Discovery of Regulatory Elements by a Phylogenetic Footprinting Algorithm[*]

Martin Tompa & Mathieu Blanchette
Computer Science & Engineering
University of Washington

March 8, 2006

1 Outline of the talk

- How are genes regulated?
- What is phylogenetic footprinting?
- A first solution
- Improvements and extentsions
- Application to regulation of several important genes

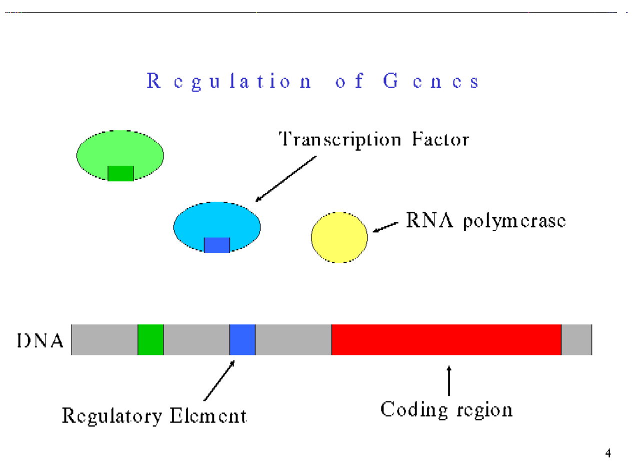
2 Regulation of Genes

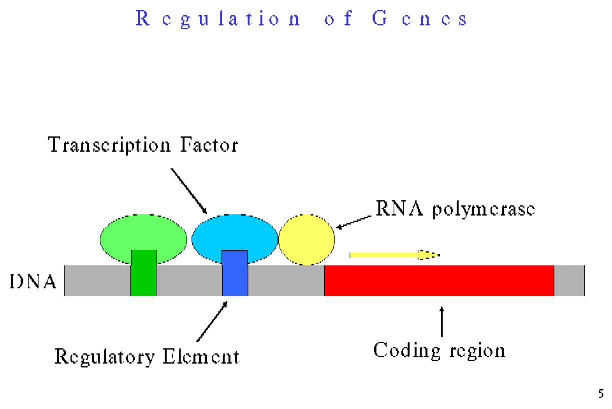
What turns genes on and off? This is a big questions in biology - nobody knows the complete answer. We want to explore mechanisms by which cells turn on and off a gene to control its gene products.

When is a gene turned on or off? If you pick an arbitrary gene and ask what's controlling it, in general we don't know. We're looking at promotors and transcription factors (TFs). We want to know which TFs regulate which genes, and under what external conditions: heat shock, nutrients, etc.

We're going to look at a tiny little piece of the puzzle - the mechanism of turning a gene on at the very beginning of the process.

[*]Talk by Martin Tompa at the University of Toronto, CSC 2427 class, March 8, 2006. Notes by Dustin Lang. Figures indiscriminately stolen from Tompa's slides.





In the promoter region in front of the (coding region of the) gene, there are regulatory elements which bind transcription factors. The TFs bind to their targets, together they recruit the RNA polymerase, and the RNA polymerase begins transcribing.

The question for today: how can you predict the regulatory elements?

3 The Goal

Identify regulatory elements in DNA sequences. These are:

- the binding sites for (transcription factor) proteins
- 5-25 base pairs long
- up to 1000 base pairs upstream from the gene (they can be farther away, especially in higher eucaryotes, but we'll focus on the 1000-bp window)
- inexactly repeating patterns (motifs)

TFs are very tolerant of slight changes in the pattern. The binding sites across different genes and different organisms look slightly different.

In yeast and bacteria, many regulatory elements are closer than 1000 base pairs to the coding region. In mammals, most are that close, though some can be (much) further.

4 Phylogenetic footprinting

An idea introduced by Tagle *et al* 1988, for finding functional sequences in general. Simply put, functional sequences evolve more slowly than nonfunctional sequences. This is due to selective pressure: if functional regions mutate too fast, they risk becoming ineffective.

We want to find unusually well-conserved regions; these are good candidates for being functional sequences. We will look at orthologous regions of genomes.

Think of "orthologous" as meaning "corresponding": the insulin gene in human and the insulin gene in mouse are orthologous. Evolutionarily, they evolved from the same ancestral sequence.

5 Substring Parsimony Problem

We'll phrase the problem of finding unusually well-conserved regions as a substring parsimory problem.

