Graphs, permutations and sets in genome rearrangement

Anthony Labarre[1]
alabarre@ulb.ac.be

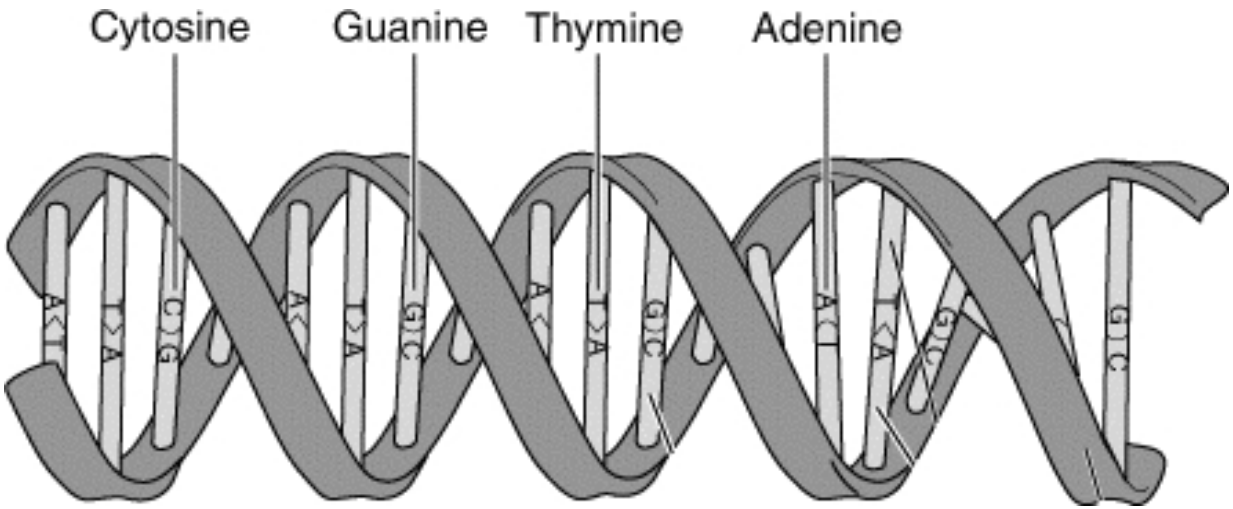
Université Libre de Bruxelles

February 6, 2006

Computers in Scientific Discovery III

[1]Funded by the "Fonds pour la Formation à la Recherche dans l'Industrie et dans l'Agriculture" (F.R.I.A.).

A few biological definitions



- Life = DNA;
- DNA = double helix of *nucleotides* (A, C, G and T);
- *Genes* = sequences of nucleotides;
- *Chromosome* = (ordered) set of genes;

Sequence alignment

- Comparison at the nucleotide level;
- Example:

$$
\begin{array}{ccccccccccccccc}
\cdots & T & C & C & T & G & C & C & A & T & - & - & C & T & A & \cdots \\
 & | & | & & | & | & & | & | & | & & & | & & | & \\
\cdots & T & C & G & T & G & A & C & A & T & G & G & C & - & A & \cdots
\end{array}
$$

- Matches, differences, insertions and deletions;

Genome rearrangement

- Comparison at the gene level;
- Species differ not only by "content", but also by order:
  - genes spread over different sets of chromosomes;
  - genes ordered differently on the same chromosome;
- Example:
  - many genes in cabbage and turnip are 99% identical;

General statement of the problem

- The problem to solve can be summarized as:

Given two (or more) genomes, find a sequence of mutations that transforms one into the other and is of minimal length.

