Net1: Last week's take home lessons

- **Macroscopic continuous concentration rates** (rbc)
  - **Cooperativity & Hill coefficients**
  - **Bistability** (oocyte cell division)
- **Mesoscopic discrete molecular numbers**
  - **Approximate & exact stochastic** (low variance feedback)
- **Chromosome Copy Number Control**
- **Flux balance optimization**
  - **Universal stoichiometric matrix**
  - **Genomic sequence comparisons** *(E.coli & H.pylori)*

Net2: Today's story & goals

- Biology to aid algorithms to aid biology
- Molecular & nano-computing
- Self-assembly
- Cellular network computing
- Genetic algorithms
- Neural nets

Algorithm Running Time

Given a size n problem, an algorithm runs $\boldsymbol{O(f(n))}$ time:

$\boldsymbol{O(f(n))}$: upper bound. (Ω :lower θ: equal)

	Time	$\boldsymbol{n} = 1$	$\boldsymbol{n} = 10$	$\boldsymbol{n} = 100$	$\boldsymbol{n} = 1000$
	$\boldsymbol{n}$	1	10	10^2	10^3
Polynomial	$\boldsymbol{n}^2$	1	10^2	10^4	10^6
	$\boldsymbol{n}^{10}$	1	10^{10}	10^{20}	10^{30}
Exponential	$2^{\boldsymbol{n}}$	2	$>10^3$	$>10^{30}$	$>10^{300}$
	$\boldsymbol{n}!$	1	$>10^6$	$>10^{150}$	$>10^{2500}$

Algorithm Complexity

- P = solutions in polynomial deterministic time.
  - e.g. dynamic programming
- NP = (non-deterministic polynomial time) solutions checkable in deterministic polynomial time.
  - e.g. RSA code breaking by factoring
- NP-complete = most complex subset of NP
  - e.g. traveling all vertices with mileage < x
- NP-hard = optimization versions of above
  - e.g. Minimum mileage for traveling all vertices
- Undecidable = no way even with unlimited time & space
  - e.g. program halting problem

NIST UCI

(http://hissa.ncsl.nist.gov/dads/HTML/npcomplete.html), **(http://www.ics.uci.edu/~eppstein/161/960312.html)**

How to deal with NP-complete and NP-hard Problems

- Redefine the problem into Class P:
  - – RNA structure Tertiary => Secondary
  - – Alignment with arbitrary function=>constant
- Worst-case exponential time:
  - – Devise exhaustive search algorithms.
  - – Exhaustive searching + Pruning.
- Polynomial-time close-to-optimal solution:
  - – Exhaustive searching + Heuristics (Chess)
  - – Polynomial time approximation algorithms

What can biology do for difficult computation problems

- DNA computing
  - A molecule is a small processor,
  - Parallel computing for exhaustive searching.
- Genetic algorithms
  - Heuristics for finding optimal solution, adaptation
- Neural networks
  - Heuristics for finding optimal solution, learning,...

Net2: Today's story & goals

- Biology to aid algorithms to aid biology
- Molecular nano-computing
- Self-assembly
- Cellular network computing
- Genetic algorithms
- Neural nets

Electronic, optical & molecular nano-computing

Steps: assembly > Input > memory > processor/math > output

Potential biological sources: harvest design evolve

A 30-fold improvement = 8 years of Moore’s law

Optical nano-computing & self-assembly

See Ebbesen et al., Extraordinary optical transmission through sub-wavelength hole arrays. *Nature* **391**, 667-669 (1998).

Vlasov et al. (2001) On-chip natural assembly of silicon photonic bandgap crystals.

Electronic-nanocomputing

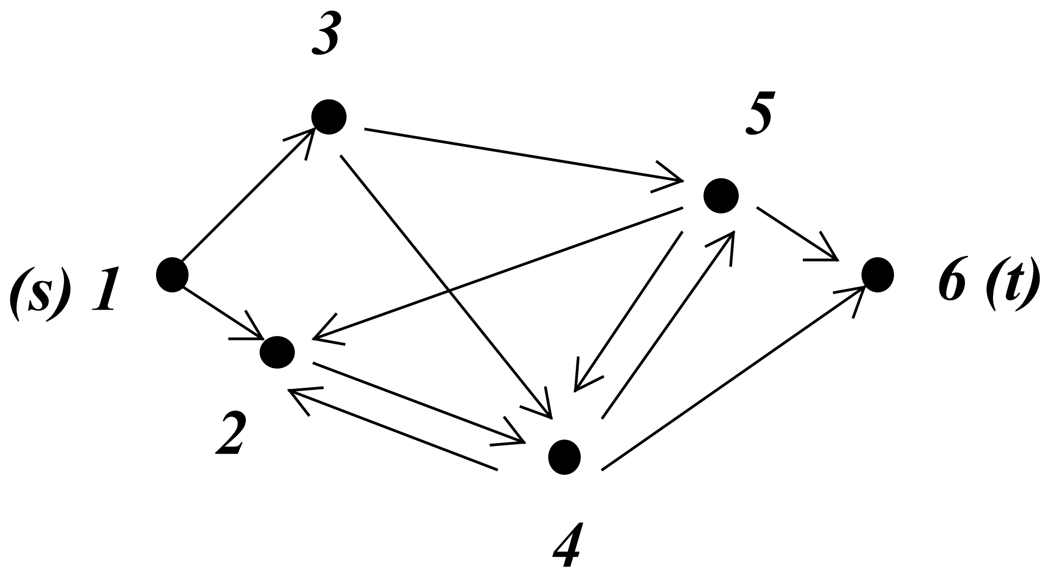
See Bachtold et al. & Huang et al. (2001) Science 294: 1317 , 1313.

(http://lib.harvard.edu:2058/cgi/content/full/294/5545/1317)

Molecular nano-computing

- R. P. Feynman (1959) American Physical Society, "There's Plenty of Room at the Bottom" (Pub)
  (http://www.zyvex.com/nanotech/feynman.html)
- K. E. Drexler (1992) Nanosystems: molecular machinery, manufacturing, and computation. (Pub)
  (http://www.zyvex.com/nanotech/nanosystems.html)
- L. M. Adleman, *Science* 266, 1021 (1994) Molecular computation of solutions to combinatorial problems.
- 727 references (Nov 2002)
  (http://www.wi.leidenuniv.nl/home/pier/webPagesDNA/index.html)

DNA computing: Is there a Hamiltonian path through all nodes?