We're given:

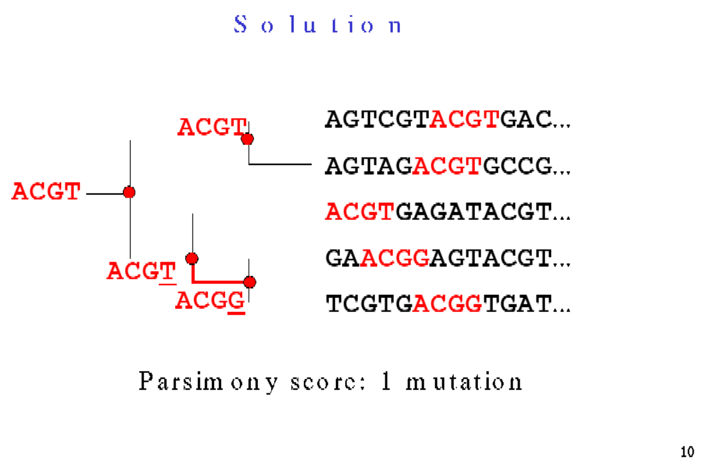
- a phylogenetic tree T
- a set of orthologous sequences at the leaves of T
- the length k of the motif we're looking for
- a parsimony threshold d

For example, we could be given a tree of mammals, their leaves being human, rat, mouse, dog, and cow. We would also be given the 1000 base pairs upstream of the insulin gene in each of these genomes; these are our orthologous sequences.

The problem: find each set S of k-mers, one k-mer from each leaf, such that the parsimony score of S in T is at most d.

A typical value of k is 10 base pairs.

The parsimony score along one edge on the tree is the minimum number of mutations to get from one sequence to the other. We have to label each interior node of T with a sequence, and the parsimony score of S in T is defined as the sum of parsimonies along each edge in the tree. Essentially, we want to find the minimum number of mutations in the phylogenetic tree required to explain the leaves we observe.



For today's talk, we'll assume a mutation is a single base-pair substitution. Insertion and deletion can be done, but it makes things more complicated.

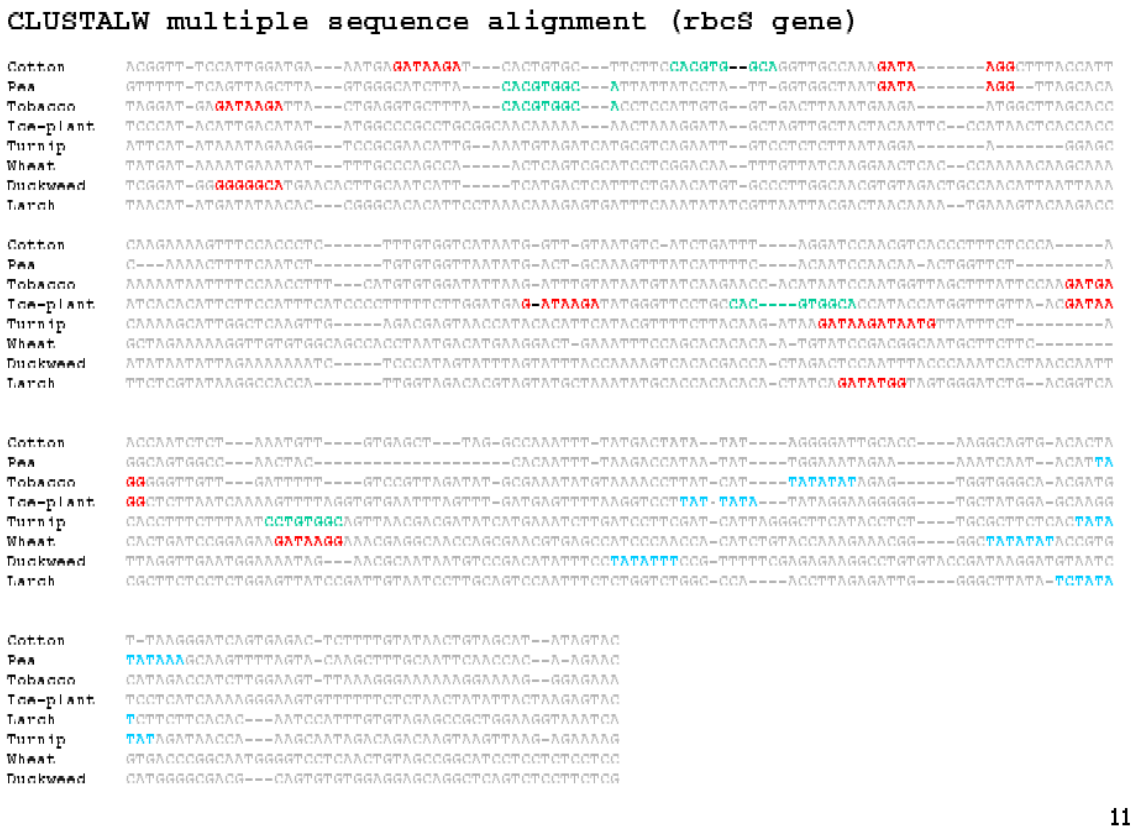
Unfortunately, this problem is NP-hard. In fact, the question, "is there such a set S?" is NP-complete.

