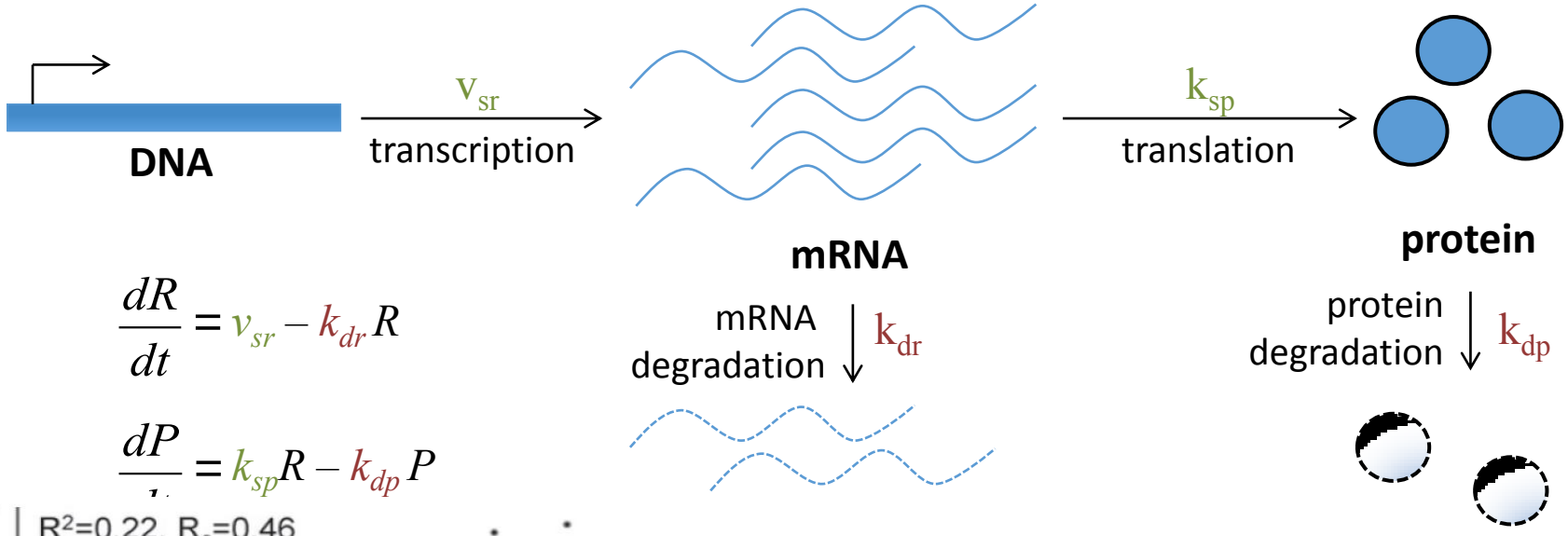


- Let's elaborate a little:
  - assume transcription and translation occur at rates v_{sr} and k_{sp} respectively
  - and mRNAs and proteins also degrade at rates k_{dr} and k_{dp} respectively
  - Then we can describe how R = mRNA conc. and P = protein conc. change over time using differential equations:

$$\frac{dR}{dt} = v_{sr} - k_{dr}R \qquad \frac{dP}{dt} = k_{sp}R - k_{dp}P$$

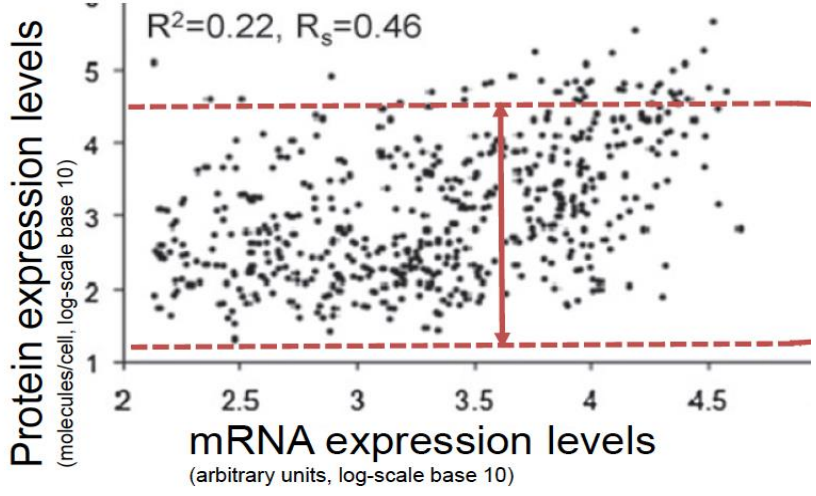
Predicting Protein Levels

Recall the central dogma:



$$\frac{dR}{dt} = v_{sr} - k_{dr} R$$

$$\frac{dP}{dt} = k_{sp} R - k_{dp} P$$

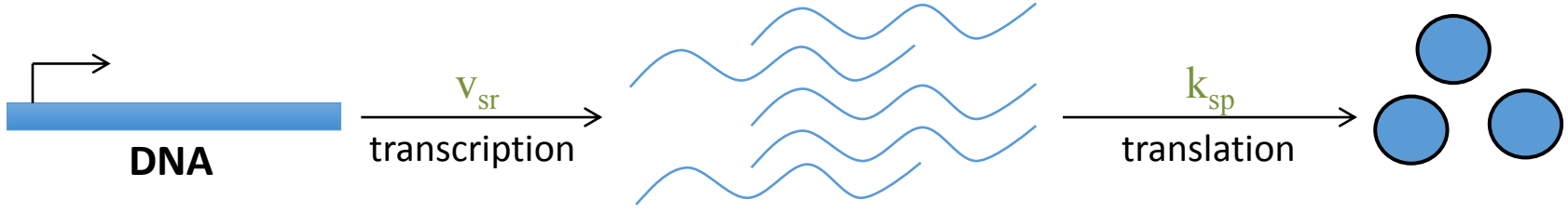


Experimentally, it is much easier to get mRNA levels (microarray, RNA-seq) than protein levels (2D electrophoresis, mass-spec), so we often use gene expression as an approximation of protein levels

However, mRNA levels alone are generally poor predictors of protein levels, since they fail to account for either translation rates or protein degradation rates

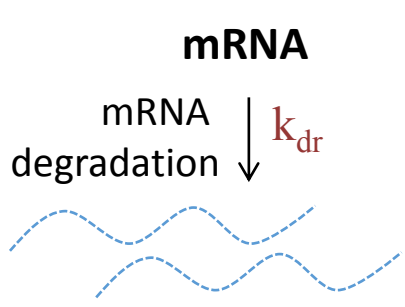
Predicting Protein Levels

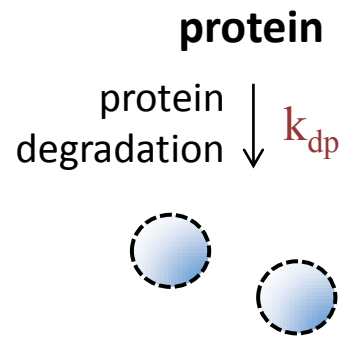
Recall the central dogma:

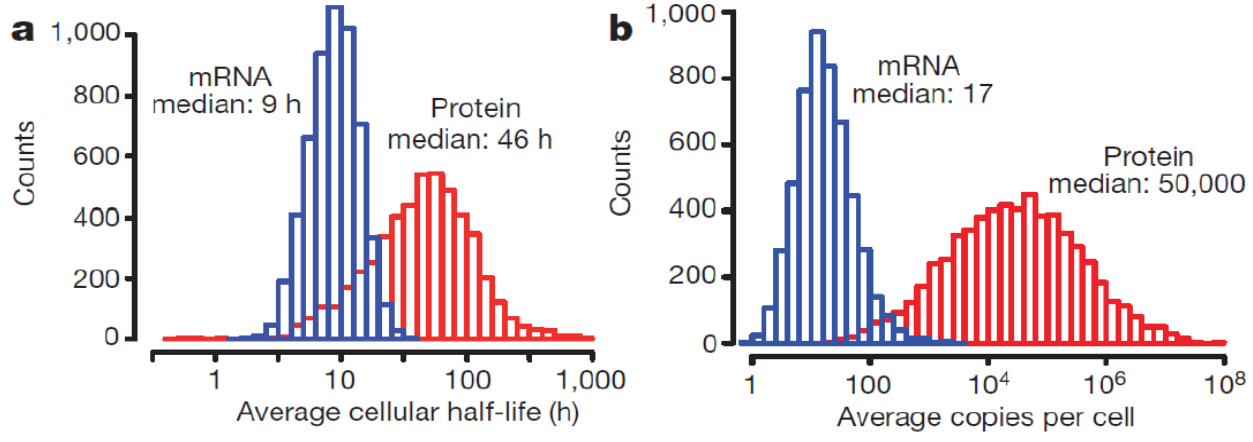


$$\frac{dR}{dt} = v_{sr} - k_{dr} R$$

$$\frac{dP}{dt} = k_{sp} R - k_{dp} P$$

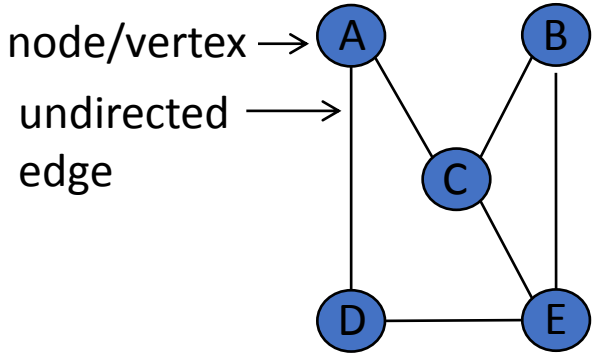




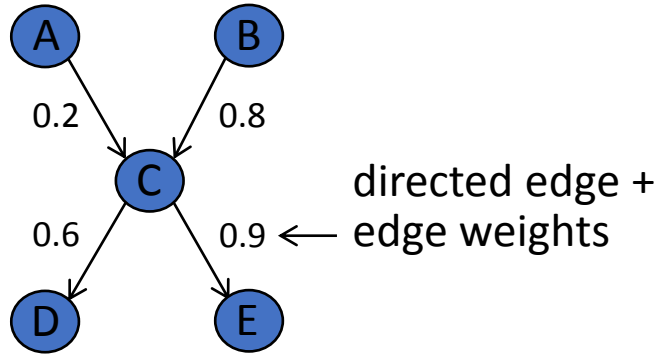


Also, proteins tend to be more stable than RNA and to stick around in the cell longer, so the total amount of protein depends on the amount of RNA over a long period of time

Graph Terminology



Graph G_1:
undirected



Graph G_2:
directed

A graph G is specified as a list of nodes/vertices V and a list of edges E, so G = (V,E)

- G_1 and G_2 both have V = [A,B,C,D,E]
- G_1 has undirected edges E = [(A,C),(A,D),(C,B),(B,E),(E,C),(D,E)] = 6 edges
- G_2 has directed edges, so must give (parent,child) lists: E = [(A,C),(B,C),(C,D),(C,E)]

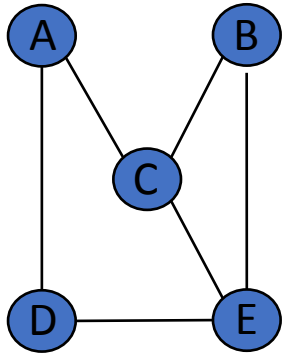
Degree(vertex) = # of edges incident on vertex, can split into in and out-degree if directed

- in G_1, Degree(C) = 3
- in G_2, vertex C has in-degree 2 and out-degree 2

Paths between two vertices have length = sum of edge weights (= # of edges if no weights)

- in G_1, one path from A to E is <A,C,E> and is of length 2.
- in G_2, path from A to E is also <A,C,E> but is of length 0.2 + 0.9 = 1.1

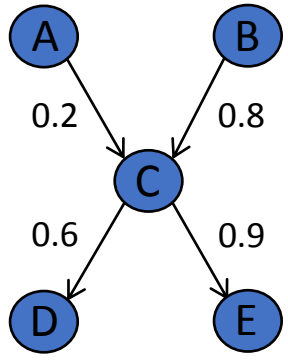
Adjacency Matrix



Graph G_1:
undirected

The adjacency matrix A of a graph with N nodes is an NxN matrix where A_{ij} = the weight of edge from node *i* to node *j* in the graph if it exists, and 0 if there is no edge

- symmetric if graph is undirected



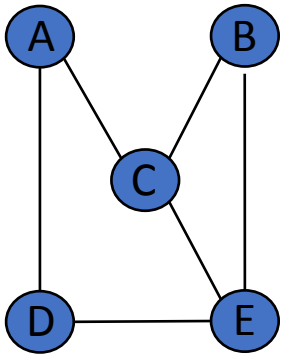
Graph G_2:
directed

Adjacency matrix for G_1

i *j*	A	B	C	D	E
A	0	0	1	1	0
B	0	0	1	0	1
C	1	1	0	0	1
D	1	0	0	0	1
E	0	1	1	1	0

Adjacency matrix for G_2

i *j*	A	B	C	D	E
A	0	0	0.2	0	0
B	0	0	0.8	0	0
C	0	0	0	0.6	0.9
D	0	0	0	0	0
E	0	0	0	0	0



Adjacency Matrix

Let A^k be adjacency matrix A raised to the *k*th power where A_{ij} is either 1 or 0 (e.g. does not take edge weights into account). Then the *ij*th entry of A^k corresponds to the number of paths from node *i* to node *j* of length *k*

- note: path length = # of edges traversed

A^1

i *j*	A	B	C	D	E
A	0	0	1	1	0
B	0	0	1	0	1
C	1	1	0	0	1
D	1	0	0	0	1
E	0	1	1	1	0

A^2

i *j*	A	B	C	D	E
A	2	1	0	0	2
B	1	2	1	1	1
C	0	1	3	2	1
D	0	1	2	2	0
E	2	1	1	0	3

A^3

i *j*	A	B	C	D	E
A	0	2	5	4	1
B	2	2	4	2	4
C	5	4	2	1	6
D	4	2	1	0	5
E	1	4	6	5	2

How many paths of length 2 are there from A to E?

A^2_{AE} = 2 so there are 2 paths of length 2 from A to E: A-C-E and A-D-E

How many paths of length 3 are there from A to E?

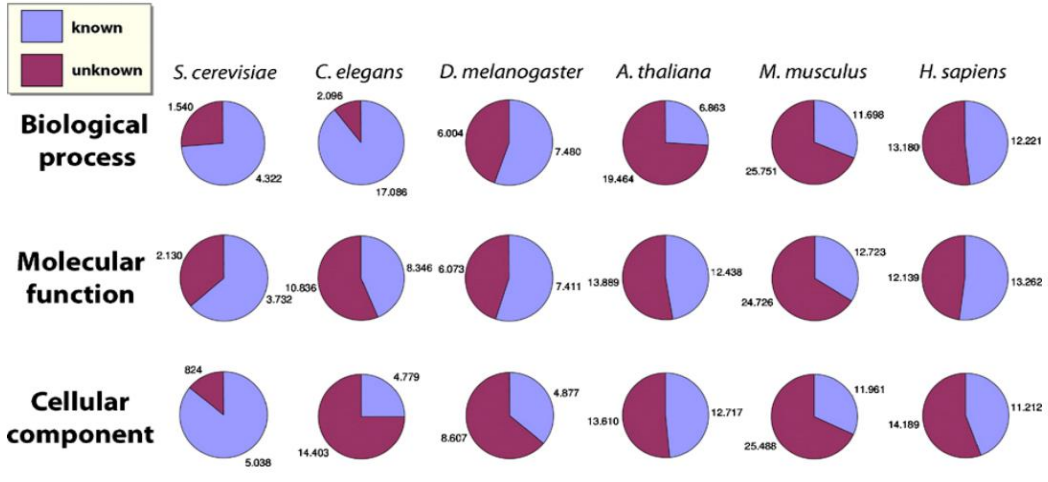
A^3_{AE} = 1 so there is 1 path of length 3 from A to E: A-C-B-E

How many paths of length 3 are there from A to D?