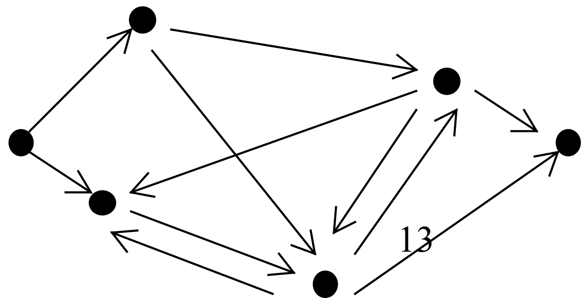
An ***st***-Hamiltonian path is (s,3,5,2,4,t).

L. M. Adleman, *Science* 266, 1021 (1994) Molecular computation of solutions to combinatorial problems.

DNA Computing for *st*-Hamiltonian Path

- Encode graph (nodes and edges) into ss-DNA sequences.
- Create all possible paths (overlapping sequences) using DNA hybridization.
- Determine whether the solution

  (or the sequence) exists.



Encode Graph into DNA Sequences

Nodes => Sequences:

…

```
3: 5′GTCACACTTCGGACTGACCT 3′
4: 5′TGTGCTATGGGAACTCAGCG 3′
5: 5′CACGTAAGACGGAGGAAAAA 3′
```

…

Edges => Sequences:

…

```
(3,4): 5′GGACTGACCTTGTGCTATGG 3′
(4,5): 5′GAACTCAGCGCACGTAAGAC 3′
```

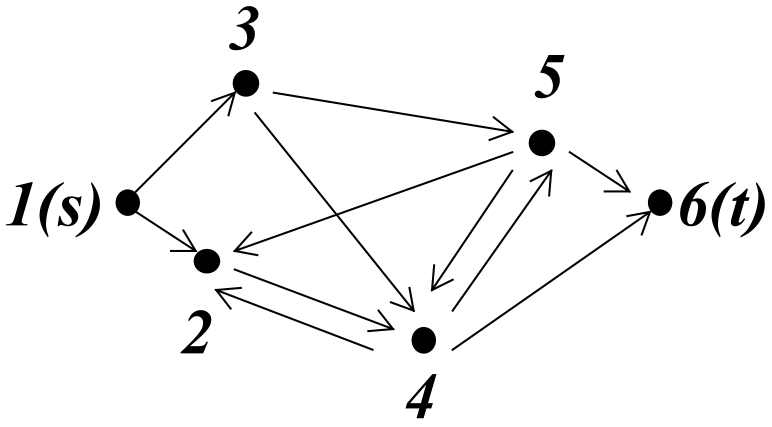
…

Reverse Sequences:

…

```
3: 5′AGGTCAGTCCGAAGTGTGAC 3′
4: 5′CGCTGAGTTCCCATAGCACA 3′
5: 5′TTTTTCCTCCGTCTTACGTG 3′
```

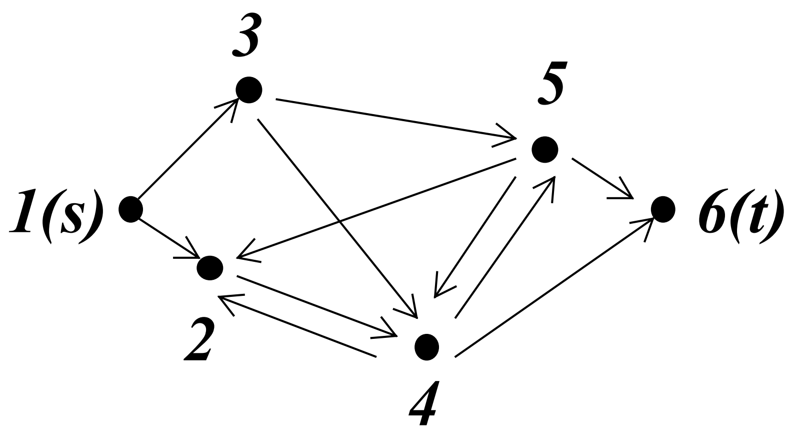
…



Edges + Nodes => Path (3,4,5):

```
                 Edge (3,4)                    Edge (4,5)
          GGACTGACCTTGTGCTATGGGAACTCAGCGCACGTAAGAC…
CAGTGTGAAGCCTGACTGGAACACGATACCCTTGAGTCGCGTGCATTCTG…
  Node 3 Reverse (3’ ← 5’)   Node 4 Reverse (3’ ← 5’)   Node 5 Reverse
```

Create All *st*-Paths



Start of a path:

(1,2): `5′ (Node1)+(PrefixOfNode2) 3′`
(1,3): `5′ (Node1)+(PrefixOfNode3) 3′`

End of a path:

(4,6): `5′ (SuffixOfNode4)+(Node6) 3′`
(5,6): `5′ (SuffixOfNode5)+(Node6) 3′`

All *st*-paths:

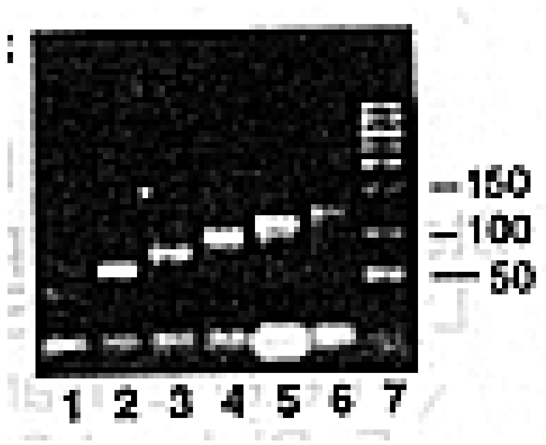
```
(1,2,4,6)
(1,3,5,6)
(1,3,5,2,4,6)
(1,3,4,5,4,6)
(1,2,4,5,2,4,5,2,4,5,6)
```
…

Path (1,2,4,6):