We can either approximate it heuristically, or we have to use an exponential time algorithm. We'll do the latter, and as it turns out it's "Nicely Exponential."

6 Approach

This problem is usually approached via multiple sequence alignment. Take all the sequences, run multiple sequence alignment, and look for 10 columns that are unusually well preserved.

The problem with global multiple sequence alignment is that we're looking for very short signals in lots of noise. Global multiple alignment tries to align all that noise, and in doing that it may lose the signal. Below is a multiple alignment produced by the program `CLUSTALW`. Regions in colour are known binding sites; they're not very well aligned.



CLUSTALW multiple sequence alignment (rbcS gene)

```
Cotton      ACGGTT-TCCATTGGATGA---AATGAGATAAGAT---CACTGTGC---TTCTTCCACGTG--GCAGGTTGCCAAAGATA-------AGGCTTTACCATT
Pea         GTTTTT-TCAGTTAGCTTA---GTGGGCATCTTA----CACGTGGC---ATTATTATCCTA--TT-GGTGGCTAATGATA-------AGG--TTAGCACA
Tobacco     TAGGAT-GAGATAAGATTA---CTGAGGTGCTTTA---CACGTGGC---ACCTCCATTGTG--GT-GACTTAAATGAAGA-------ATGGCTTAGCACC
Tea-plant   TCCCAT-ACATTGACATAT---ATGGCCCGCCTGCGGCAACAAAAA---AACTAAAGGATA--GCTAGTTGCTACTACAATTC--CCATAACTCACCACC
Turnip      ATTCAT-ATAAATAGAAGG---TCCGCGAACATTG--AAATGTAGATCATGCGTCAGAATT--GTCCTCTCTTAATAGGA-------A-------GGAGC
Wheat       TATGAT-AAAATGAAATAT---TTTGCCCAGCCA----ACTCAGTCGGCATCCTCGGACAA--TTTGTTATCAAGGAACTCAC--CCAAAAACAAGCAAA
Duckweed    TCGGAT-GGGGGGGCATGAACACTTGCAATCATT----TCATGACTCATTTCTGAACATGT-GCCCTTGGCAACGTGTAGACTGCCAACATTAATTAAA
Larch       TAACAT-ATGATATAACAC---CGGGCACACATTCCTAAACAAAGAGTGATTTCAAATATATCGTTAATTACGACTAACAAAA--TGAAAGTACAAGACC

Cotton      CAAGAAAAGTTTCCACCCTC------TTTGTGGTCATAATG-GTT-GTAATGTC-ATCTGATTT----AGGATCCAACGTCACCCTTTCTCCCA-----A
Pea         C---AAAACTTTTCAATCT-------TGTGTGGTTAATATG-ACT-GCAAAGTTTATCATTTTC----ACAATCCAACAA-ACTGGTTCT---------A
Tobacco     AAAAATAATTTTCCAACCTTT---CATGTGTGGATATTAAG-ATTTGTATAATGTATCAAGAACC-ACATAATCCAATGGTTAGCTTTATTCCAAGATGA
Tea-plant   ATCACACATTCTTCCATTTCATCCCCTTTTTCTTGGATGAG-ATAAGATATGGGTTCCTGCCAC----GTGGCACCATACCATGGTTTGTTA-ACGATAA