A^3_{AD} = 4 so 4 paths: A-D-E-D, A-C-E-D, A-C-A-D, A-D-A-D

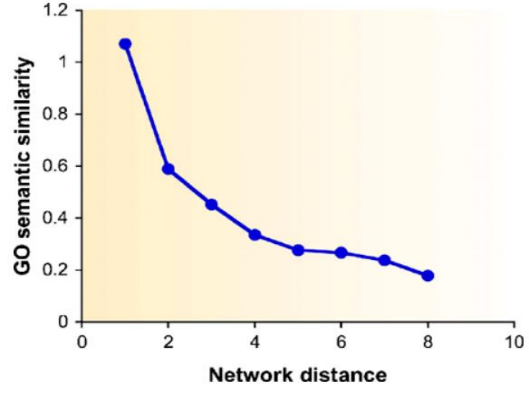
Network Models in Biology

- Many types of large (hairball) networks
  - PPI, genetic interactions, expression networks….
- Many large networks exist today – we can use these to predict function for the large number of functionally unannotated genes
- Today, we will cover algorithms to annotate nodes in these networks



A large number of genes across many organisms still have unknown function

Source: Sharan, Roded, Igor Ulitsky, et al. "Network-based Prediction of Protein Function." *Molecular Systems Biology* 3, no. 1 (2007).



Networks can help us predict function – closeness in a network may indicate functional similarity

Annotating graphs

- Directly annotating unknown nodes
  - K-nearest neighbors
  - Local search
  - Simulated annealing

- Clustering graphs (then annotate each cluster)
  - Betweenness clustering
  - Markov clustering

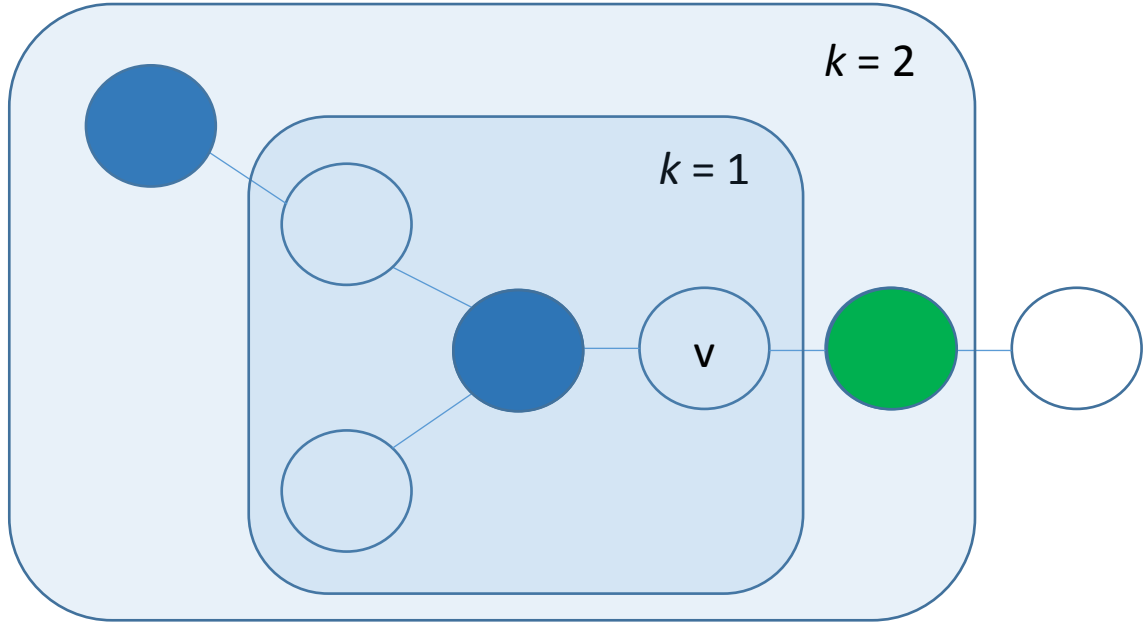
Annotating graphs

- Directly annotating unknown nodes
  - K-nearest neighbors
  - Local search
  - Simulated annealing

- Clustering graphs (then annotate each cluster)
  - Betweenness clustering
  - Markov clustering

K-nearest neighbor approach

- Assumes that a node has related function to its neighbors
- Start with a partially known graph (shaded nodes)
- Assign function based on its neighbors within distance *k* – easy to compute



***k* = 1:** No neighbors are annotated and we are unable to annotate node *u.*

***k* = 2:** Two neighbors are now annotated. We assign the same annotation to node *u*.

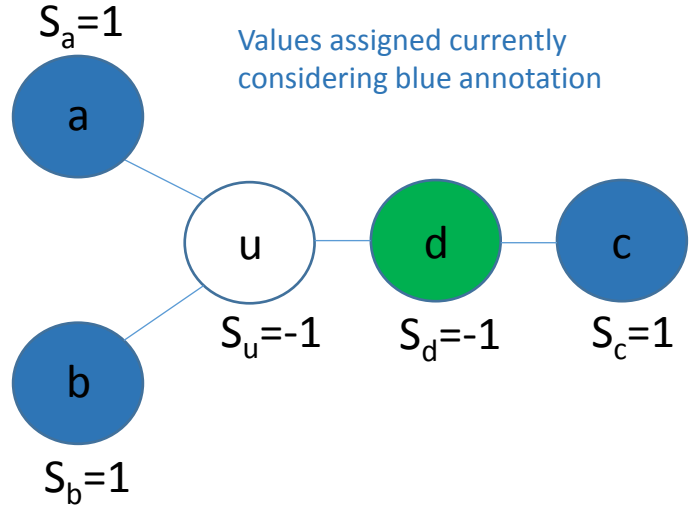
What if nodes have different annotations? Should u and v have the same annotation?

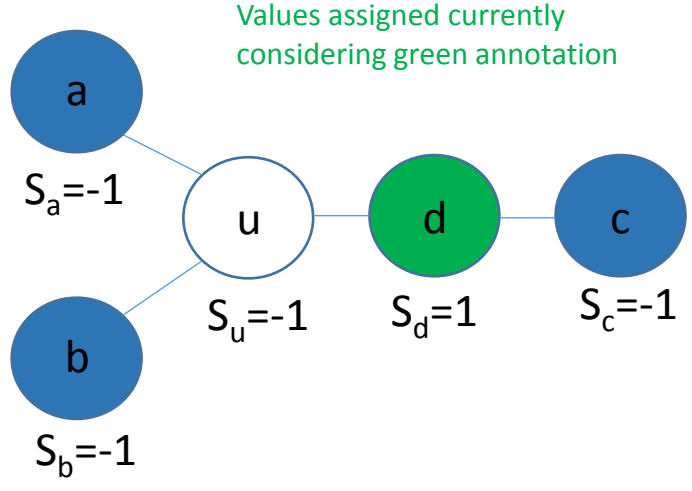
Local Search Algorithm

- Allows us to deal with multiple nearby annotations

- For each annotation (e.g. green, blue):
  - For each node *i*, set S_i=1 if i has that annotation, S_i=-1 otherwise
  - For each unannotated node, set S_i such that it maximizes $\sum S_i S_v$ over all neighbors v

If most neighbors are +1, this will lead to S_i=+1. If most neighbors are -1, this will lead to S_i=-1.



