Introduction to Biological Data Structures and C Programming

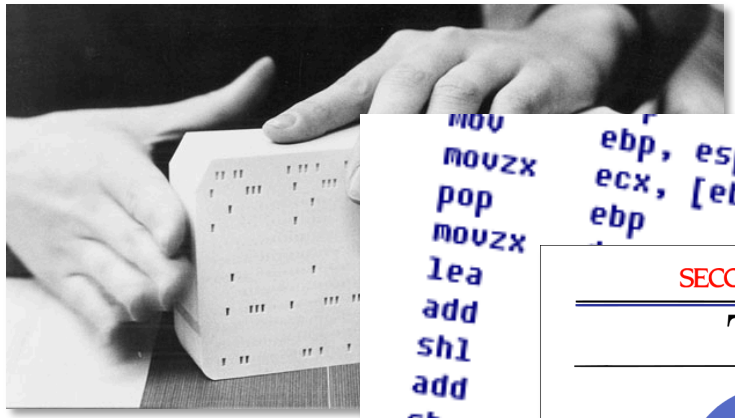
Martin D Brazeau

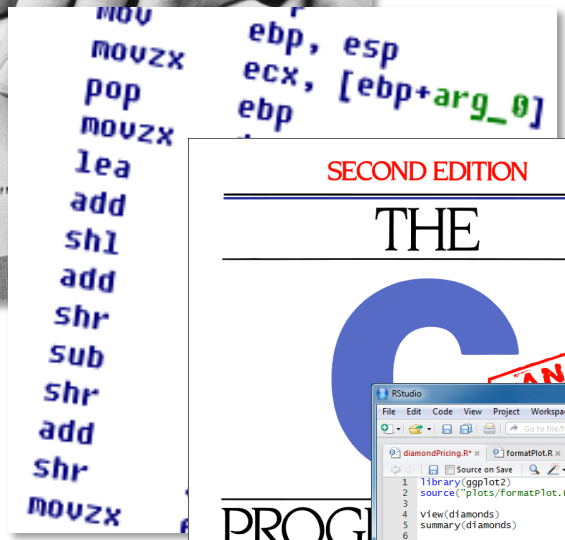
m.brazeau@imperial.ac.uk

Objectives

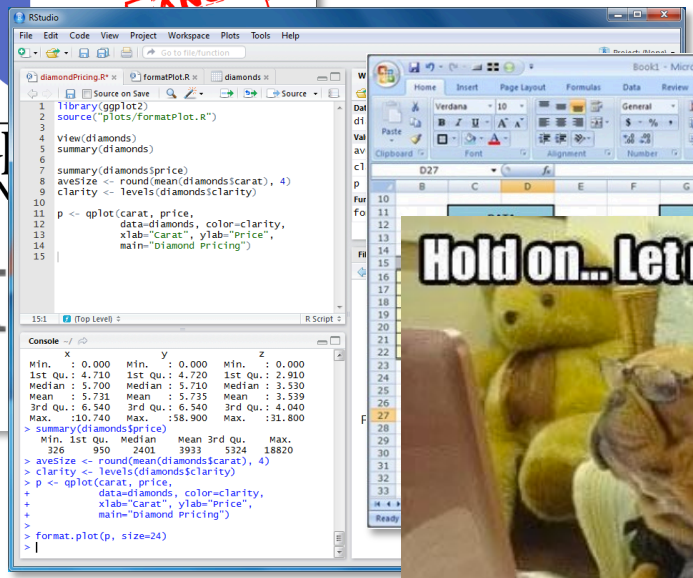
- Learn basics of C (or C-style) programming
- Understand 'primitive' data types
- Learn how low-level programming can speed up complex calculations
- Learn to create (phylogenetic) networks in machine memory
- Learn how to interact with these structures and perform calculations
- Learn how to call C (or C++) from R

Low-level





"for…"
"if-else"



High-level

Why use C?

Practical:

- Maybe you shouldn't
- You should probably use an object-oriented compiled language (e.g. C++, Objective-C, or C#)
- But C is a subset of C++ or Objective-C
- Enough to get you going, and supplies basics of a compiled language

Why use C?

Technical:

- Speeds up calculations
- Create standalone executables
- Access to bit-level
- Popular language, lots of support
- Improve understanding of machine level and optimisation
- Reading/using abundant C code in comp. biol.

What is the connection between C and biological data structures?