Edge (1,2): 5’ $\rightarrow$ 3’	**Edge (2,4): 5’ $\rightarrow$ 3’**	**Edge (4,6): 5’ $\rightarrow$ 3’**	
Node 1 Reverse (3’$\leftarrow$5’)	**Node 2 Reverse (3’$\leftarrow$5’)**	**Node 4 Reverse (3’$\leftarrow$5’)**	**Node 6 Reverse (3’$\leftarrow$5’)**

DNA Computing Process

•Encode graph into DNA sequences.	•Oligonucleotide synthesis
•Create all paths from ***s*** to ***t***.	•PCR
•Extract paths that visit every node.	•Serial hybridization
•Extract all paths of ***n*** nodes.	•Electrophoretic size
•Report Yes if any path remains	•Graduated PCR electrophoretic fluorescence

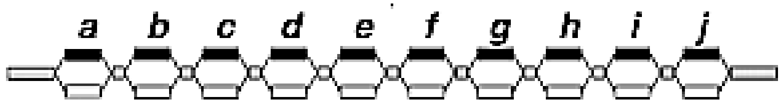


Molecular computation: RNA solutions to chess problems.

See Faulhammer, et al. 2000 PNAS 97, 1385-1389. (Pub)
(**http://www.pnas.org/cgi/content/full/97/4/1385)**

split & pool oligonuc. synthesis

split & pool RNase H elimination



$$((\neg h \wedge \neg f) \vee \neg a) \wedge ((\neg g \wedge \neg i) \vee \neg b) \wedge ((\neg d \wedge \neg h) \vee \neg c) \wedge ((\neg c \wedge \neg i) \vee \neg d) \wedge ((\neg a \wedge \neg g) \vee \neg f).$$

Problems of DNA Computing

- Polynomial time but exponential volumes
- A 100 node graph needs $>10^{30}$ molecules.
- Far slower than a PC.
- Experimental errors:
  - mismatch hybridization
  - incomplete cleavage
- Non-reusable.

Promises of DNA Computing

- High parallelism
- Operation costs near thermodynamic limit
  - 2 vs 34x10^{19} ops/J (${10}^9$ for conventional computers)
- Solving one NP-complete problem implies solving many.
- Possible improvement
  - Faster readout techniques (eg. DNA chips).
  - Natural selection.

A sticker-based model for DNA computation.

Roweis et al. J Comput Biol 1998; 5:615-29 (Pub, JCB)
(http://www.cs.sandia.gov/jcb/v5/n4/v5n4art1.html)
Unlike previous models, the stickers model has a random access memory that requires no strand extension and uses no enzymes.

In theory, ...reusable. [We] propose a specific machine architecture for implementing the stickers model as a microprocessor-controlled parallel robotic workstation…

Concerns about molecular computation (Smith, 1996; Hartmanis, 1995; Linial et al., 1995) are addressed:

1) General-purpose algorithms can be implemented by DNA-based computers
2) Only modest volumes of DNA suffice.
3) [Altering] covalent bonds is not intrinsic to DNA-based computation.
4) Means to reduce errors in the separation operation are addressed in
   Karp et al., 1995; Roweis and Winfree, 1999).

3SAT

Given n boolean (0/1) variables $x = (x_1, x_2, \dots, x_n)$,

and m 3-variable clauses $c = (c_1, c_2, \dots, c_m)$,

is $c_1 \wedge c_2 \wedge \dots \wedge c_m$ satisfiable for some x?

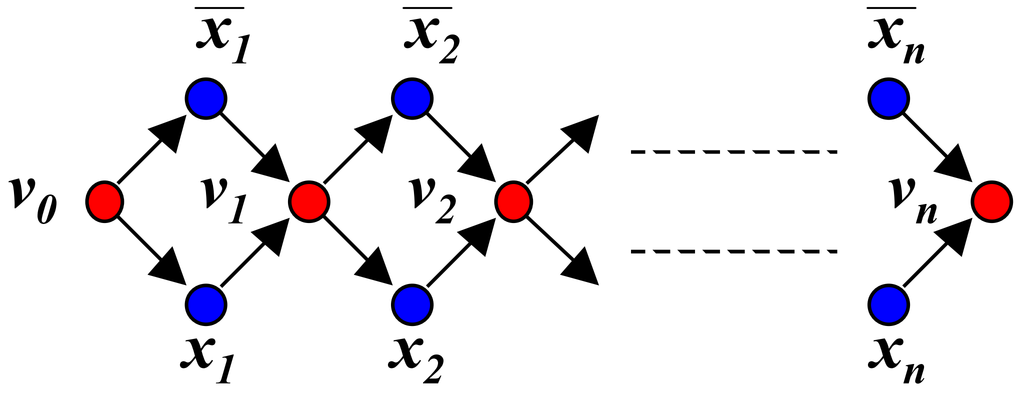
$$c_1 = x_1 \vee \bar{x}_3 \vee \bar{x}_7$$

$$c_2 = \bar{x}_1 \vee x_2 \vee x_4$$

…

$$c_m = x_1 \vee x_{m-1} \vee \bar{x}_m$$

DNA Computing for 3SAT



ALGORITHMS:

1. Encode Graph ***G*** into DNA sequences.
2. Create all paths from v_0 to v_n.
3. For every clause
4. Select sequences that satisfy this clause.
5. Report Yes or No.

DNA computing on surfaces

Liu Q, et al. Nature 2000;403:175-9 A set of DNA molecules encoding all candidate solutions to the computational problem of interest is synthesized on a surface. Cycles of hybridization operations and exonuclease digestion identify & eliminate non-solutions.

The solution is identified by PCR and hybridization to an addressed array. The advantages are scalability and potential to be automated (solid-phase formats simplify repetitive chemical processes, as in DNA & protein synthesis). Here we solve a NP-complete problem (SAT) (Pub)

(http://www.nature.com/cgitaf/DynaPage.taf?file=/nature/journal/v403/n6766/full/403175a0_fs.html&filetype=&content_filetype=&_User Reference=D82349ED46B4ACCCE594B859D7113A214DE4)

Braich RS, Chelyapov N, Johnson C, Rothemund PW, Adleman L. Solution of a 20-variable 3-SAT problem on a DNA computer. Science. 2002 Apr 19;296(5567):499-502.