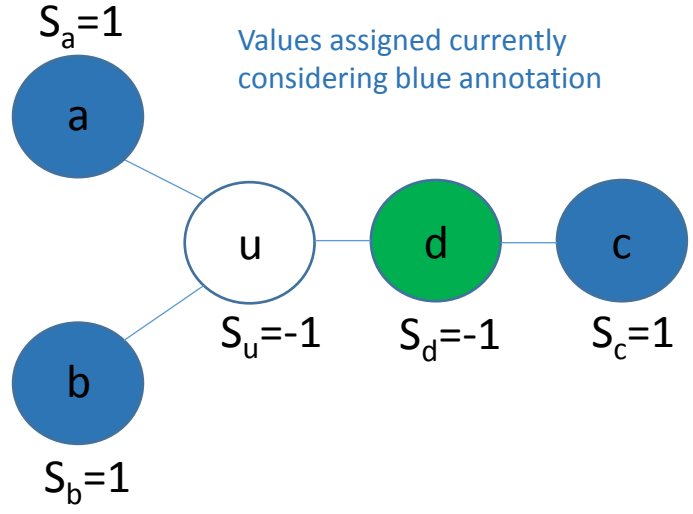


Local Search Algorithm

- Allows us to deal with multiple nearby annotations

- For each annotation (e.g. green, blue):
  - For each node *i*, set S_i=1 if i has that annotation, S_i=-1 otherwise
  - For each unannotated node, set S_i such that it maximizes $\sum S_i S_v$ over all neighbors v

If most neighbors are +1, this will lead to S_i=+1. If most neighbors are -1, this will lead to S_i=-1.

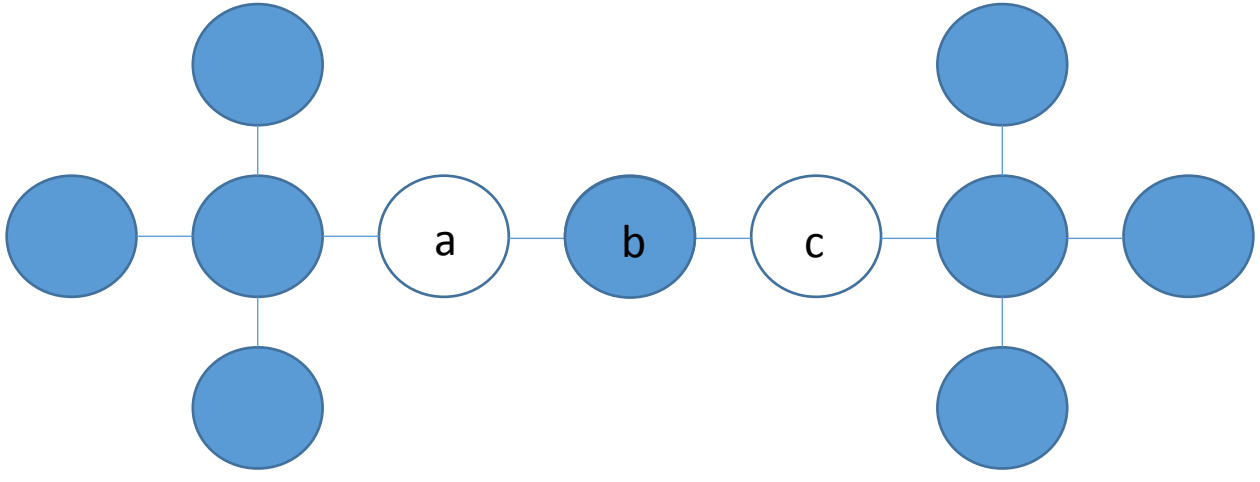


If we want to annotate S_u, we see that it should be blue (+1 from the blue perspective) in order to maximize:

$S_u(S_a+S_b+S_d)$
$=S_u(1+1-1)$
$=S_u$ ➔ S_u=1

Local Search Algorithm

- In complex graphs with lots of unannotated nodes, local search may not find good solutions – local optimization can't find global solution
- For example, if we try to annotate node *b* first, turning it to blue would actually make the score worse, so we would never turn *b* blue



Simulated Annealing

As before, to perform global optimization, we use our usual Simulated Annealing approach with Metropolis Acceptance criterion. If we want a high scoring graph (instead of low energy as in protein structure):

- Initialize with starting T and a graph G with score S
- For some number of iterations:
    - Randomly perturb the network (may add/remove/reweight an edge or include/exclude a vertex), giving G_{test} with score S_{test}
    - If $S_{test} > S$, accept the new graph and update G and S
    - Otherwise, accept the new graph with probability $P=e^{-(S-Stest)/kT}$ and update accordingly
- Lower *T* according to the cooling schedule

Annotating graphs

- Directly annotating unknown nodes
  - K-nearest neighbors
  - Local search
  - Simulated annealing

- Clustering graphs (then annotate each cluster)
  - Betweenness clustering
  - Markov clustering

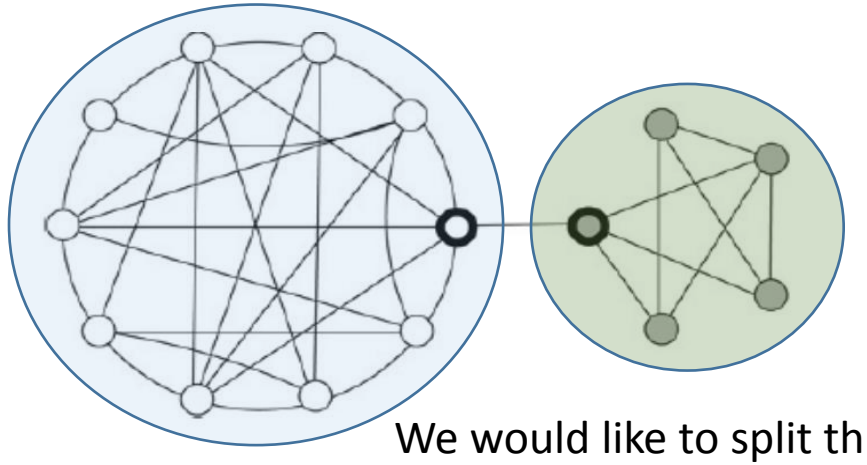
Annotating graphs

- Directly annotating unknown nodes
  - K-nearest neighbors
  - Local search
  - Simulated annealing

- Clustering graphs (then annotate each cluster)
  - Betweenness clustering
  - Markov clustering

Clustering graphs

- Divide large graph into subgraphs, each of which has:
  - Lots of internal connections
  - Few connections to the rest of the graph



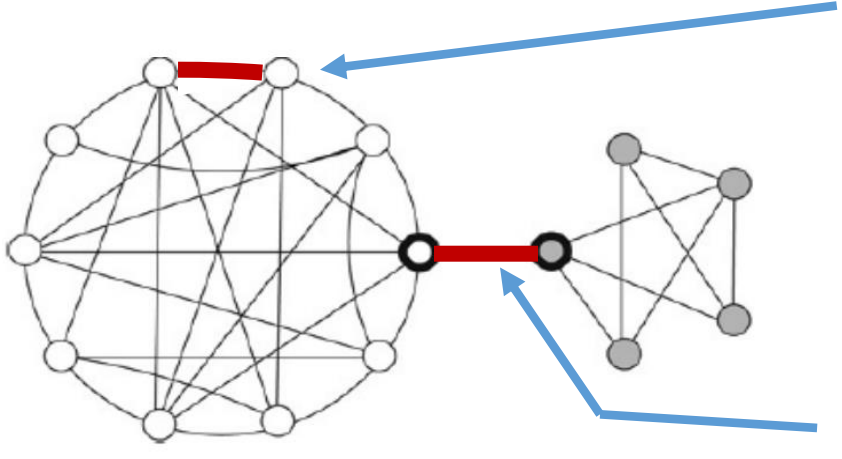
We would like to split this graph into its 2 logical subgraphs.