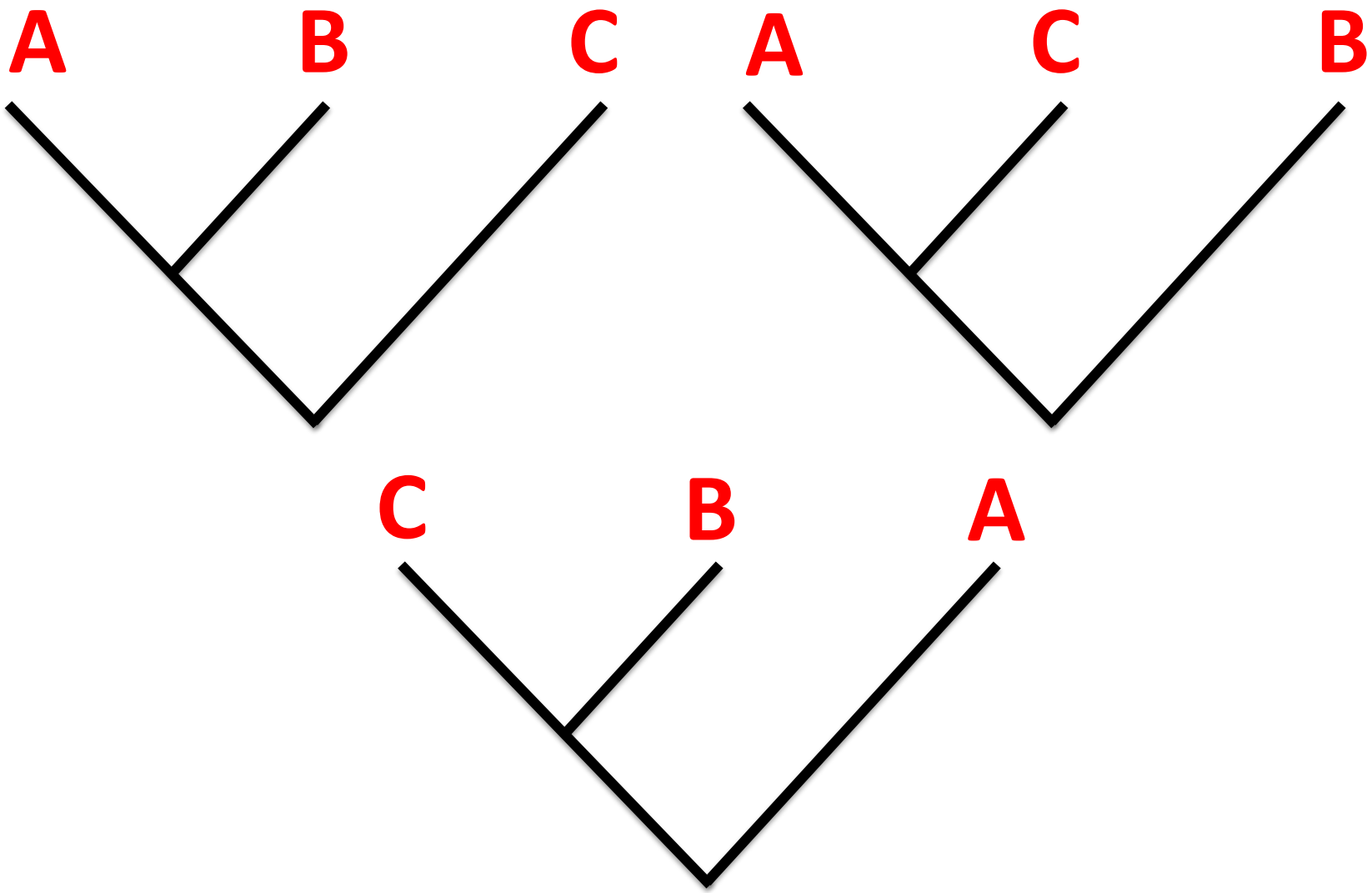
- None inherently
- Data structures often imply computational intensivity
- Likely to need C more with large data structures and computationally hard problems than anywhere else

Part I: How Many Trees Are There?