(http://80-www.ncbi.nlm.nih.gov.ezp1.harvard.edu/entrez/query.fcgi?cmd=Retrieve&db=PubMed&list_uids=11896237&dopt=Abstract)

Net2: Today's story & goals

- Biology to aid algorithms to aid biology
- Molecular nano-computing
- Self-assembly
- Cellular network computing
- Genetic algorithms
- Neural nets

Logical computation using algorithmic self-assembly of DNA triple-crossover molecules.

Aperiodic mosaics form by the self-assembly of 'Wang' tiles, emulating the operation of a Turing machine … a logical equivalence between DNA sticky ends and Wang tile edges. Algorithmic aperiodic self-assembly requires greater fidelity than periodic, because correct tiles must compete with partially correct tiles. Triple-crossover molecules that can be used to execute four steps of a logical (cumulative XOR) operation on a string of binary bits. (a XOR b is TRUE only if a and b have different values)

Mao et al. Nature 2000 Sep 28;407(6803):493-6(Pub)

(http://www.nature.com/cgitaf/DynaPage.taf?file=/nature/journal/v407/n6803/full/407493a0_fs.html&_UserReference=D82349ED46B4F23D3460377A1B753A238D2E)

Nanoarray microscopy readout
(vs gel assays)

See Winfree et al, 1998; Nature 394, 539 - 544 (Pub)
(http://seemanlab4.chem.nyu.edu/two.d.html)

Micro-ElectroMechanical Systems (MEMS)

"Ford Taurus models feature Analog Devices' advanced airbag sensors"

"A unit gravity signal will move the beam 1% of the beam gap and result in a 100fF change in capacitance. Minimal detectable deflections are 0.2 Angstroms; less than an atomic diameter. "
(tech specs)
(http://www.analog.com/publications/whitepapers/products/Sensordetroit/Sensordetroit.html)

Nano-ElectroMechanical Systems (NEMS)

See Soong et al. Science 2000; 290: 1555-1558.Powering an Inorganic Nanodevice with a Biomolecular Motor. (Pub)
(http://www.sciencemag.org/cgi/content/full/290/5496/1555)

Nanosensors

See Meller, et al. (2000) "Rapid nanopore discrimination between single polynucleotide molecules." PNAS 1079-84. Akeson et al. Microsecond time-scale discrimination among polyC, polyA, and polyU as homopolymers or as segments within single RNA molecules. Biophys J 1999;77:3227-33

(http://www.pnas.org/cgi/content/full/97/3/1079)
(http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Retrieve&db=PubMed&list_uids=10585944&dopt=Abstract)

poly(dA)$_{100}$ & poly(dC)$_{100}$ at 15°C

See Vercoutere M., et al, Rapid discrimination among individual DNA hairpin molecules at single-nucleotide resolution using an ion channel. Nat Biotechnol. 2001 Mar;19(3):248-52.

Net2: Today's story & goals

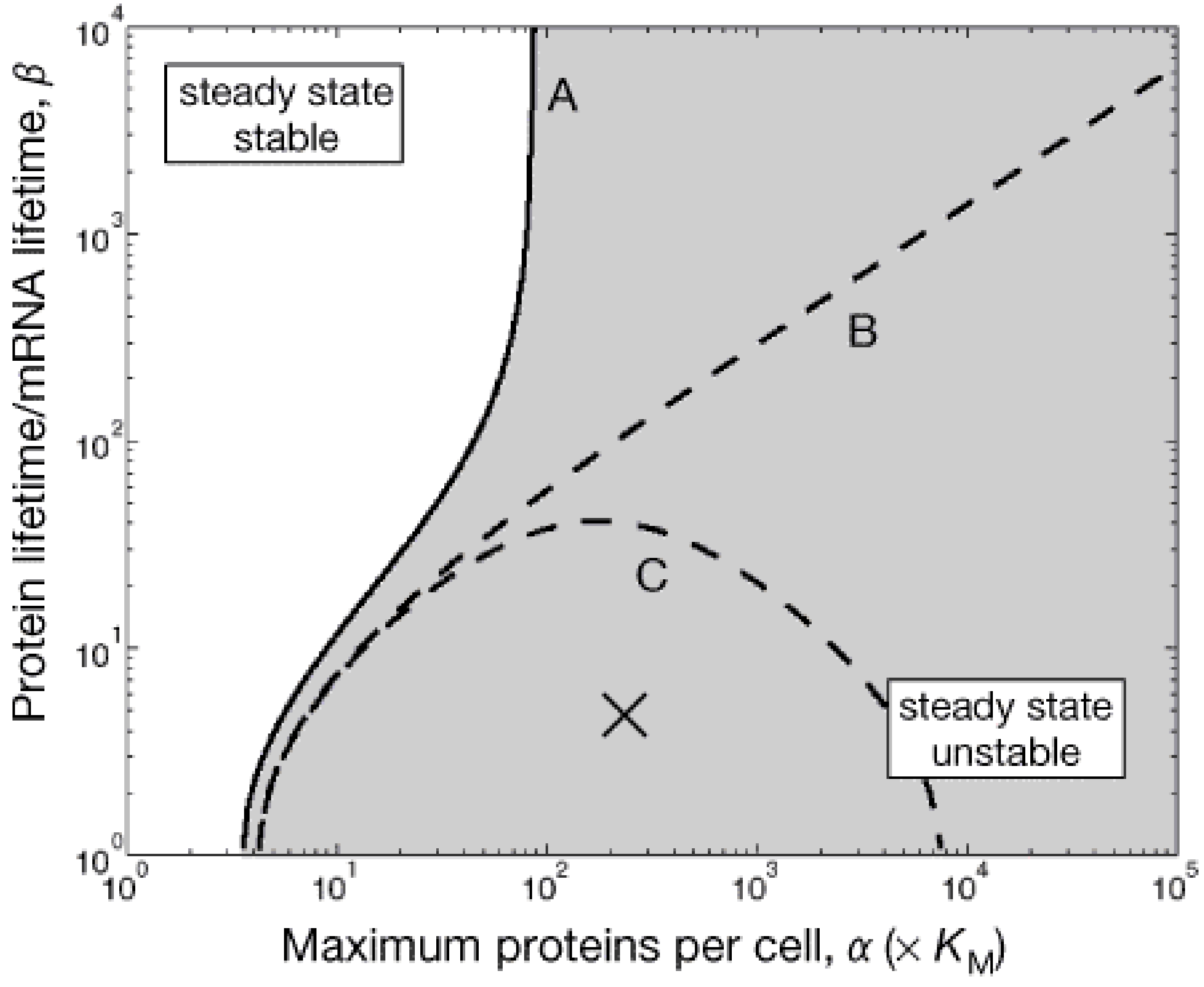
- Biology to aid algorithms to aid biology
- Molecular nano-computing
- Self-assembly
- Cellular network computing
- Genetic algorithms
- Neural nets

