

Minimal Conflicting Sets in ancestral genome reconstruction

Tamon Stephen



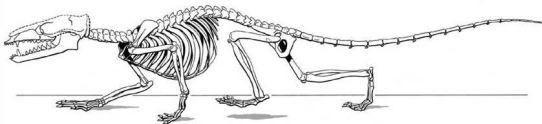
SIMON FRASER
UNIVERSITY

joint work with

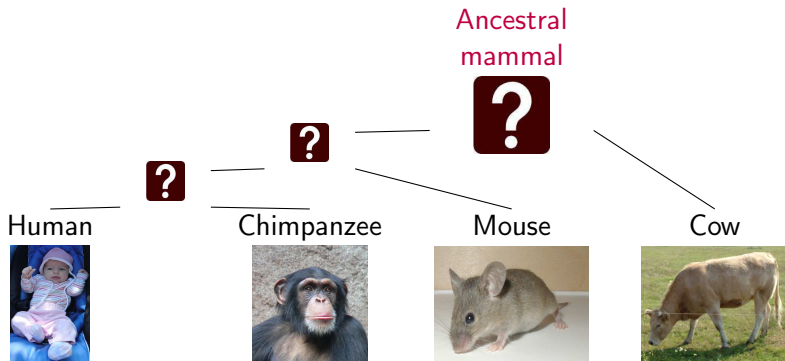
Cedric Chauve, Utz-Uwe Haus and Vivija You

Université Pierre et Marie Curie, December 16th, 2010

- Ancestral Genome Reconstruction.
- Conflicting Sets and the Conflicting Index.
- Monotone Boolean Function Generation.
- Simulated Data Experiments.
- Real Data Experiments.
- Maximal Reconstructions.
- Conclusions.



Ancestral Genome Reconstruction



We are interested in deducing information about the genomes of **ancestral species** based on the genomes of species descended from them.

- Consider the situation where we have identified several **genetic markers** that appear uniquely in each of the extant genomes.
- Assume that these are present in ancestral genome, but that mutations may have changed the order of some of the markers.
- We would like to find the **order** of the markers in the ancestral genome.
- We know which markers are adjacent in the extant genomes, and expect that the order of the markers in the ancestral animal will remain the same except for a few mutations.

Problem setup

- An **ancestral synten**y is a set of markers that are thought to have been consecutive in the ancestor.
- The set of all ancestral syntenies can be encoded in a binary matrix M . The following matrix represents 4 markers and 5 syntenies:

	m_1	m_2	m_3	m_4
s_1	1	1	1	0
s_2	1	0	1	1
s_3	1	0	0	1
s_4	0	1	1	1
s_5	0	0	1	1

- Note that in this case, no ordering of the markers is consistent with all the syntenies.

The Consecutive Ones Property

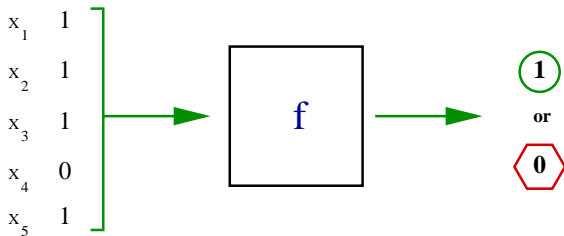
- If all syntenies are correct, then the columns of M can be rearranged so that the ones on each row are consecutive. If this is possible, M is said to have the **consecutive ones property (C1P)**.
- There is an excluded minor characterization on C1P matrices and a non-trivial linear time algorithm to test if a matrix is C1P.
- If a matrix is not C1P, then at least some of the rows must represent “false positive” syntenies.
- We would like to identify these false positives.
- One possible approach is to look for the largest C1P subset of the rows. However, this may not be unique, and in any case is NP-complete.

Conflicting Sets

- If a subset of the rows defines a matrix that is not C1P, we call it a **conflicting set**.
- A conflicting set of rows is called a **minimal conflicting set** if any proper subset of the row is C1P.
- One approach to reconstructing the ancestral genome is to identify rows that are likely “false positives” by looking at the **conflicting index**, the number of minimal conflicting sets containing the row.
- Computing the conflicting index is $\#P$ hard. Nevertheless, we believe that in some reasonable cases it is computable by enumeration.

Boolean Functions

- The property of being conflicting, i.e. **not C1P** is an example of a **boolean function**: its inputs are binary (a row may or may not be included) and its output is binary (a set of rows may or may not be conflicting).



- Boolean functions are a simple and widely applicable model.

Monotone Boolean Functions

This boolean function is **monotone** in the sense that if a set of rows is conflicting, it will remain so if more rows are included. Many useful boolean functions are monotone, but some are not.

