



Combinatorics in computational biology: strings and permutations

Steven Kelk, CWI

Email: S.M.Kelk@cwi.nl

Web: <http://homepages.cwi.nl/~kelk>



Introduction

- The increasingly sophisticated use of computers in biology allows enormous quantities of biological data to be efficiently collected.
- But data alone is of limited use. What does the data mean? What useful information can we (efficiently) infer from such vast amounts of data?
- Often it is not too difficult to think of naïve algorithms for processing and interpreting biological data. But such naïve algorithms are often cripplingly slow and/or space-consuming. Thus, efficient algorithm design, underpinned by combinatorial analysis to obtain provable bounds on performance, is necessary if we are to move beyond heuristics.



1. **Identify** biological problem.
2. Create mathematical **abstraction** (a combinatorial optimisation problem.)
3. **Analysis.** Is this problem (quickly) solvable in polynomial time? NP-hard? How hard is it to approximate? And so on.
4. Next steps:
 1. How biologically realistic is the model? Addition of further **biological constraints** can make problem easier or harder. (Back to steps 1 & 2.)
 2. "Spin-off" problems for mathematicians.



Example 1. Inferring complete haplotypes of a (diploid) individual from partial haplotype data.

- DNA as a string over nucleotide alphabet {A,C,G,T}.
- At most sites on our DNA, we have same nucleotide. But at some sites variation is observed. These sites are called *Single Nucleotide Polymorphisms* (SNPs.)

Me: ...AAGGCTAA.....AAGTGTAC...
You: ...ATGGCTAA.....AAGCGTAC...


- Positions of SNPs assumed to be known. Thus, focus only on SNP locations to obtain *haplotype* data (about 100-300 times shorter than DNA.)

Me: AT
You: TC

- Furthermore: most SNPs are 2-state, so assume (by relabelling) that the haplotype is a string over a binary alphabet.




- Diploid organisms (e.g. humans): two chromosomes covering each interval of the genome. So for a given interval of the genome, there are TWO distinct haplotypes (i.e. two binary strings.)
- **Problem:** For a given region of DNA, reconstruct two, complete haplotypes of an individual by combining short, incomplete and error-prone haplotype fragments obtained from that individual.
- Crucially: we DO NOT know which chromosome each haplotype fragment came from!



1	0	1	-	-	-
-	0	1	1	0	-
0	0	0	-	-	-
-	0	0	0	1	-
-	-	-	1	0	0
-	-	0	0	1	1

Example input

- Each row is an aligned haplotype fragment from the individual; each column is an SNP.
- A '-' symbol, called a *hole*, indicates absence of data.
- Two rows *conflict* if they disagree at a given location e.g. 4th and 5th row conflict, 3rd and 4th row do not conflict. Rows thus induce a *conflict graph*.
- What were the two original haplotypes?



1	0	1	-	-	-
-	0	1	1	0	-
0	0	0	-	-	-
-	0	0	0	1	-
-	-	-	1	0	0
-	-	0	0	1	1

- In other words: can we partition the rows into two sets, such that – within each set – no two rows conflict with each other?
- In simple cases like this (i.e. when the conflict graph is bipartite), the answer is yes. Once we have obtained this partition, we can reconstruct the haplotypes by merging rows.
- But in general there will be errors in the 0/1 input data, and then a conflict-free bipartition might not be possible. What do we do then?
This depends on the model we choose.

1	0	1	1	0	0
---	---	---	---	---	---

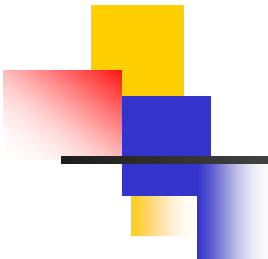
First haplotype

0	0	0	0	1	1
---	---	---	---	---	---

Second haplotype



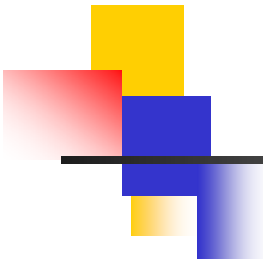
- Different abstractions cope with errors in different ways.
- Here I demonstrate the abstraction known as the *Longest Haplotype Reconstruction* (LHR) problem.
- Here we are allowed to **discard** an arbitrary number of data fragments (i.e. **rows** of the matrix) until a conflict-free bipartition of the remaining rows is possible. We want to maximise the **sum of the lengths** of the two haplotypes induced by this bipartition, i.e. minimise the number of sites where the haplotypes have holes. (Holes appear in the final haplotypes where there are no 0s or 1s in that column of the corresponding partition.)
- The underlying motivation is that we want to find a feasible solution that maximises the amount of information i.e. tells us the most about the individual's two haplotypes.



1	0	1	-	-	-
-	0	1	1	0	-
-	1	1	0	-	-
0	0	-	-	-	-
-	-	-	-	0	0
-	-	-	0	1	1

- You cannot produce a conflict-free bipartition of this matrix (e.g. 1st, 3rd and 4th rows are in mutual conflict.)
- But if we remove the third row...