- Two algorithms
  1. Betweenness clustering
  2. Markov clustering

- Once we have clustered a graph, we can annotate each cluster

Betweenness clustering

- Betweenness – property of edges
  - Number (or summed weight) of shortest paths between all pairs of vertices that pass through that edge
  - Split weight among multiple shortest paths if there are more than 1 for a pair of nodes

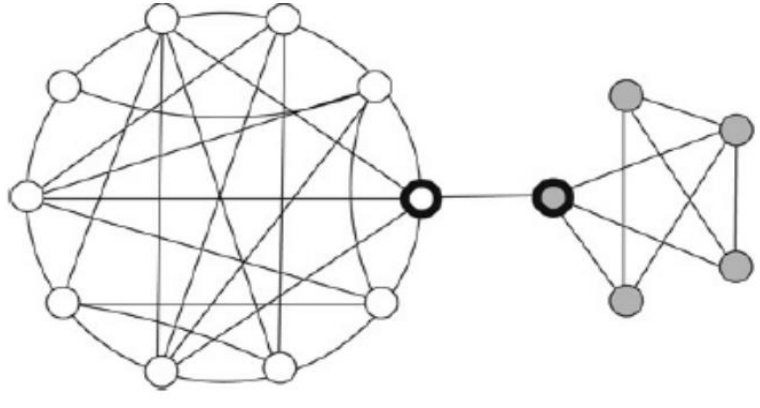


This edge (and others within the respective clusters) have low betweenness. In this particular case, this edge is contained in only one shortest path (the shortest path between the two nodes it connects)

This edge (connecting the two clusters) has high betweenness. This edge is contained in all shortest paths between the left and right subgraphs.

Betweenness clustering algorithm

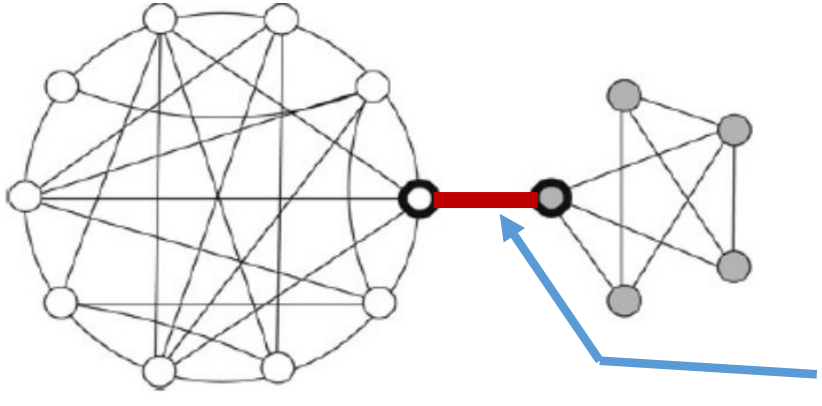
- Precompute shortest paths between all pairs of nodes (Floyd-Warshall/Johnson's algorithm)
- Repeat until max betweenness < threshold:
  - For each edge in the graph:
    - Compute betweenness
  - Remove edge with high betweenness



- As we remove edges with high betweenness, we are breaking the graph into chunks that are more connected internally

Markov clustering

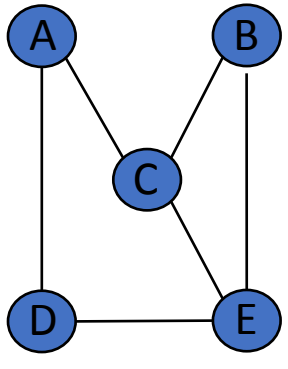
- Intuition: A random walk will spend more time within a cluster than passing between cluster



If we assume all edges are equally weighted in this graph, there is only one way to get from the left subgraph to the right subgraph, while there are many more options (and therefore we are more likely) to stay in a particular subgraph.

Stochastic Matrix

- If we normalize along the rows of the adjacency matrix, we generate a new matrix M (with the same dimensions as A), where the *ij*th entry of M represents the probability of moving from node *i* to node *j* during the random walk

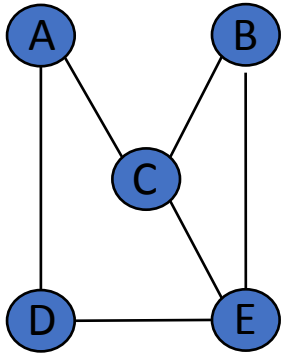


A	A	B	C	D	E
A	0	0	1	1	0
B	0	0	1	0	1
C	1	1	0	0	1
D	1	0	0	0	1
E	0	1	1	1	0

M	A	B	C	D	E
A	0	0	½	½	0
B	0	0	½	0	½
C	.33	.33	0	0	.33
D	½	0	0	0	½
E	0	.33	.33	.33	0

$$\sum_j M_{ij} = 1$$

Stochastic Matrix



A	A	B	C	D	E
A	0	0	1	1	0
B	0	0	1	0	1
C	1	1	0	0	1
D	1	0	0	0	1
E	0	1	1	1	0

M	A	B	C	D	E
A	0	0	½	½	0
B	0	0	½	0	½
C	.33	.33	0	0	.33
D	½	0	0	0	½
E	0	.33	.33	.33	0

$$\sum_j M_{ij} = 1$$

- Recall: the *ij*th entry of A^k corresponds to the number of paths from node *i* to node *j* of length *k*
- Similarly: the *ij*th entry of M^k corresponds to the probability of moving from node *i* to node *j* in *k* steps
  - The probability of moving from *i* to *j* in two steps is given by $(M^2)_{ij} = \sum_k M_{ik} M_{kj}$

Stochastic Matrix

- the *ij*th entry of M^k corresponds to the probability of moving from node *i* to node *j* in *k* steps
  - The probability of moving from *i* to *j* in two steps is given by $(M^2)_{ij} = \sum_k M_{ik} M_{kj}$
- If we keep multiplying the stochastic matrix, we compute probabilities of longer walks – we expect that transitions will occur more frequently within a natural cluster than between them
- To produce discrete clusters, Markov clustering also includes an inflation step to exaggerate these effects – make high probabilities higher and low probabilities lower
  - Inflation step: raise each element to the *k* power and renormalize

$$p_A = 0.9 \quad p_B = 0.1 \xrightarrow{k=2} \begin{matrix} p_A \to \frac{.81}{.81+.01} \\ p_B \to \frac{.01}{.81+.01} \end{matrix} \quad p_A = 0.99 \quad p_B = 0.01$$

Markov clustering algorithm

- **G** is a graph
- add self-transition loops to G to allow for no transition (e.g. staying in current node)
- Set inflation parameter (e.g. k=2)
- Initialize $\mathbf{M_1}$ to be the initial stochastic matrix of G
- Until convergence (e.g. while the stochastic matrix keeps changing):
  - $\mathbf{M_2 = M_1 * M_1}$
  - $\mathbf{M_1}$ **= inflate** $\mathbf{M_2}$
- Once you've converged, set the clustering to be the components of $\mathbf{M_1}$

Clustering graphs

- Markov clustering is a faster algorithm since it only requires matrix operations

- How do we decide which function to assign to members of a cluster?
  - Consensus based on nodes within each cluster that are already annotated
  - Determine significance by hypergeometric / use Gene Ontology