Monotone:

AND	0	1
0	0	0
1	0	1

OR	0	1
0	0	1
1	1	1

Not Monotone:

XOR	0	1
0	0	1
1	1	0

Representing Monotone Boolean Functions

- Monotone boolean functions can be described in terms of either their minimal true or maximal false **clauses** (sets).
- For instance, the **AND** function has a single minimal true clause, $x_1 = 1, x_2 = 1$, i.e. $(1,1)$; it is also characterized by its two maximal false clauses: $(1,0)$ and $(0,1)$.
- The **OR** function has two minimal true clauses, $(1,0)$ and $(0,1)$; and one maximal false one: $(0,0)$.
- These minimal true and maximal false clause descriptions are in **dual** descriptions of the same function.
- **Question:** Given a monotone boolean f as a black box (oracle), can we efficiently generate its minimal true clauses?
- In genome reconstruction problem, this is the list of minimal conflicting sets.

Monotone Boolean Functions are Sperner Hypergraphs

- A **hypergraph** consists of a base set of **vertices** and **edges** which are subset of the vertices. It is **Sperner** if there are no nested edges.
- Given a monotone boolean function f , we can define a Sperner hypergraph whose vertices are the variables, and whose edges are the minimal true clauses.

AND



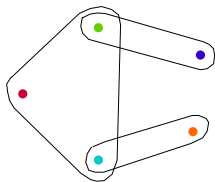
OR



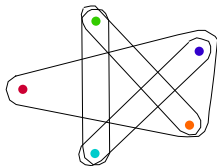
- Similarly, any Sperner hypergraph defines a monotone boolean function.

Dual Hypergraphs and Monotone Boolean Functions

The two dual representations of a monotone boolean function have a nice interpretation as **dual** hypergraphs in the sense that one is obtained from the other by finding family of minimal sets intersecting all hyperedges. Such pairs of hypergraphs are also called **transversal**.



\mathcal{H}



$\text{tr } \mathcal{H}$

The Algorithm of Fredman and Khachiyan

- Fredman and Khachiyan (1996) proposed a novel recursive algorithm for computing the dual of a monotone boolean function.
- It can be modified to produce both a function and its dual from an oracle evaluating the function.
- The algorithm works in time $m^{\text{poly}(\log m)}$, where m is the combined size of the function and its dual.
- If the function is given as an oracle, a complete description of the dual is required to verify the completeness of the description of the (primal) function.
- The algorithm of Fredman and Khachiyan is poorly understood in practice. The worst-case is not always realized.

Implementation

- Our interest in generation of boolean functions arose in another application (in metabolic networks).
- We found few publicly available implementations of this algorithm, and these had oracles hard-coded to a given application.
- We have an open-source implementation of the algorithm that works from a user-supplied oracle:
<http://primaldual.de/cl-jointgen/>
- It is written in Common Lisp to take advantage of the heavily recursive nature of the algorithm.
- For this application, the oracle is a C program which tests if a given matrix is C1P.
- We used CL-Jointgen to generate all minimal conflicting sets on both simulated and real data.

Experiments on Simulated Data

- Our simulated data consists of an “ancestor” of 40 genomic markers, labelled 1 to 40, with consecutive numbers adjacent.
- This gives us 39 “true positive” ancestral syntenies consisting of adjacent pairs.
- We add 6 “false positive” ancestral syntenies uniformly at random spanning between 2 and g markers for $g = 2, 5, 10$.
- We generated 10 data sets for each g .
- For each date set we computed the full list of minimal cut sets (MCS). This allows us to compute the **Conflicting Index** and **Conflicting Ratio (CR)** of each syntyeny. These are just the total number and percentage of MCS’s containing the syntyeny.
- We computed similar statistics for the dual.

Results on simulated data

g	Min. # MCS	Max. # MCS	Avg. # MCS	Min. size dual	Max. size dual	Avg. size dual
2	17	25	21.4	464	3120	1393.2
5	16	29	22.4	1360	10800	4927.9
10	22	46	32.4	601	16954	5796.8

- For this simulated data, the existing joint generation code was able to generate all minimal conflicting sets in a reasonable time (minutes to hours).
- In fact we typically see very few minimum conflicting sets.
- The duals are somewhat larger, but not so large as to seriously impede generation.

Effectiveness of conflicting ratio

Dataset	Avg. FP_CR	Avg. TP_CR	Avg. FP_dR	Avg. TP_dR	Avg. FP MCS rank	Avg. FP Md rank
2	0.20	0.05	0.66	0.82	39.65	15.47
5	0.21	0.07	0.69	0.79	39.53	17.72
10	0.27	0.11	0.62	0.81	38.20	14.02

- The simulated data suggests that the conflicting index is a useful indicator for false positives (FP): false positives typically had a much higher conflicting index than true positives (TP).
- The duals also provide information about false positives, as they tend to appear less often in dual clauses. The effect is less pronounced.

Experiments on real data

- We considered a real dataset, used to reconstruct an ancestral *Saccharomycetaceae* genome from non-duplicated yeasts genomes (*S. kluyveri*, *K. thermotolerans*, *K. lactis*, *A. gossypii* and *Z. rouxii*).
- These genomes are represented with 1420 markers and a total of 3106 ancestral syntenies were computed, giving a binary matrix M with 3106 rows and 1420 columns. From this large matrix, five submatrices $\{M_1, \dots, M_5\}$ were extracted that contained all MCS.
- The matrices M_4 and M_5 are fairly large, with more than 800 rows and 300 columns.