- Different assumptions yield different models:
  - gene order;
  - gene orientation;
  - duplications/deletions in the genome;
  - mutations taken into account;
  - weights given to mutations;
  - miscellaneous restrictions;

Known gene order: permutations

- Assumptions:
  - gene order is known;
  - each gene appears exactly once in each genome;
- Therefore:
  - $\{\text{genes}\} = \{1, 2, \ldots, n\}$;
  - genome = *permutation* of $\{1, 2, \ldots, n\}$;
- One or several operations;
- "Computing the X distance" $\equiv$ "Sorting by X's";

Sorting by transpositions [Bafna and Pevzner, 1995]

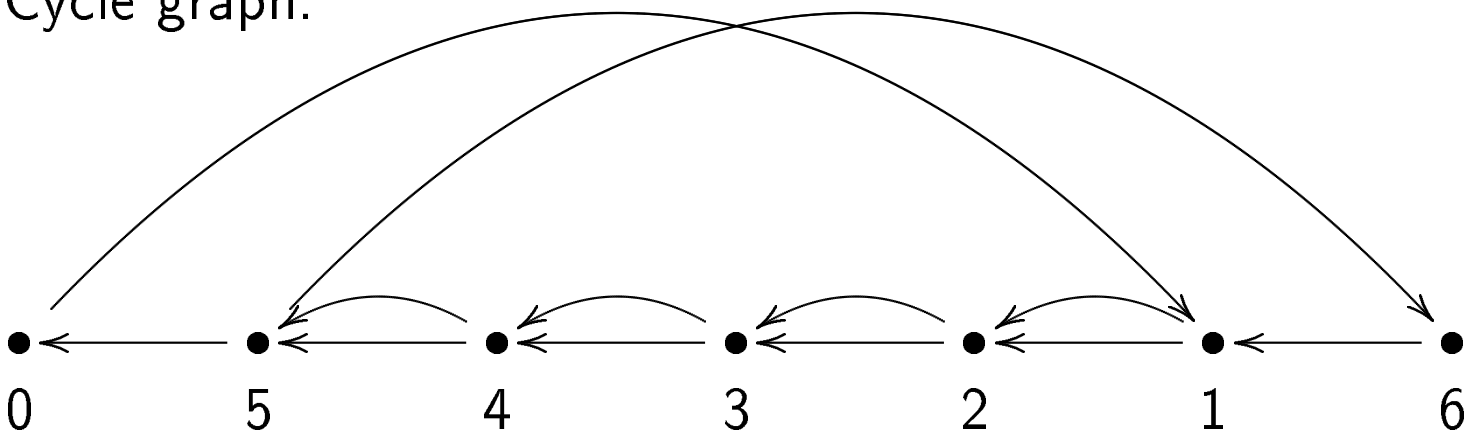
- A *transposition* exchanges adjacent intervals:

$$(5 \; \boxed{4 \; 3} \; \boxed{2 \; 1})$$

$$\downarrow$$

$$(5 \; \boxed{2 \; 1} \; \boxed{4 \; 3})$$

- Cycle graph:



- Complexity and diameter are unknown;

Sorting by transpositions: example

$$(\boxed{5\ 4}\ \boxed{3\ 2}\ 1)$$

$$\downarrow$$

$$(3\ \boxed{2\ 5}\ \boxed{4\ 1})$$

$$\downarrow$$

$$(\boxed{3\ 4}\ \boxed{1\ 2}\ 5)$$

$$\downarrow$$

$$(1\ 2\ 3\ 4\ 5)$$

Sorting by transpositions: some personal results

- In [Labarre, 2005]:
  - Classes of permutations for which the distance can be computed;
  - New tight upper bound;
- In [Doignon and Labarre, 2006]:
  - Bijection between cycle graph structures and factorisations of permutations;
  - Formula for the number of permutations with a given cycle graph structure;
- In [Labarre, 2006]: tighter bounds and other tractable classes of permutations;

Genes spread over different chromosomes: sets

- A genome G is defined by:
  - a set of n genes $\{1, 2, \ldots, n\}$, and
  - a collection of k chromosomes C_1, C_2, ..., C_k ;
- Problem: given genomes

$$\begin{cases} G_1 = \{C_{1,1}, C_{1,2}, \ldots, C_{1,k}\} \\ G_2 = \{C_{2,1}, C_{2,2}, \ldots, C_{2,l}\} \end{cases}$$