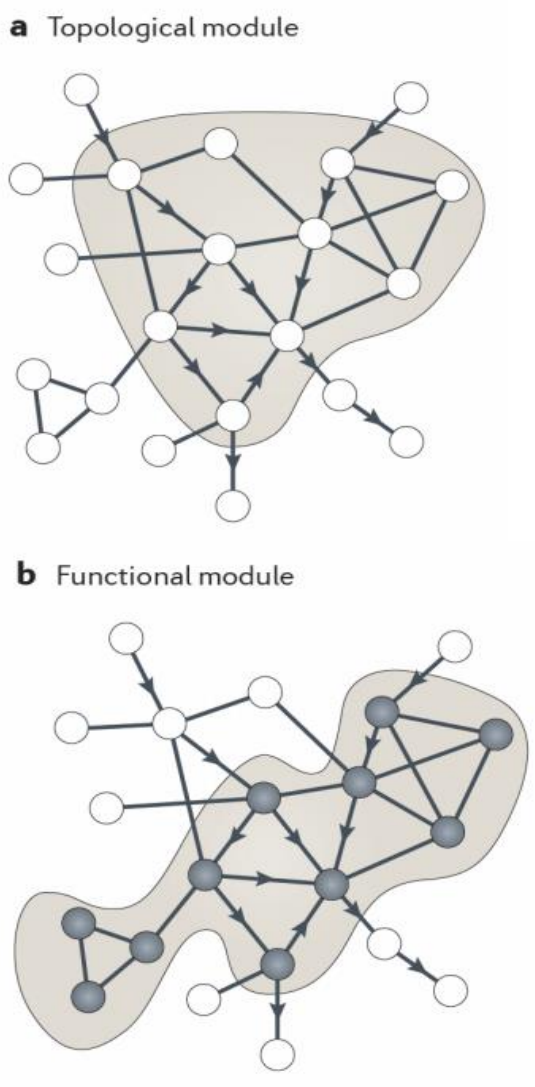
Annotating graphs

- Directly annotating unknown nodes
  - K-nearest neighbors
  - Local search
  - Simulated annealing

- Clustering graphs (then annotate each cluster)
  - Betweenness clustering
  - Markov clustering

Network Modules

- Networks have a modular structure
- Topological modules
  - subgraphs which are locally dense and have more connections among nodes within the module than with nodes outside the module
  - we found topological modules in our *clustering* algorithms
- Functional modules
  - subgraphs with a high density of functionally related nodes
  - we will look for functional modules in the ***active subnet*** algorithms (next)



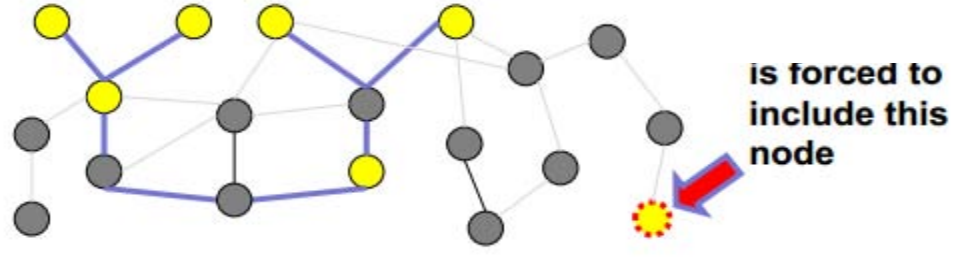
Active Subnet Problem

- So far, we have focused on labelling the nodes of a graph
- Now, assume we already have a labelled graph (e.g. We took a PPI network and labelled the edges with expression data – high/low)
- We would now like to find connected components of these graphs are enriched in a particular label – this is the **active subnet problem** (e.g. finding PPI subnetwork with high expression)
- This can reveal the hidden components of a biological response

Active Subnet Problem

- This problem has already been studied in theoretical Computer Science
- Steiner Tree Problem (NP-hard):
  - Given a graph **G = (V, E, w)** (*vertices, edges, and weights)* and a subset (*vertices of interest*)
  - **Find** the minimum weight tree (connected set of edges) that spans all the vertices in S
- (we didn't cover any algorithms in class – a couple optimization algorithms were sited)

We would like to find the minimum tree connecting all yellow nodes (given by blue lines).



However, we would also like to exclude the rightmost yellow node as its distance suggests it may be a false positive.

Active Subnet Problem

- To avoid false positives, we solve a problem variant – the ***prize-collecting*** Steiner tree problem
- Intuition:

$$\sum_{v \text{ not in } T} \beta\,\text{penalty}(v) + \sum_{e \text{ in } T} \text{cost}(e)$$

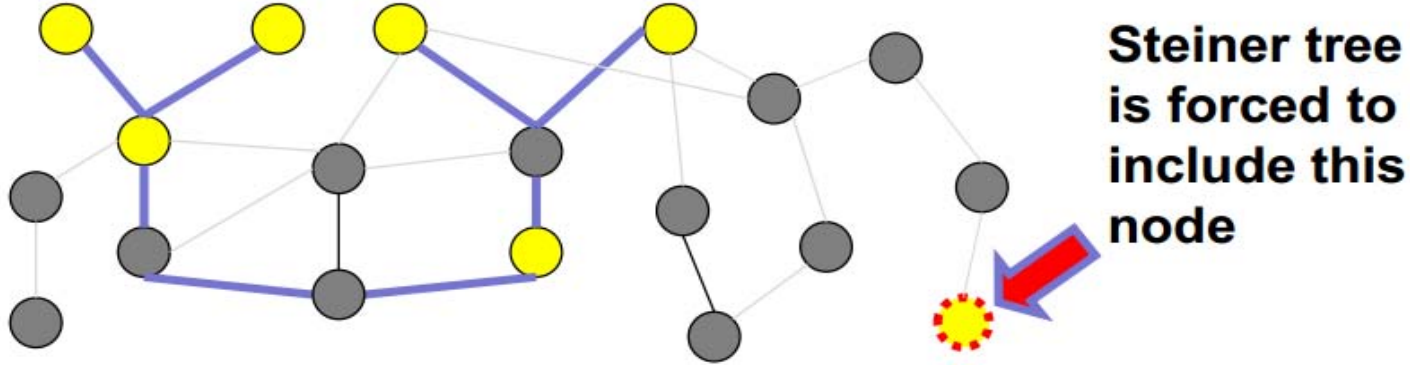
We pay a penalty for each vertex we do not include in the Steiner tree (we *collect a prize* for each vertex we do include)

We pay a cost for including edges in the Steiner tree

- The terms in the objective function balance each other
  - Adding too many edges to get to a node far away will induce a high edge penalty while collecting lower prize for the node

Active Subnet Problem

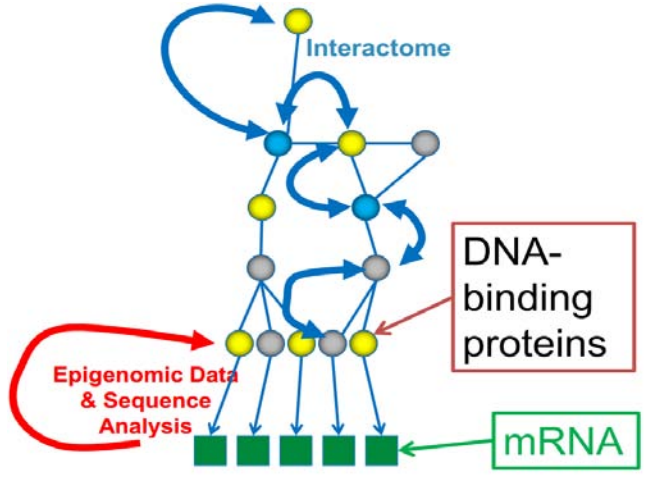
- While a traditional Steiner Tree would include the distant node, given some setting of the edge penalties and vertex prizes, the prize collecting Steiner Tree could exclude the node
- Eg. If this is a PPI network, the distantly labelled node could be highly expressed, but we would consider it unlikely to interact given its distance in the network



Data Integration

- When looked at individually, different data sources (mRNA levels vs gene knockout fitness) can give very different results
  - Genetic screens usually detect master regulators
  - mRNA measurements detect effectors (metabolic proteins)
- We can integrate these sources by piling the data into one network