A synthetic oscillatory network of transcriptional regulators

See Elowitz &Leibler, (Pub), Nature 2000;403:335-8

(http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?CMD=Text&DB=PubMed)
(http://www.nature.com/cgitaf/DynaPage.taf?file=/nature/journal/v403/n6767/full/403335a0_fs.html&_UserReference=D82349EC46B4ABC190D3999B98E33A23D0CE)

Synthetic oscillator network

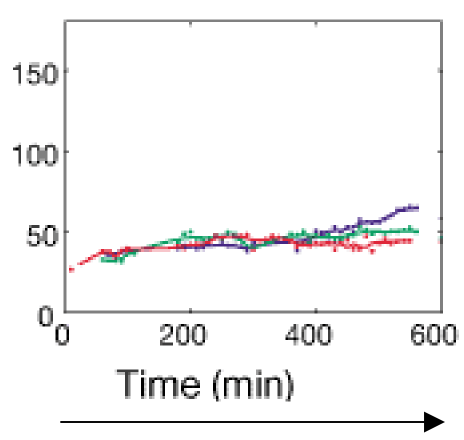


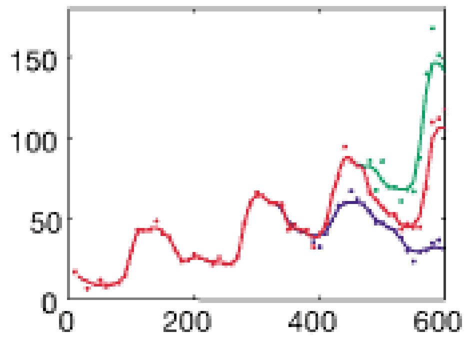
Synthetic oscillator network

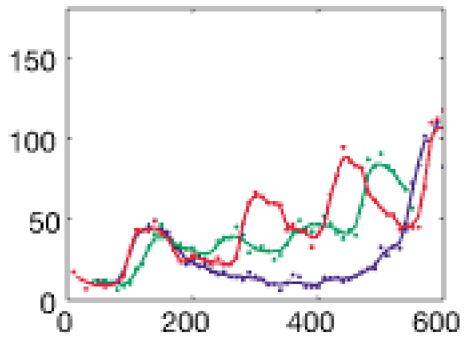
Controls with IPTG

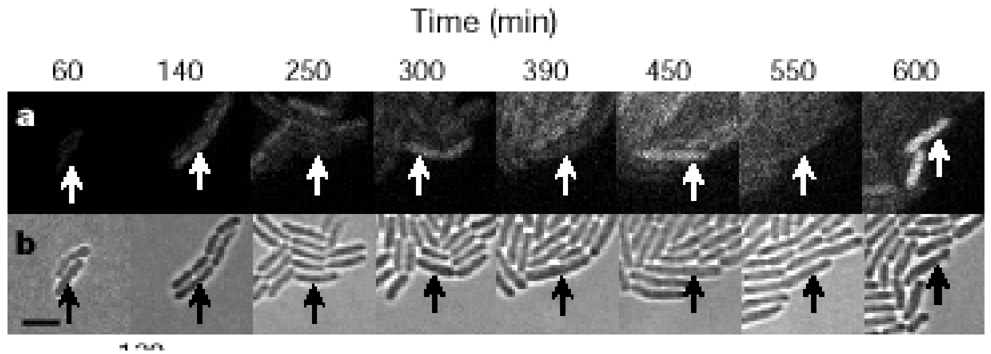
Variable <u>amplitude</u> & <u> period</u> in sib cells

Single cell GFP levels

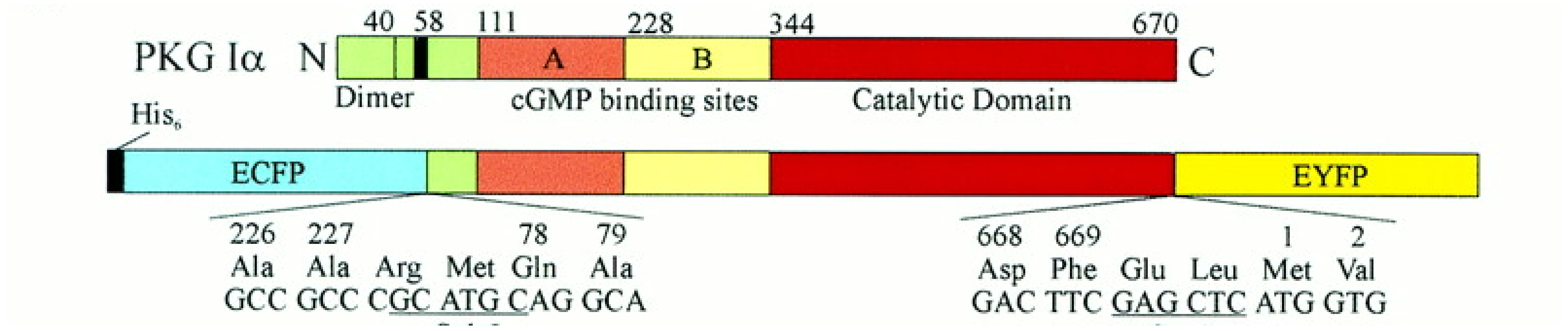








Internal state sensors



See Honda et al (2001) PNAS 98:2437-42 Spatiotemporal dynamics of **cGMP** revealed by a genetically encoded, fluorescent indicator.
(http://www.ncbi.nlm.nih.gov/entrez/utils/fref.fcgi?http://www.pnas.org/cgi/pmidlookup?view=full&pmid=11226257)

and

Ting et al. protein kinase/phosphatase activities
(http://www.tsienlab.ucsd.edu/HTML/People/Alice/Alice Ting.htm)

Net2: Today's story & goals

- Biology to aid algorithms to aid biology
- Molecular nano-computing
- Self-assembly
- Cellular network computing
- Genetic algorithms
- Neural nets

Genetic Algorithms (GA)

1. Initialize a random population of individuals (strings)
2. Select a sub-population for offspring production
3, Generate new individuals through genetic operations
    (mutation, variation, and crossover)
4. Evaluate individuals with a fitness function.
5. If solutions are not found, Go to step 2
6. Report solution.