Turnip      CAAAAGCATTGGCTCAAGTTG-----AGACGAGTAACCATACACATTCATACGTTTTCTTACAAG-ATAAGATAAGATAATGTTATTTCT---------A
Wheat       GCTAGAAAAAGGTTGTGTGGCAGCCACCTAATGACATGAAGGACT-GAAATTTCCAGCACACACA-A-TGTATCCGACGGCAATGCTTCTTC--------
Duckweed    ATATAATATTAGAAAAAAATC-----TCCCATAGTATTTAGTATTTACCAAAAGTCACACGACCA-CTAGACTCCAATTTACCCAAATCACTAACCAATT
Larch       TTCTCGTATAAGGCCACCA-------TTGGTAGACACGTAGTATGCTAAATATGCACCACACACA-CTATCAGATATGGTAGTGGGATCTG--ACGGTCA

Cotton      ACCAATCTCT---AAATGTT----GTGAGCT---TAG-GCCAAATTT-TATGACTATA--TAT----AGGGGATTGCACC----AAGGCAGTG-ACACTA
Pea         GGCAGTGGCC---AACTAC--------------------CACAATTT-TAAGACCATAA-TAT----TGGAAATAGAA------AAATCAAT--ACATTA
Tobacco     GGGGGTTGTT---GATTTTT----GTCCGTTAGATAT-GCGAAATATGTAAAACCTTAT-CAT----TATATATAGAG------TGGTGGGCA-ACGATG
Tea-plant   GGCTCTTAATCAAAAGTTTTAGGTGTGAATTTAGTTT-GATGAGTTTTAAGGTCCTTAT-TATA---TATAGGAAGGGGG----TGCTATGGA-GCAAGG
Turnip      CACCTTTCTTTAATCCTGTGGCAGTTAACGACGATATCATGAAATCTTGATCCTTCGAT-CATTAGGGCTTCATACCTCT----TGCGCTTCTCACTATA
Wheat       CACTGATCCGGAGAAGATAAGGAAACGAGGCAACCAGCGAACGTGAGCCATCCCAACCA-CATCTGTACCAAAGAAACGG----GGCTATATATACCGTG
Duckweed    TTAGGTTGAATGGAAAATAG---AACGCAATAATGTCCGACATATTTCCTATATTTCCG-TTTTTCGAGAGAAGGCCTGTGTACCGATAAGGATGTAATC
Larch       CGCTTCTCCTCTGGAGTTATCCGATTGTAATCCTTGCAGTCCAATTTCTCTGGTCTGGC-CCA----ACCTTAGAGATTG----GGGCTTATA-TCTATA

Cotton      T-TAAGGGATCAGTGAGAC-TCTTTTGTATAACTGTAGCAT--ATAGTAC
Pea         TATAAAGCAAGTTTTAGTA-CAAGCTTTGCAATTCAACCAC--A-AGAAC
Tobacco     CATAGACCATCTTGGAAGT-TTAAAGGGAAAAAAGGAAAAG--GGAGAAA
Tea-plant   TCCTCATCAAAAGGGAAGTGTTTTTTTCTCTAACTATATTACTAAGAGTAC
Larch       TCTTCTTCACAC---AATCCATTTGTGTAGAGCCGCTGGAAGGTAAATCA
Turnip      TATAGATAACCA---AAGCAATAGACAGACAAGTAAGTTAAG-AGAAAAG
Wheat       GTGACCCGGCAATGGGGTCCTCAACTGTAGCCGGCATCCTCCTCTCCTCC
Duckweed    CATGGGGCGACG---CAGTGTGTGGAGGAGCAGGCTCAGTCTCCTTCTCG
```

11

An Exact Algorithm - Tompa & Blanchette - without using global multiple alignment.

The version I'll present here is terribly inefficient - impractically so. Later I'll hint at how you might go about making it practical.