Results on real data

Matrix	# MCS	# dual	# reliable ancestral syntenies
M_1	6	4	6
M_2	402	13	166
M_3	2214	90	199
M_4	1976*	483*	617
M_5	1277*	883*	1291

- Reliable ancestral syntenies are the syntenies that are observed every extant species.
- The large matrices M_4 and M_5 had not completed after 3 days, so the computation was interrupted and partial results were reported.
- For the large matrices, the oracle calls were quite slow.
- Curiously, in this case the number of MCS's were less than the size of the dual.

Maximal C1P Submatrices

- The natural monotone boolean function structure of conflicting sets draws our attention to the dual function of the minimal conflicting sets.
- This turns out to be the function that identifies whether the complement of a set of rows form a C1P matrix.
- Thus its minimal clauses are the complements of the *maximal* C1P submatrices.
- Indeed, the maximal C1P submatrices are candidate reconstructions of the ancestral genome.
- These candidate reconstructions are a useful byproduct of the conflicting index calculation.

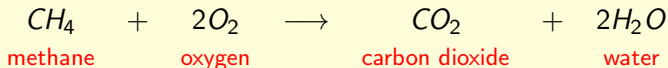
- The **conflicting index** is useful in discriminating between true positive and false positive syntenies in ancestral genome reconstruction.
- One approach to compute the conflicting index is via joint generation of monotone boolean functions.
- We advertise the open-source implementation of Fredman and Khachiyan's generation algorithm for such functions:
<http://primaldual.de/cl-jointgen/>
- Boolean functions are a nice modelling tool in complex systems.
- In our experiments the linear-time solvable C1P problem is a worse bottleneck than joint generation.
- When a monotone boolean function appears, it is worth looking at its dual.

- Is there a reason that the synthetic data has large duals, but the real data has small duals?
- Can we efficiently test if there is *any* minimal conflicting set containing a given row?
- Can random or partial generation be used to estimate the conflicting index?
- Can we deepen our understanding of the joint generation algorithm?
- Is there a way to take advantage of the special structure of C1P in computing the conflicting index instead of using joint generation?
- Further characterizations and identifications of false positive ancestral syntenies.

Thank you!

Bonus: Metabolic Networks

- Set of co-dependent metabolic processes (i.e. chemical reactions) active in a cell.
- Reactions are characterized by their **stoichiometry**, the relationship between the **metabolites** (molecules) they produce and consume. For instance:



- In a steady state, the amounts of metabolites produced and consumed by the various reactions must balance.

Example: Growth in *e.coli*

- Growth in *e.coli* bacteria is a complex process involving dozens of metabolites. It is modelled as a single reaction converting a fixed set of precursors into biomass.
- To sustain growth, supplies of all these precursors are required.
- We use a detailed model of the *e.coli* cell that has 89 metabolites with 110 possible reactions on them.
- We would like to deduce which sets of reactions **support** biomass synthesis.
- Similarly, we would like to know which sets of reactions are **required** for biomass synthesis.

Problem setup

- A model of network of k metabolites with q reactions can be recorded as $k \times q$ **stoichiometric matrix** N .

	m_1	m_2	m_3	m_4	m_5	m_6
r_1	1	2	-1	-2	0	0
r_2	0	-1	-1	0	1	1
r_3	1	0	0	1	-2	0
r_4	0	1	0	1	0	-3
r_5	-2	0	3	1	0	-2

- Some reactions are reversible: it is possible to produce a given output from a given input, or vice-versa, but most are not.
- We can record in a vector x the relative frequencies of active reactions in a steady-state. Inactive reactions will have 0 coefficient; reversible reactions may have negative coefficients.
- Thus the possible **modes** representing steady states of the network are:

$$\{x \in \mathbb{R}^q \mid Nx = 0, x_i \geq 0 \forall i \in U\} \quad (1)$$

where U is the set of irreversible reactions.

Elementary Modes

- Non-negative linear combinations of modes are again modes, so the system (1) usually has many solutions. These can be decomposed into sums of support minimal non-zero solutions, known as **elementary modes** (EMs).
- These are the simplest functioning subsystems of the network, and as such are of interest to biologists.
- The EMs are, up to a constant multiple, the extreme rays of the cone (1). They can be computed using variants of the double description method.
- Double description may work well, but has a poor worst-case complexity and is not trivial to implement.
- Elementary modes are characterized up to a scalar multiple by the binary vector representing their support pattern.

Minimal Cut Sets

- Closely related to the concept of elementary modes are **cut sets**. These are subsets of the reactions without which the network cannot operate in a steady state. For a cut set C , the system:

$$\{Nx = 0, x_i \geq 0 \forall i \in U, x_c = 0 \forall c \in C\} \quad (2)$$

has only the trivial solution $x = 0$.

- Note that the relation of being a cut set also defines a monotone boolean function.
- As with modes, we focus on support **minimal cut sets** (MCSs). These are the smallest combinations of reactions that we need to shut down the system.
- The list of such possible “knock out” strategies is particularly useful in the design of experiments.

Thank you!