Genetic Operations

Mutation

...ACCGGTTACGTTGGA...

$\downarrow$

...ACCGGTTGCGTTGGA...

Crossover

...ACCGGTTT|TCGTTGGA...

...CGTACGCC|GTTTACCC...

$\downarrow$

...ACCGGTTTGTTTACCC...

...CGTACGCCTCGTTGGA...

SAGA: Sequence Alignment by Genetic Algorithm

[DP: $O(2^N L^N)$ N sequences length L]

Improve fitness of a population of alignments by an objective function which measures multiple alignment quality, [using] automatic scheduling to control 22 different operators for combining alignments or mutating them between generations.

See C. Notredame & D. G. Higgins, 1996 (Pub)

(http://igs-server.cnrs-mrs.fr/~cnotred/Publications/Html/Saga_paper_html/saga_paper.html)

SAGA continues

The 16 block shuffling operators, the two types of crossover, the block searching, the gap insertion and the local rearrangement operator, make a total of 22. Each operator has a probability of
being used that is a function of the efficiency it has recently (e.g. 10 last generations) displayed at improving alignments.

Comparison of ClustalW & SAGA

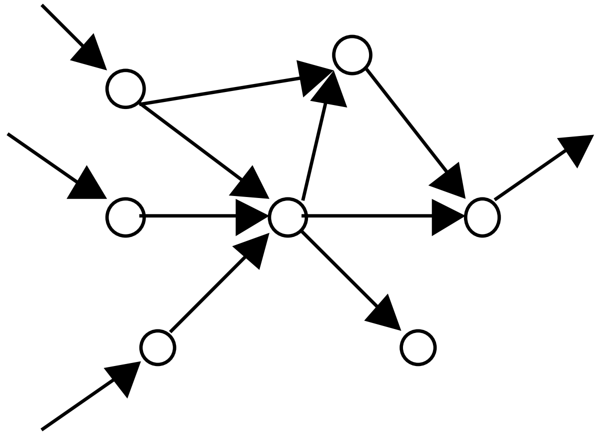
Test case	Nseq	CLUSTAL W versus structure (%)	CPU-time	SAGA versus structure (%)	CPU-time
Igb	32	55.86	60	55.97	41 135
Ac Protease2	10	41.02	16	43.50	12 236
S Protease2	12	64.37	21	66.18	20 537
Globin2	12	94.90	18	94.01	2538

Net2: Today's story & goals

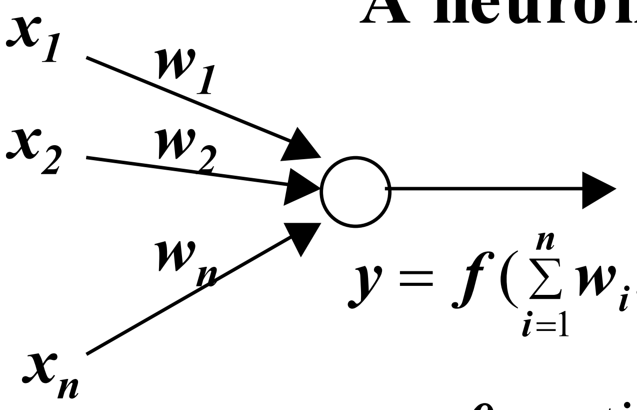
- Biology to aid algorithms to aid biology
- Molecular nano-computing
- Self-assembly
- Cellular network computing
- Genetic algorithms
- Neural nets

Artificial Neural Networks

A neural network:



A neuron



$$y = f\left(\sum_{i=1}^{n} w_i x_i\right)$$

$y>=0$: active

$y<0$: inactive

Neural Networks

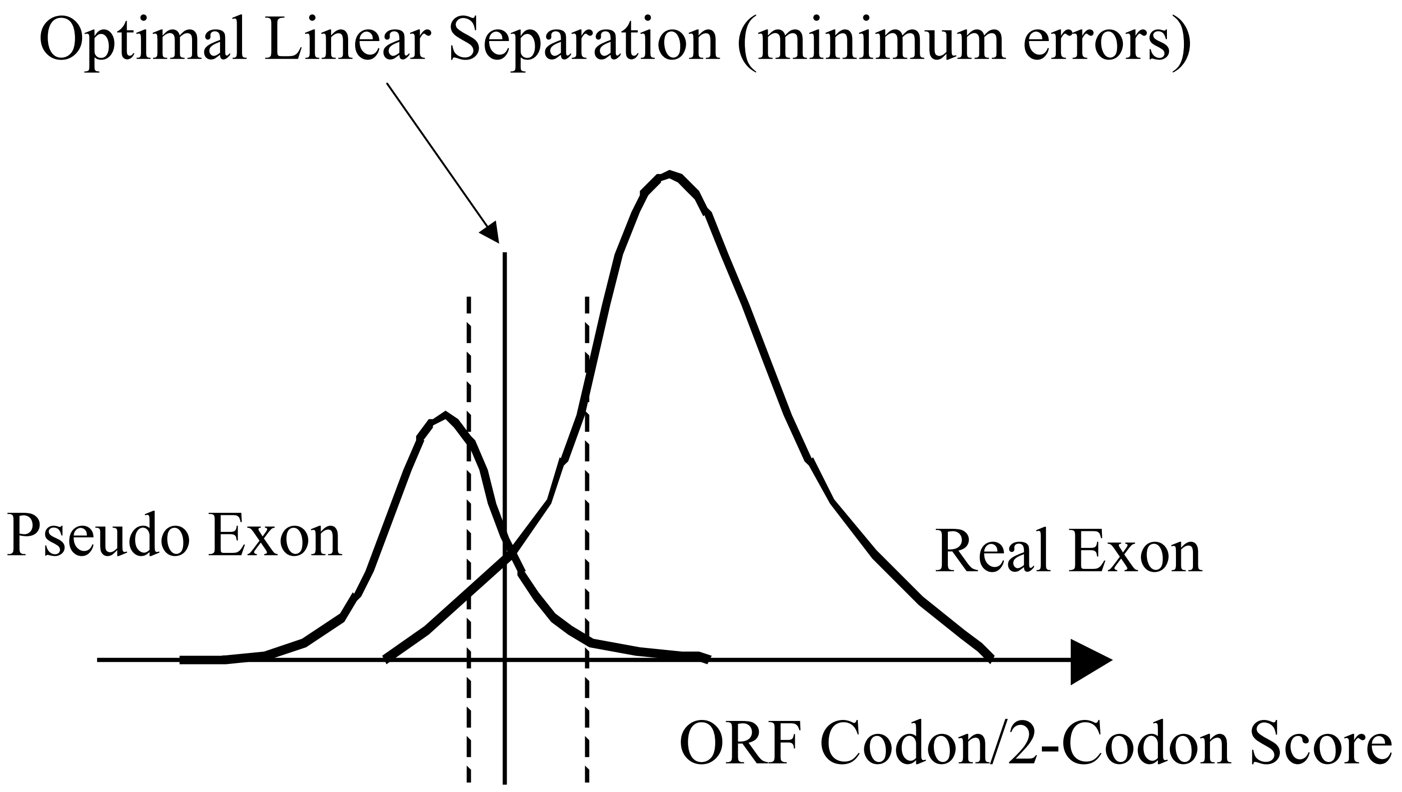
McCulloch and Pitts (1943) Neurology inspired "& /OR"operations