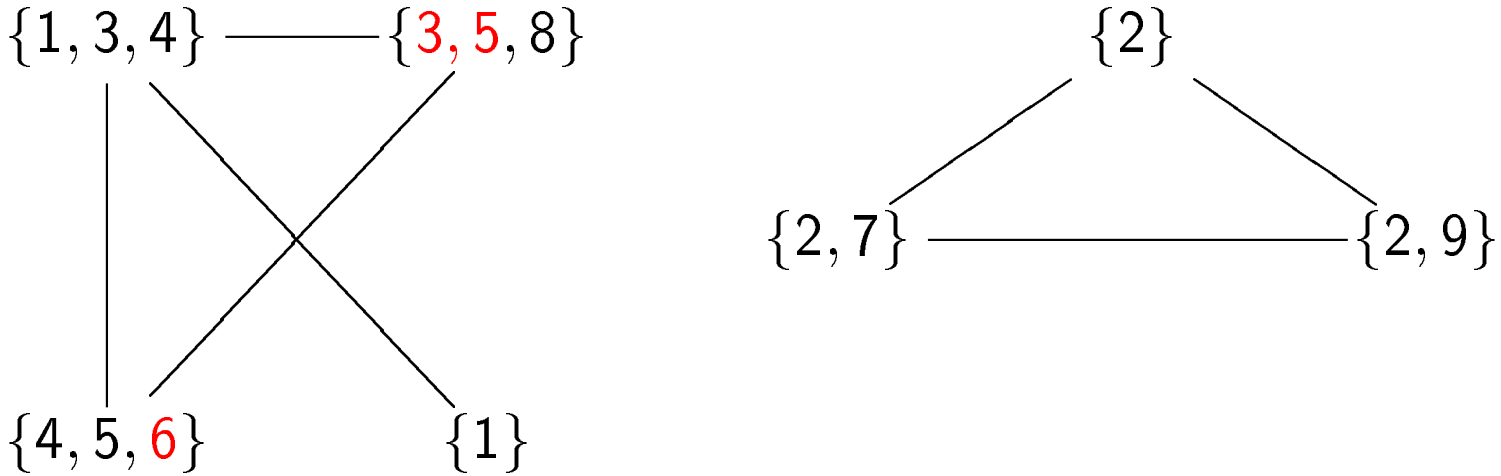
and a set of operations, find the minimum sequence of operations bringing G_1 into G_2;

Syntenic distance [Ferretti et al., 1996]

- Two genes on the same chromosome are "in *synteny*";
- The set of operations consists of:
  1. fissions;
  $$C \to \{C_1, C_2\}$$
  2. fusions;
  $$\{D_1, D_2\} \to D$$
  3. translocations;
  $$\{C_1 \cup C_2, D_1 \cup D_2\} \to \{C_1 \cup D_1, C_2 \cup D_2\}$$
- Canonical form for this problem: Transform genome $G = \{C_1, C_2, \ldots, C_k\}$ into $\{\{1\}, \{2\}, \ldots, \{n\}\}$;
- Problem is **NP**–hard [DasGupta et al., 1998];