Perturbation	Differentially expressed genes	Genetic hits	Number of overlapping genes
Growth arrest (Hydroxyurea)	59	86	0
DNA damage (MMS)	198	1448	43
Protein biosynthesis block (Cycloheximide)	20	164	0
ER stress (Tunicamycin)	200	127	5
ATP synthesis block (Arsenic)	828	50	9

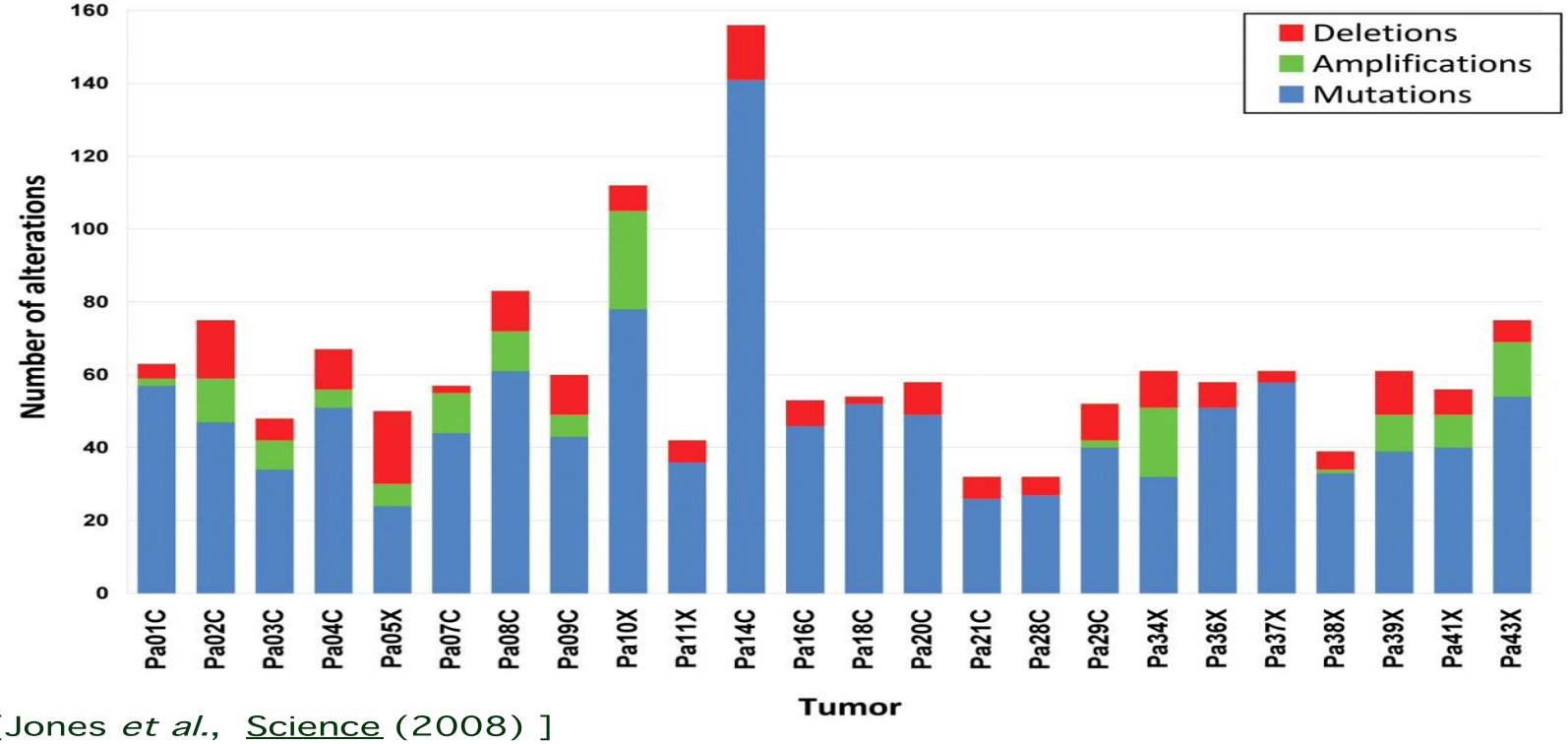


Logic modeling of cell signaling networks

- We can study gene signaling networks at many data levels
  - DNA sequence
  - mRNA expression
  - Protein levels
  - Dynamic protein operations (e.g. phosphorylation)
- Integrating these is difficult, but important to understand the full range of processes that might contribute to differences between normal and disease states

Difficulties in studying cancer

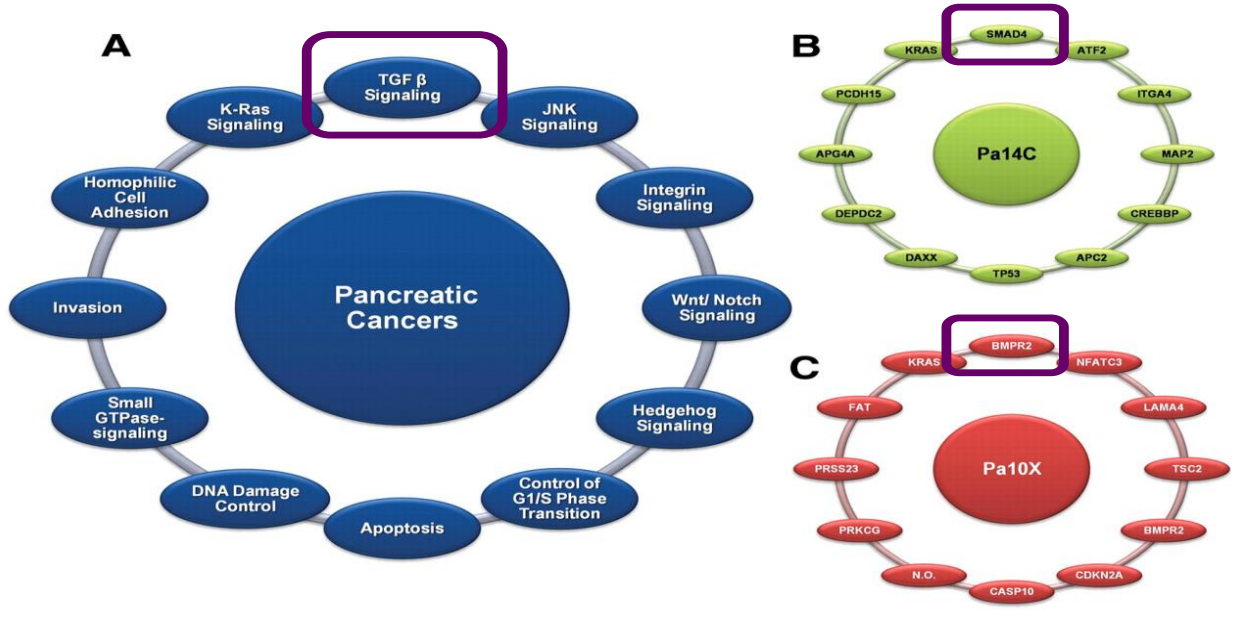
- There are many different genetic alterations that are observed within a tumor type



[Jones *et al.*, Science (2008)]

Difficulties in studying cancer

- There are many different genetic alterations that are observed within a tumor type
- But many of these mutations converge into a limited set of proteins and pathways



Source: Jones, Siân, Xiaosong Zhang, et al. "Core Signaling Pathways in Human Pancreatic Cancers Revealed by Global Genomic Analyses." *Science* 321, no. 5897 (2008): 1801-6.