Example input matrix, with read errors



1	0	1	-	-	-
-	0	1	1	0	-
	1	1	0		
0	0	-	-	-	-
-	-	-	-	0	0
-	-	-	0	1	1

- You cannot produce a conflict-free bipartition of this matrix (e.g. 1st, 3rd and 4th rows are in mutual conflict.)
- But if we remove the third row...

Example input matrix, with read errors



1	0	1	-	-	-
-	0	1	1	0	-
	1	1	0		
0	0	-	-	-	-
-	-	-	-	0	0
-	-	-	0	1	1

- You cannot produce a conflict-free bipartition of this matrix (e.g. 1st, 3rd and 4th rows are in mutual conflict.)
- But if we remove the third row...
- ...*then* a conflict-free bipartition is possible.
- Sum of the lengths of the two induced haplotypes is 11. (Optimal in this case is actually 12; can you see how to achieve this?)

1	0	1	1	0	0
---	---	---	---	---	---

First haplotype (length 6)

0	0	-	0	1	1
---	---	---	---	---	---

Second haplotype (length 5)



- It turns out that the complexity of the LHR problem is linked to the maximum number of *gaps* in the input rows. A gap is a contiguous block of holes, flanked on both sides by 0s/1s.

1	0	1	1	0	0
---	---	---	---	---	---

No gaps

-	0	1	1	-	-
---	---	---	---	---	---


No gaps

1	-	1	1	0	0
---	---	---	---	---	---

1-gap

1	-	1	-	-	0
---	---	---	---	---	---

2-gaps

- 
- Good news:- If all rows of the input matrix are gapless, then LHR is solvable in polynomial-time by dynamic programming.
 - Bad news:- If there is allowed to be at most one gap per input row, LHR becomes NP-hard and APX-hard by approximation-preserving reduction from CUBIC-MAXIMUM-INDEPENDENT-SET.
 - Informally:- it is (in general) difficult to efficiently obtain an exact solution for LHR, and arbitrarily good constant-factor approximation algorithms are also not possible.
 - Such mixed news is of course not the end of the story, but only the beginning...
 - Full details (not just of LHR but of other error-correcting models) available in *"On the complexity of the Single Individual SNP Haplotyping Problem"*, Cilibrasi, van Iersel, Kelk, Tromp (2005).



Example 2. Inferring haplotypes of a *population* from its observed genotypes, under assumption of pure parsimony

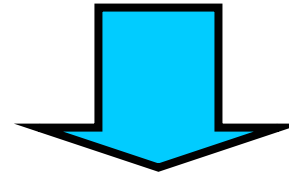
- Sometimes our raw input data is *genotype* information, not haplotype information. Abstractly, genotype data is a string over the $\{0,1,2\}$ alphabet.
- In a diploid organism, the genotype is *induced* by the two haplotypes corresponding to that region. A '2' indicates that the individual is heterozygous at that site (i.e. 0/1 or 1/0.) We call this an *ambiguous* site.

First haplotype

1	0	1	0	1	0	1	1
---	---	---	---	---	---	---	---


Second haplotype


1	0	0	1	1	0	1	0
---	---	---	---	---	---	---	---



Genotype

1	0	2	2	1	0	1	2
---	---	---	---	---	---	---	---

- 
- A genotype g is *explained* by two haplotypes if those two haplotypes induce g . However, a genotype g does not, in general, uniquely specify the haplotypes that induced it i.e. genotypes carry less information than pairs of haplotypes.
 - E.g. genotype **122** could be explained by $\{100,111\}$ or $\{101, 110\}$.
 - Suppose we have a population of individuals, and for each individual we observe its genotype. (We assume the genotype is error-free and complete.)
 - Under the assumption of pure parsimony (Informally:- that many of the individuals share common haplotypes) we seek to find the **smallest set of haplotypes** such that every genotype is explained by *some* pair of the haplotypes.
 - This is the *Pure Parsimony Haplotyping* (PPH) problem.

- 
- For example, suppose we observe genotypes **122**, **201**, **022**.
 - Then a smallest set of haplotypes that can explain these genotypes has cardinality 4 e.g. **101**, **001**, **110**, **010**.
 - **122** comes from {**101**, **110**}, **201** comes from {**001**, **101**}, and **022** comes from {**010**, **001**}.
 - But in general...
 - Bad news:- if up to 3 ambiguous sites (i.e. 2s) are allowed per genotype, PPH becomes NP-hard and APX-hard. (Lancia et al, 2004)
 - Good news:- if up to 2 ambiguous sites are allowed per genotype, PPH becomes (non-trivially) solvable in polynomial time.
 - Key observation is that the problem is related to the INDEPENDENT SET problem restricted to bipartite graphs, a polynomial-time solvable problem.