Synteny graph

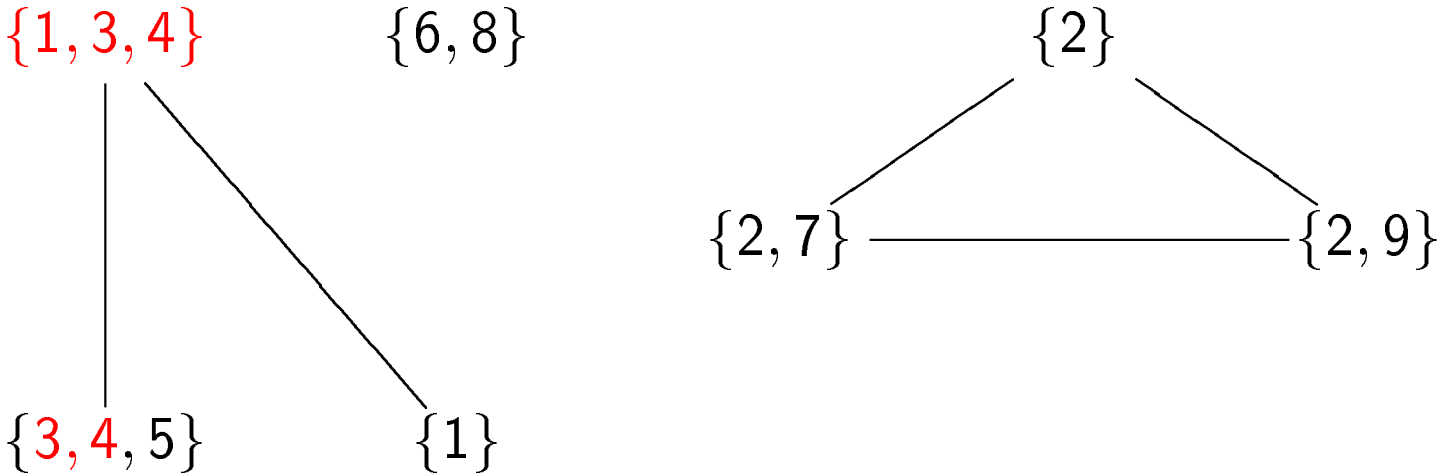
- Vertices are subsets C_i;
- Edges are $\{C_i, C_j\}$ $(i \neq j)$ such that $C_i \cap C_j \neq \emptyset$;



- Goal: eliminate all edges and obtain only singletons;
  - translocations: 0
  - fissions: 0

Synteny graph

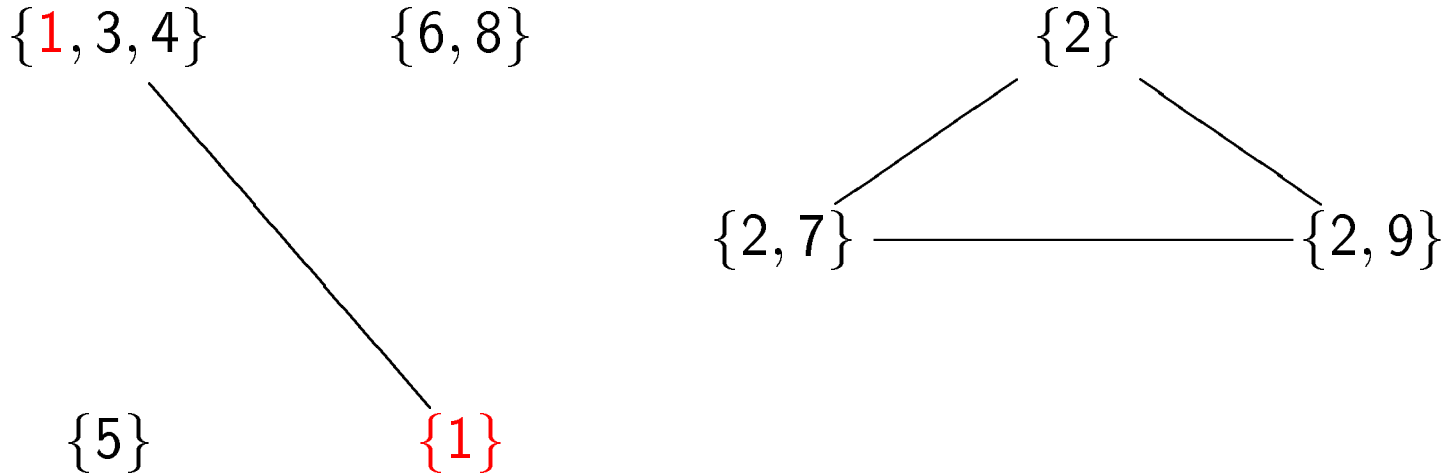
- Vertices are subsets C_i;
- Edges are $\{C_i, C_j\}$ $(i \neq j)$ such that $C_i \cap C_j \neq \emptyset$;



- Goal: eliminate all edges and obtain only singletons;
  - translocations: 1
  - fissions: 0