Werbos 1974 back-propagation learning method

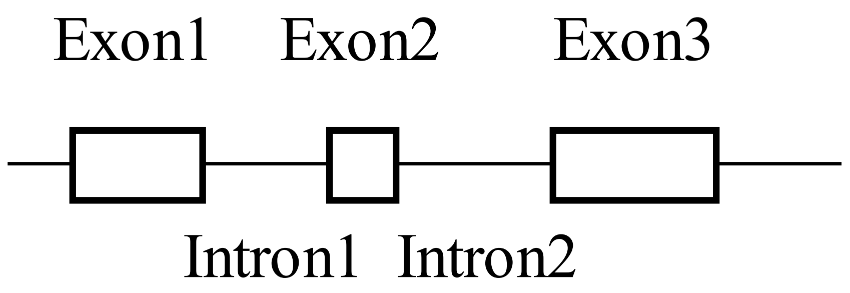
Hopfield 1984, PNAS 81:3088-92 Neurons with graded response have collective computational properties like those of two-state neurons. (Pub)
(http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Retrieve&db=PubMed&list_uids=6587342&dopt=Abstract)

(ANN)
(http://www-dse.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html)

An ORF Classification Example



Measuring Exons



Exon Features {

Donor Site Score,
Acceptor Site Score,
In-frame 2-Codon Score,
Exon Length (log),
Intron Scores,
…… }

Linear Discriminate Function and Single Layer Neural Network

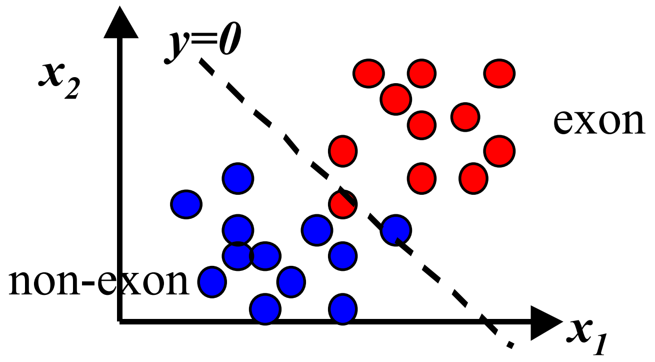
Exon: e=($x_1 x_2 \ldots x_d$)

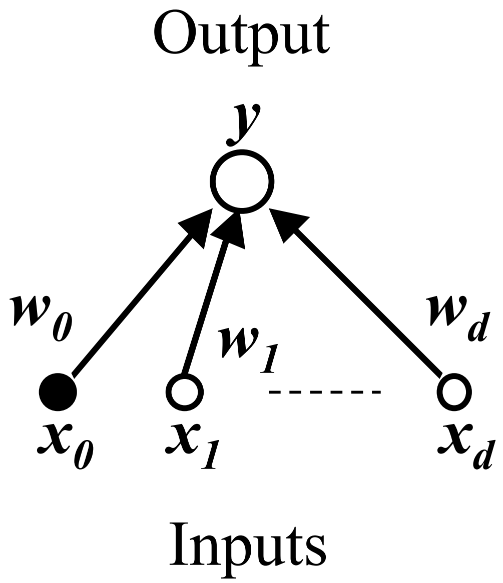
A linear separator :

$$y = \sum_{i=1}^{d}(w_i x_i) + w_0$$

$$y > 0 : \textbf{Exon} \quad y < 0 : \textbf{Non-Exon}$$

A 2-feature linear separation





An activation function :

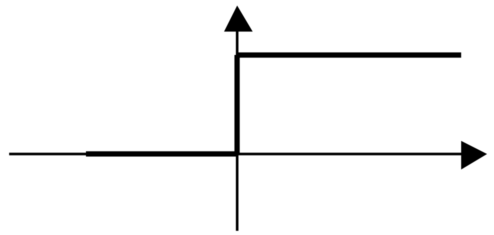
$$y = f\left(\sum_{i=0}^{d} w_i x_i\right)$$