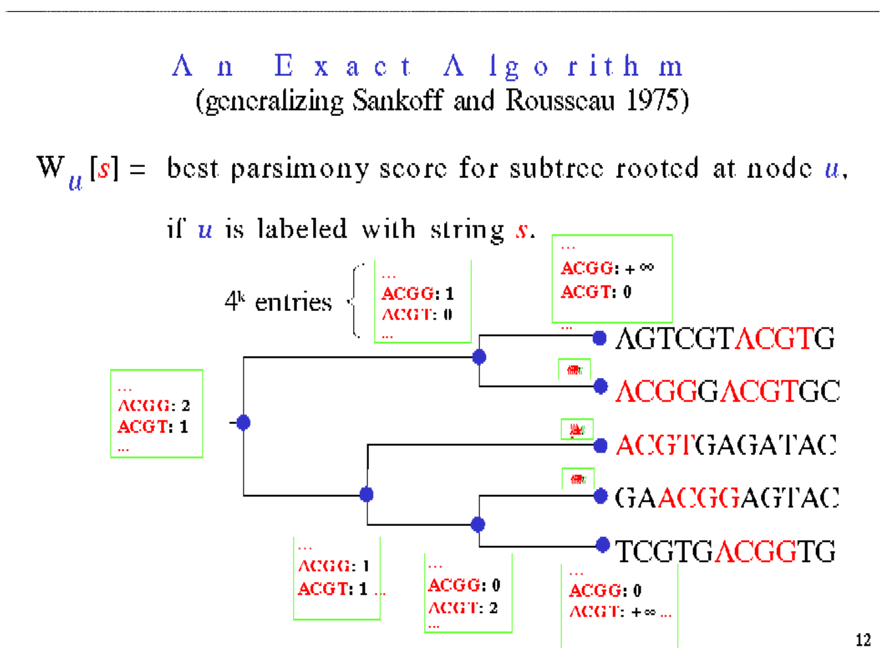
In the examples we'll take $k = 4$, but think of it being about 10.

It's a dynamic programming algorithm. We'll build up the table W, where $W_u[s]$ is the best parsimony score for the subtree rooted at node u, if u is labelled with string s. "If I insist that u is labelled with string s, what would the minimum parsimony score be for the subtree rooted at u?"

The algorithm works from leaves to root. At each node, it's building a table (one-dimensional array). The table at any node can be constructed given the tables of the children. At the root, we just read off the answer.

Now here's the inefficient part: each table has 4^k entries: one for each k-mer in an alphabet of 4 bases. With $k = 10$, each table has a million entries. If $k = 20$, it's not iffy, it's ridiculously impractical.

Referring to the figure below, let's construct the table entry for ACGT. Look at the lower child: if we also labelled it ACGT, then we'd incur no penalty along that branch. The cost is zero plus the cost of ACGT in the child, which is 2, for a total of 2. If we labelled the child ACGG, we incur one mutation along that branch, plus the child cost, which is 0, for a total of 1.



The recurrence is:

$$W_u[s] = \sum_{v:\text{ child of } u} \min_t (W_v[t] + d(s,t))$$

where d is the Hamming distance along the branch.

At leaf nodes, it's very simple. We *observe* the sequences at the leaf nodes, so we can't assign an incompatible label. If a k-mer is a substring of the observed sequence at a leaf, then it has cost zero. Otherwise it has cost ∞.

Why ∞? You can't label a leaf node ACGG if that sequence doesn't exist in that leaf: the sequence at the leaf is an observed quantity.

The running time of the algorithm is $O(k4^{2k})$ time per node. We have to compute 4^k entries, and for each entry have to take the min over 4^k values. Each min requires taking a Hamming distance, which costs k. This gives a total time $O(nk(4^{2k} + l))$, where l is the average sequence length, and n is the number of species.

A better algorithm improves it to $O(nk(4^k + l))$, using a breadth-first search idea. Mathieu realized that you could keep a sparse representation of the tables. Since we want parsimony at most d, once any table entry exceeds d, we don't need to keep it around. There are stronger pruning bounds. This leads to an algorithm that is exponential in d and polynomial in k.

"Q: you expect d to increase up the tree. Can you use that? Doesn't d get big in large families?"

"A: in large trees, we don't really expect very strongly conserved regions throughout the whole tree. You're probably be more interested in motifs conserved in large subtrees. The basic formulation doesn't allow this, but we did an extension. We want motifs that occur in a lot of places, but have low parsimony score; these are competing goals. We let the user specify that they want, eg, parsimony 3 in subtrees of size 20 (measured in units of user-supplied branch length). A generalization of the algorithm works for this more general problem."

The tool that implements this algorithm is called **FootPrinter**. A web version is available: `http://bio.cs.washington.edu/software.html`.

"Q: how well does it perform compared to other tools?"

"A: come to this afternoon's talk. The answer depends very much on the application. This afternoon I'll tell you about a case in which it's terrible. Part of the problem is that it doesn't care about the position of the sequence: it can say that a sequence at the beginning of one sequence matches a sequence at the end of another sequence."

"It's a tough problem. Given a bunch of known binding sites, sometimes it's hard for a biologist to see what's in common between them."