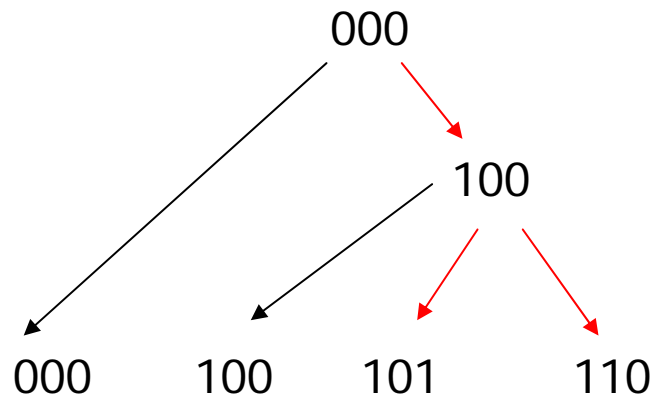


Example 3. Phylogenetics

- Phylogenetics is a (huge!) meta-field.
- Embraces the biologically-realistic idea that genetic diversity does not simply “appear”, but is the consequence of variation introduced by biological processes such as reproduction, mutation and evolution.
- Often forms a biologically-relevant restriction on mathematical abstractions that would otherwise be simply too abstract.
- For example:- *perfect phylogeny*.
- A set of binary strings has a perfect phylogeny if the strings can be written (under certain restrictions) as the leaves of an ‘evolutionary tree.’
- Demanding that a data set permits a perfect phylogeny is like saying, “*make sure that the diversity within the data set can be plausibly explained by known biological phenomena, such as mutations.*”



- For example, consider the strings 000, 100, 101 and 110. Do they allow a perfect phylogeny (under the assumption that the ancestor is 000, that each site mutates at most once, and that no recombination occurs?) Yes:



- Red arrows denote a site mutation from 0 to 1.

- But not all sets of strings permit a perfect phylogeny! Simple example:- the four strings 00, 01, 10 and 11. (Some site must mutate twice.)



- Often forms a biologically-relevant restriction on mathematical abstractions that would otherwise be simply too abstract.
- Example from earlier: For a given set of genotypes, find the smallest set of haplotypes which explains the genotype data *and also such that the haplotypes permit a perfect phylogeny* - MPPH.
- Perfect phylogeny is only one branch of phylogenetics...
- We are starting to work on phylogenetic networks. These are generalisations of phylogenetic trees where *recombination* is allowed (i.e. evolutionary paths can re-intersect) and *homoplasy events* (i.e. sites can mutate more than once) can be modeled. Such models are at the frontier of combinatorics in biological analysis. (Dan Gusfield et al)
- Members of our group have also worked with fungi specialists in the area of phylogenetic analysis.



Example 4. Computing minimum rearrangement scenarios between two genomes (reversals etc.)

- Many genomes are more closely related than they first appear.
- Often the major differences between the genomes of two species can be explained by a relatively small number of large-scale rearrangement events, in which large chunks of the genome are shifted around and/or reversed.
- We can model n genes in a genome by a (signed) permutation on n elements, where the sign of a gene denotes which way it points.

-3 +1 +2 -5 +4

- If we have two species A and B that have the same set of genes, but some of the genes occur in different places, and some have been reversed, a natural question (inspired by parsimony) is as follows:

What is the smallest number of large-scale rearrangement events that can explain the transformation of genome A into genome B?




- The first, and most famous, abstraction, considered only *reversal* events.
- This inspired the combinatorial problem *Sorting By Reversals* (MIN-SBR).
- “Sorting” because we may assume that B is the (positive) identity permutation. In each move, we are allowed to reverse an arbitrary-sized substring of the permutation. Elements within the reversed substring change sign.

• Toy example:

A:	-3	<u>+1</u>	<u>+2</u>
	<u>-3</u>	<u>-2</u>	<u>-1</u>
B:	+1	+2	+3

(At least 2 reversals are needed to transform A into B.)

- MIN-SBR famously proven (by Hannenhalli and Pevzner, 1995) to be solvable in polynomial-time: “*Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals*”.
Unsigned version shown NP-hard in 1998 by Caprara. (Also spectacular!)

- 
- There have been (and continue to be) many papers written on this theme, which we might broadly describe as *sorting permutations by rearrangements*.
 - Still many fascinating *rearrangement* fields to explore. For example:-
 - Modeling different rearrangement events (e.g. transpositions, fissions, fusions, transversals), and combinations of such events.
 - Adding biologically-motivated restrictions to the rearrangement events (e.g. addition of weighting, requiring that common gene intervals are preserved, and so on.)
 - More recently, “spin-off” papers have started to appear which study rearrangement operations not only on permutations, but on strings (i.e. where symbols can be repeated.)
 - A subset of our group is busy completing a paper which examines the behaviour of so-called “prefix reversals” on strings. **To appear soon!**



And finally...

Good places to start if you wish to explore the literature in this area are the following two conferences:-

- Research in Computational Molecular Biology (RECOMB)
- Workshop on Algorithms in Bioinformatics (WABI)



And finally...

Good places to start if you wish to explore the literature in this area are the following two conferences:-

- Research in Computational Molecular Biology (RECOMB)
- Workshop on Algorithms in Bioinformatics (WABI)

Thankyou for listening!

Steven Kelk, CWI
Email: S.M.Kelk@cwi.nl
Web: <http://homepages.cwi.nl/~kelk>