Synteny graph

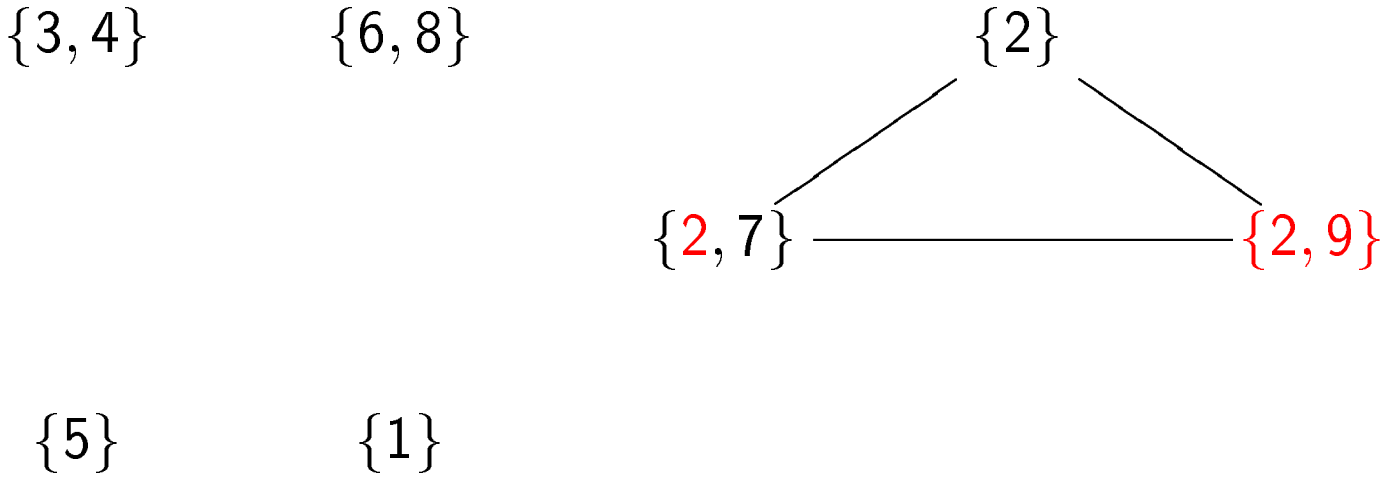
- Vertices are subsets C_i;
- Edges are $\{C_i, C_j\}$ $(i \neq j)$ such that $C_i \cap C_j \neq \emptyset$;



- Goal: eliminate all edges and obtain only singletons;
  - translocations: 2
  - fissions: 0

Synteny graph

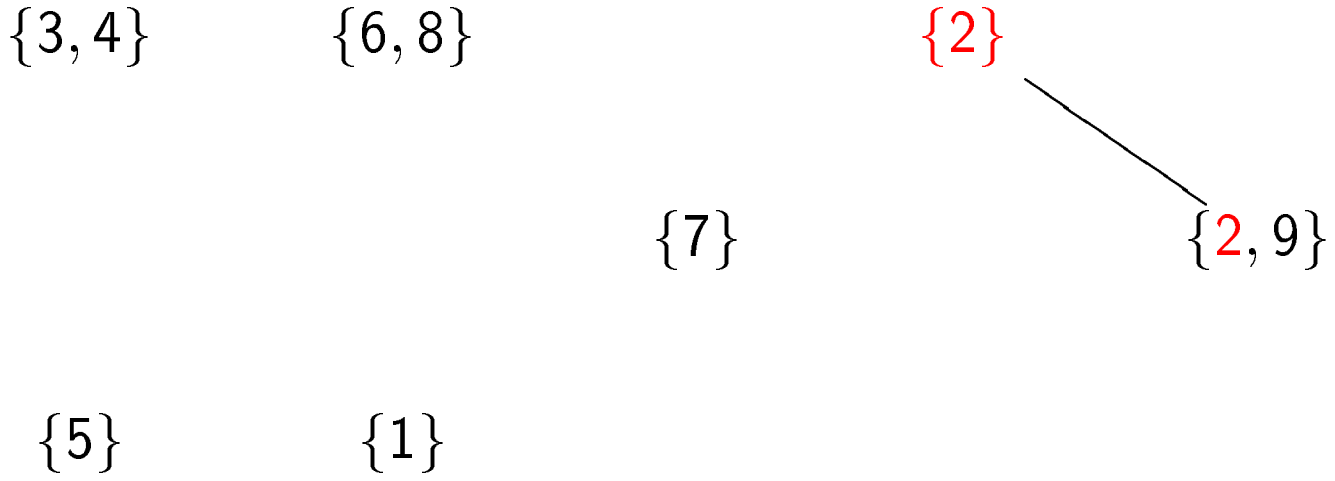
- Vertices are subsets C_i;
- Edges are $\{C_i, C_j\}$ $(i \neq j)$ such that $C_i \cap C_j \neq \emptyset$;