Difficulties in studying cancer

- There are many different genetic alterations that are observed within a tumor type
- But many of these mutations converge into a limited set of proteins and pathways
  - We need to turn these pathways into computable models and circuitry that we can make predictions from and use as a basis for intervention strategies

Computational Modeling Approaches

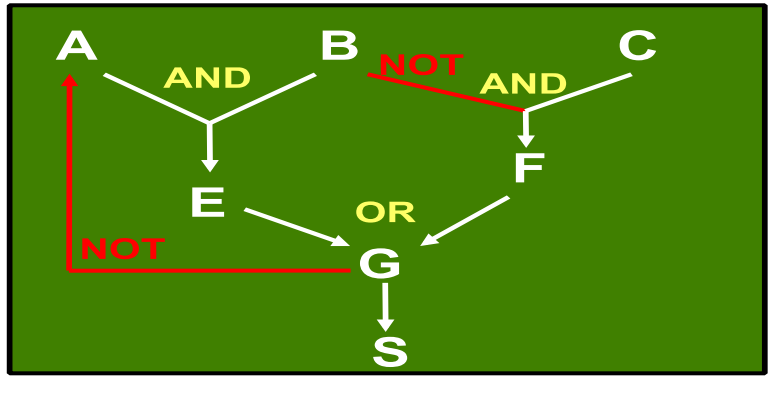
- Differential equations ("theory-driven")
  - Can write down differential equation models for up to dozens of variables to represent cellular mechanisms
  - However, we generally don't know enough about the rate constants to make differential equation models useful
- "Data-driven": regression, clustering of large data sets (e.g. RNA-seq, signaling pathway output measurements, mass-spec. proteomics)

Logic-based modeling

- We often take an intermediate approach that incorporates what is known (theory-driven) with data-driven experimental analyses
  - Incorporate existing pathway/interactome databases
    - Challenge: different databases often give disparate or even conflicting results!
    - Data is also very diverse: may come from different species, cell type, growth conditions, etc.
    - We can use these pathways and interactions as a starting point, but due to these limitations they can’t tell us everything
    - We must turn this starting network topology into an actionable, computable model by integrating relevant experimental data

Boolean Framework

- Often used on top of the network structure to explain experimental observations
- Variables (nodes – e.g. TFs or signaling molecules) are ON/OFF
- We connect nodes through AND & OR logic gates to represent cellular interactions & pathways

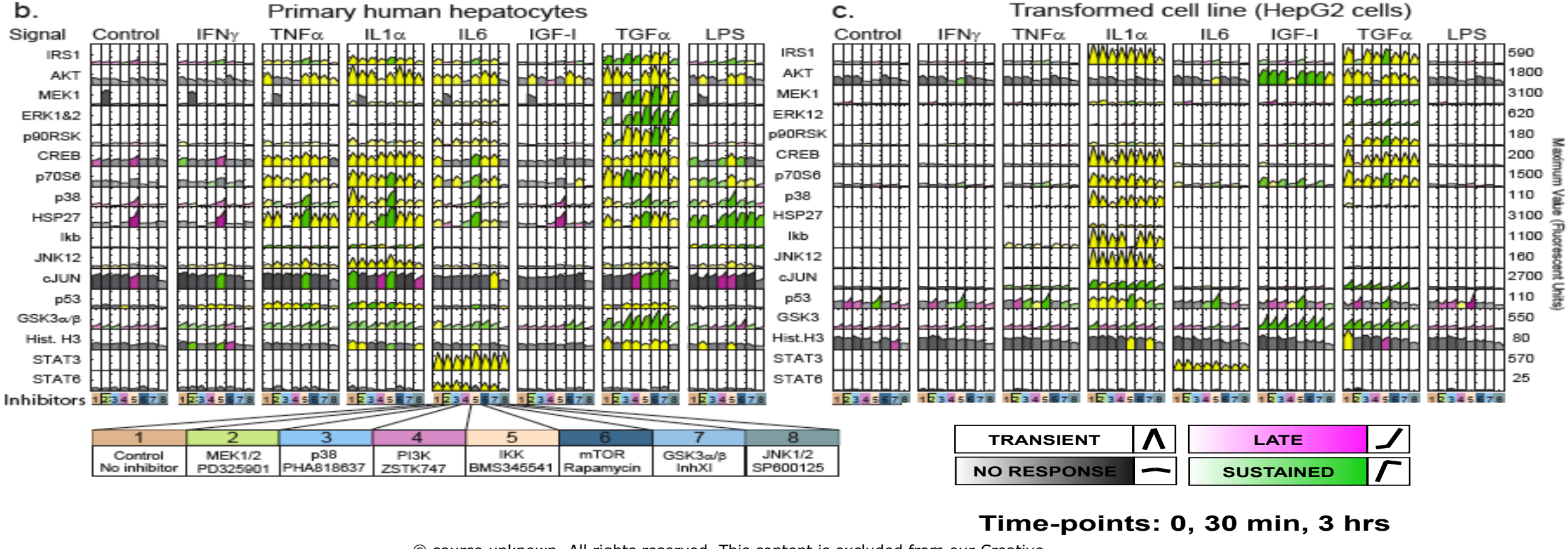


Boolean Framework

- Often used on top of the network structure to explain experimental observations
- Variables (nodes – e.g. TFs or signaling molecules) are ON/OFF
- We connect nodes through AND & OR logic gates to represent cellular interactions & pathways
- We can then perturb the variables *in silico* to predict outcomes if we introduce a therapeutic

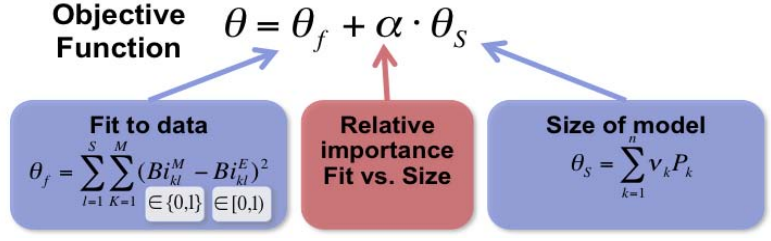
Sample Experimental Data

- Phoshoproteomic data (measuring the activity state of pathway molecules)
  - Multiple relevant signaling pathways
  - Introduce perturbations (e.g. inhibitor molecules) and measure outputs at various time points



Sample Experimental Data

- Phoshoproteomic data (measuring the activity state of pathway molecules)
  - Multiple relevant signaling pathways
  - Introduce perturbations (e.g. inhibitor molecules) and measure outputs at various time points
- Try to model the experimental data using the multiple potential networks (graph structures and associated quantitative relationships) from the databases
  - We must define some way to evaluate the model's performance to decide which is the most likely network



  - We want to consider not necessarily the single best model (because of experimental noise), but look at the full spectrum of top performing models to come up with a representative family of models

Improving upon models

- From the top performing model(s), make some perturbations of the model (via genetic algorithms) to improve the fit (similar to what we saw for Bayesian networks, e.g. Pebl)
- How well does this work in practice?
  - If we simply take the best result from the database and fit the experimental data to the model, we get ~45% error (because not all interactions are real and some are missing, might be the wrong cell type or environmental condition, etc.)
  - After model improvement (e.g. adding and removing edges), we may be able to reduce the error to below 10%
  - Often the model is not as complex as we might expect! Introducing too many variables may lead to simply fitting to noise and increase false positives

What are models used for?

- Can make predictions for new inhibitor combinations (therapeutic drug cocktails) that may be infeasible to exhaustively test experimentally
  - Comparing where our model predictions went wrong based on new experimental data informs us where our network model may be incorrect and how we might improve it
  - Edges that are included in the model and needed to explain the data can inform us about cellular processes (interactions that are missing in the databases or off-target effects of drugs)

Extending Boolean Logic for analog data

- Boolean logic deals with qualitative YES/NO (ON/OFF) relationships, but quantitative data is continuous
  - Instead of a step function from OFF to ON at a particular value, use continuous functions that have a graded slope from OFF to ON over a range of values
    - More realistic, but this ~doubles the number of free parameters in the model, so we need much more experimental data

Fin

7.91J / 20.490J / 20.390J / 7.36J / 6.802J / 6.874J / HST.506J Foundations of Computational and Systems Biology
Spring 2014