Activation Function

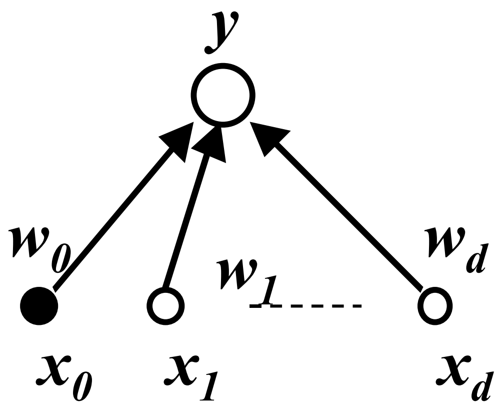
$$\begin{cases} f(a) = 0 & a < 0 \\ f(a) = 1 & a \geq 0 \end{cases}$$

$$f(a) = a$$

Step Function



Output

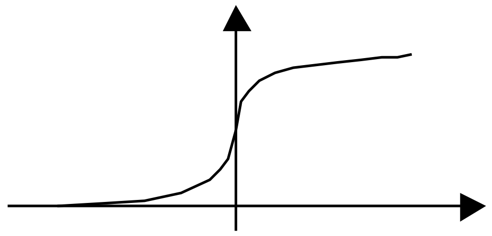


Inputs

$$y = f\left(\sum_{i=0}^{d} w_i x_i\right)$$

$$f(a) = \frac{1}{1 + e^{a}}$$

Sigmoid Function



Determining Edge Weights from Training Sets

Given a set of n known exons/nonexons :

$$(\overline{e}_1, t_1), (\overline{e}_2, t_2), \ldots, (\overline{e}_n, t_n)$$

Step1 **Initialize w**

Step2 **Sum of squares error function :**

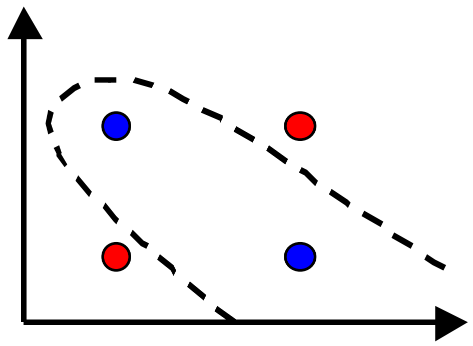
$$E(\overline{w}) = \tfrac{1}{2} \sum_{k=1}^{n} \{f(\overline{e}_k, \overline{w}) - t_k\}^2$$

Step3 **Updating w_j**

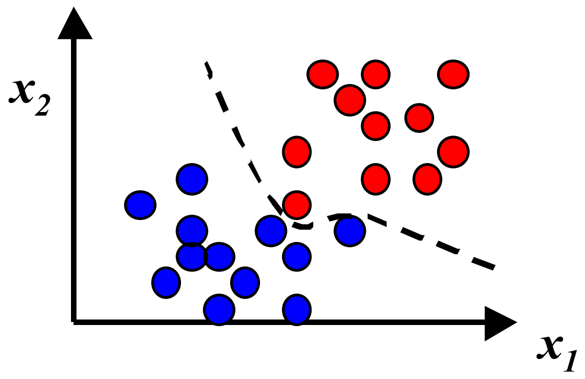
$$w_j^{\tau+1} = w_j^{\tau} - \lambda \frac{\partial E(w)}{\partial w_j} \Big|_{\overline{w}^{\tau}}$$

Non-linear Discrimination

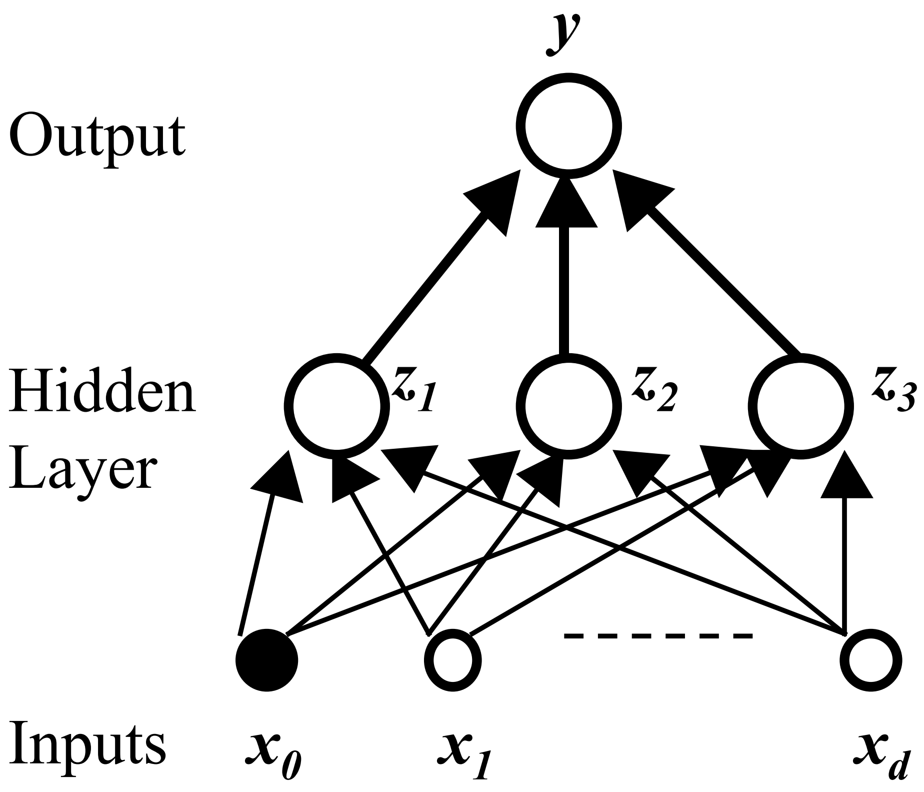
Exclusive-OR Problem



A 2-feature non-linear separation



The Multi-Layer Perceptron



$$y = g\left(\sum_{i=1}^{3} w_i^{(2)} z_i\right)$$

$$z_j = f\left(\sum_{i=0}^{d} w_{ji}^{(1)} x_i\right)$$

Training: Error Back Propagation.

GRAIL

Located 93% of all exons regardless of size with a false positive rate of 12%. Among true positives, 62% match actual exons exactly (to the base), 93%
match at least one edge exactly.

See Xu et al, Genet Eng 1994;16:241-53
Recognizing exons in genomic sequence using GRAIL II.
(Pub)
(http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Retrieve&db=PubMed&list_uids=7765200&dopt=Abstract)

Net2: Today's story & goals

- Biology to aid algorithms to aid biology
- Molecular nano-computing
- Self-assembly
- Cellular network computing
- Genetic algorithms
- Neural nets