- Goal: eliminate all edges and obtain only singletons;
  - translocations: 3
  - fissions: 0

Synteny graph

- Vertices are subsets C_i;
- Edges are $\{C_i, C_j\}$ $(i \neq j)$ such that $C_i \cap C_j \neq \emptyset$;



- Goal: eliminate all edges and obtain only singletons;
  - translocations: 4
  - fissions: 0

Synteny graph

- Vertices are subsets C_i;
- Edges are $\{C_i, C_j\}$ $(i \neq j)$ such that $C_i \cap C_j \neq \emptyset$;

$\{3, 4\}$ $\{6, 8\}$ $\{2\}$

$\{7\}$ $\{9\}$

$\{5\}$ $\{1\}$

- Goal: eliminate all edges and obtain only singletons;
  - translocations: 5
  - fissions: 0

Synteny graph

- Vertices are subsets C_i;
- Edges are $\{C_i, C_j\}$ $(i \neq j)$ such that $C_i \cap C_j \neq \emptyset$;

$\{3\}\{4\}$ $\{6, 8\}$ $\{2\}$

$\{7\}$ $\{9\}$

$\{5\}$ $\{1\}$

- Goal: eliminate all edges and obtain only singletons;
  - translocations: 5
  - fissions: 1

Synteny graph

- Vertices are subsets C_i;
- Edges are $\{C_i, C_j\}$ $(i \neq j)$ such that $C_i \cap C_j \neq \emptyset$;

$\{3\}\{4\}$ $\{6\}\{8\}$ $\{2\}$

$\{7\}$ $\{9\}$

$\{5\}$ $\{1\}$

- Goal: eliminate all edges and obtain only singletons;
  - translocations: 5
  - fissions: 2

Comparative genomics

- Extensive use of computers; in order to compare a set of species, you need to:
  1. get their genomes:
     - from a database if it has been done;
     - by sequencing them otherwise;
  2. infer their phylogeny:
     - 2.1 compute distances pairwise (can be **NP**-hard);
     - 2.2 reconstruct putative scenarios (exponential number);
     - 2.3 discriminate (exponential number of good scenarios);
  3. make a choice between topologies;
- Use of software for each task;

The On-Line Encyclopedia of Integer Sequences[2]

- "What is the set of all objects that satisfy property P?";
  1. for $k = 0, 1, 2, \ldots$, generate all elements and count those that verify P;
  2. cardinalities form a sequence;
  3. input sequence into the Encyclopedia;
  4. assuming there are matches, try to relate the sets of objects counted;
- Examples:
  - maximal instances for a particular distance;
  - instances with distance k (distribution);
  - graphs with a given structure [Doignon and Labarre, 2006];

[2] http://www.research.att.com/~njas/sequences/

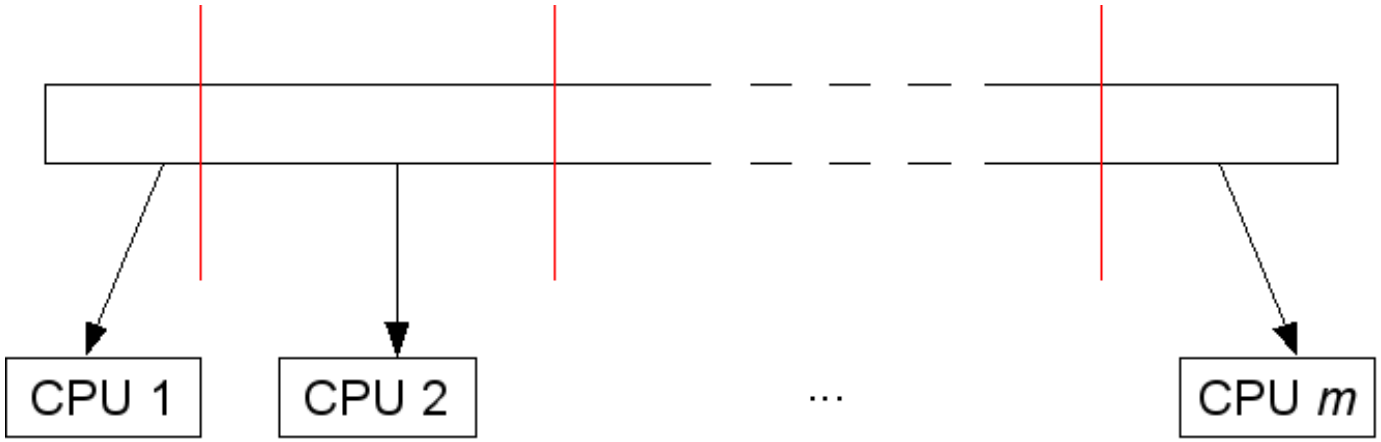
Computer-assisted proofs

- Best approximation ratio for sorting by transpositions is 11/8 [Elias and Hartman, 2005];
- The proof is computer-assisted:
  1. generate cases to test (more than 80,000);
  2. solve cases;
  3. verify solutions;
- Previous notorious examples:
  - the Four Colour Theorem [Appel and Haken, 1977, Appel et al., 1977];
  - the proof of Kepler's conjecture, which is to become even more computer-driven (see papers by Thomas Hales[3]);
- Heated topic;

[3] http://www.math.pitt.edu/~thales/

Parallel computing

- Genomes can be huge;
- So are the running times of exact algorithms for **NP**-hard (GR) problems;
- When possible:
  - partition instances into "independently sortable components";
  - assign each component to a different CPU/machine;



Further possible uses

- Underlying graph problems in genome rearrangement;
  - possible uses of GraPHedron?
- Characterization of special classes of permutations:
  - development of a conjecture-making tool on permutations?
- Efficiently solving GR problems (work by Fertin et al.):
  - SAT is a well-studied **NP**-hard problem;
  - transform instances of GR into instances of SAT;
  - solve GR problem through SAT solvers;

Appel, K. and Haken, W. (1977).
Every planar map is four colorable. I. Discharging.
Illinois J. Math., 21(3):429–490.

Appel, K., Haken, W., and Koch, J. (1977).
Every planar map is four colorable. II. Reducibility.
Illinois J. Math., 21(3):491–567.

Bafna, V. and Pevzner, P. A. (1995).
Sorting permutations by transpositions.
In *Proceedings of SODA*, pages 614–623. ACM/SIAM.

DasGupta, B., Jiang, T., Kannan, S., Li, M., and Sweedyk, E. (1998).
On the complexity and approximation of syntenic distance.
Discrete Applied Mathematics, 88(1-3):59–82.

Doignon, J.-P. and Labarre, A. (2006).
On Hultman numbers.
Submitted.

Elias, I. and Hartman, T. (2005).
A 1.375-approximation algorithm for sorting by transpositions.
In *Proceedings of WABI*, LNBI 3692, pages 204–214.

Ferretti, V., Nadeau, J. H., and Sankoff, D. (1996).
Original synteny.
In *Proceedings of CPM*, LNCS 1075, pages 159–167.

Labarre, A. (2005).
A new tight upper bound on the transposition distance.
In *Proceedings of WABI*, LNBI 3692, pages 216–227.

Labarre, A. (2006).
New bounds and tractable instances for the transposition distance.
Submitted.