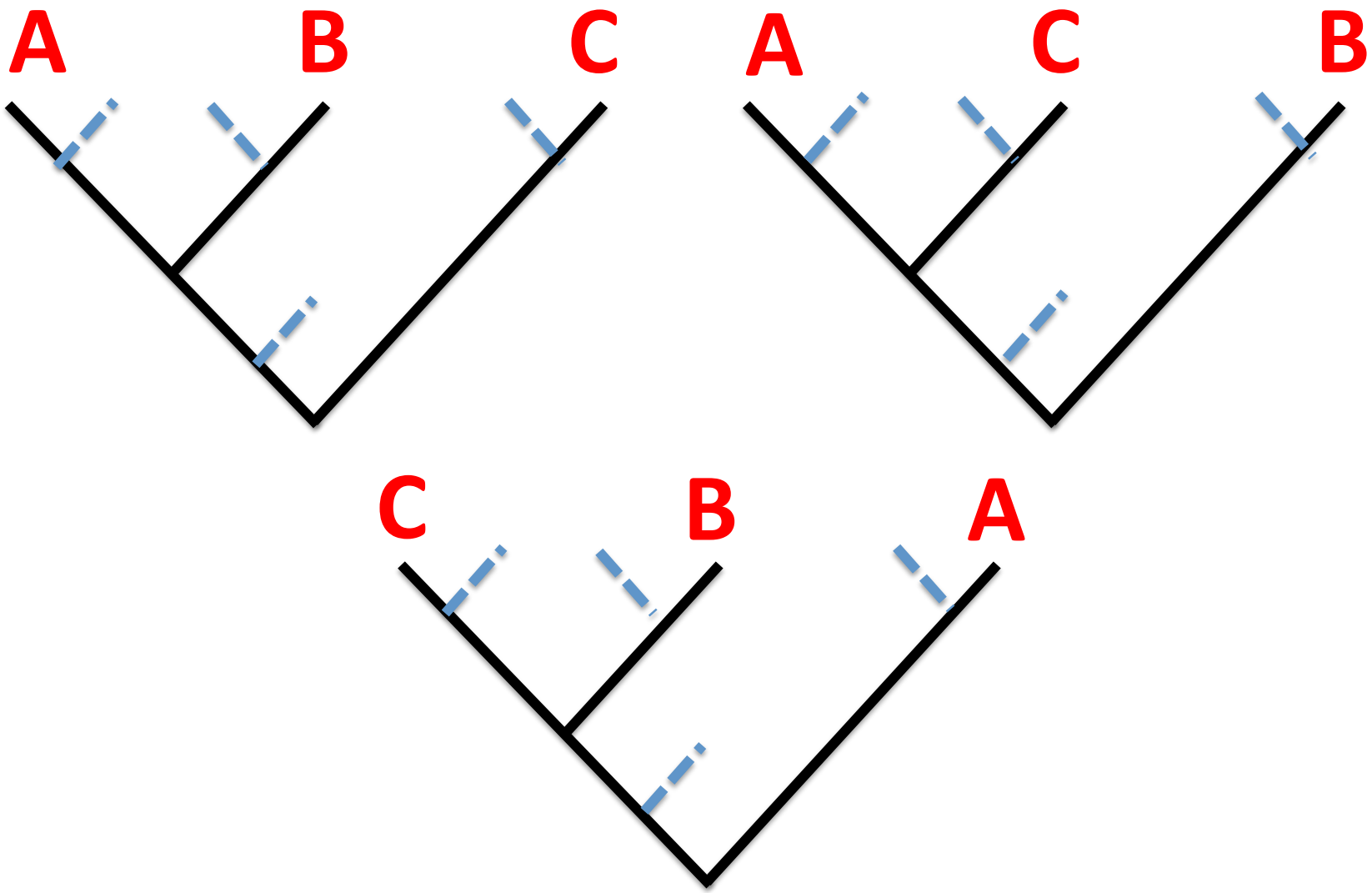
Finding the best phylogeny… a very difficult problem

- No analytical solution
- NP-complete: no solution in polynomial time
- Trial and error

First, generate the trees for 4 species



First, generate the trees for 4 species



How many trees?

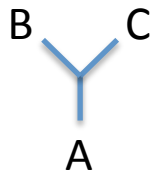
This many:

$$\frac{(2n-3)!}{2^{n-3}(n-3)!}$$

Number of taxa	Number of rooted trees
3	3
4	15
5	105
6	945
7	10395
8	135135
9	2027025
10	34459425
20	8.20×10^{21}
30	4.95×10^{38}
40	1.01×10^{57}
50	2.75×10^{76}
60	5.86×10^{96}
70	6.85×10^{117}

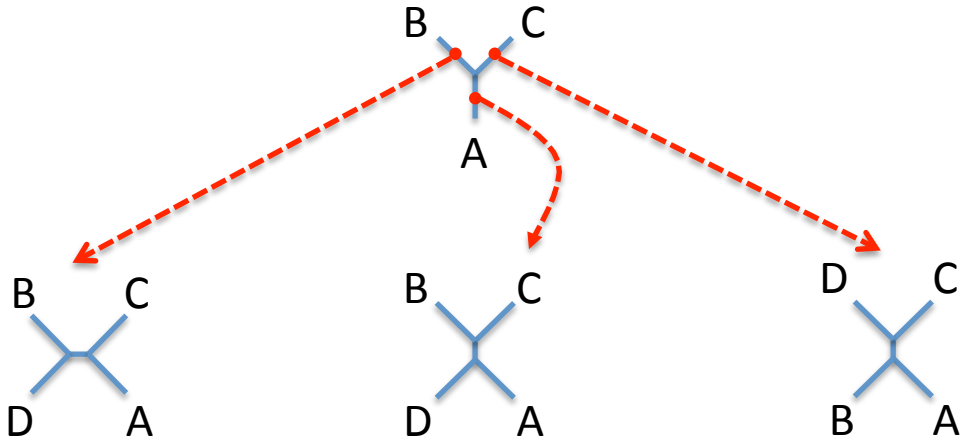
Branch-and-bound searching

- Start with an (arbitrary) three-taxon, unrooted tree



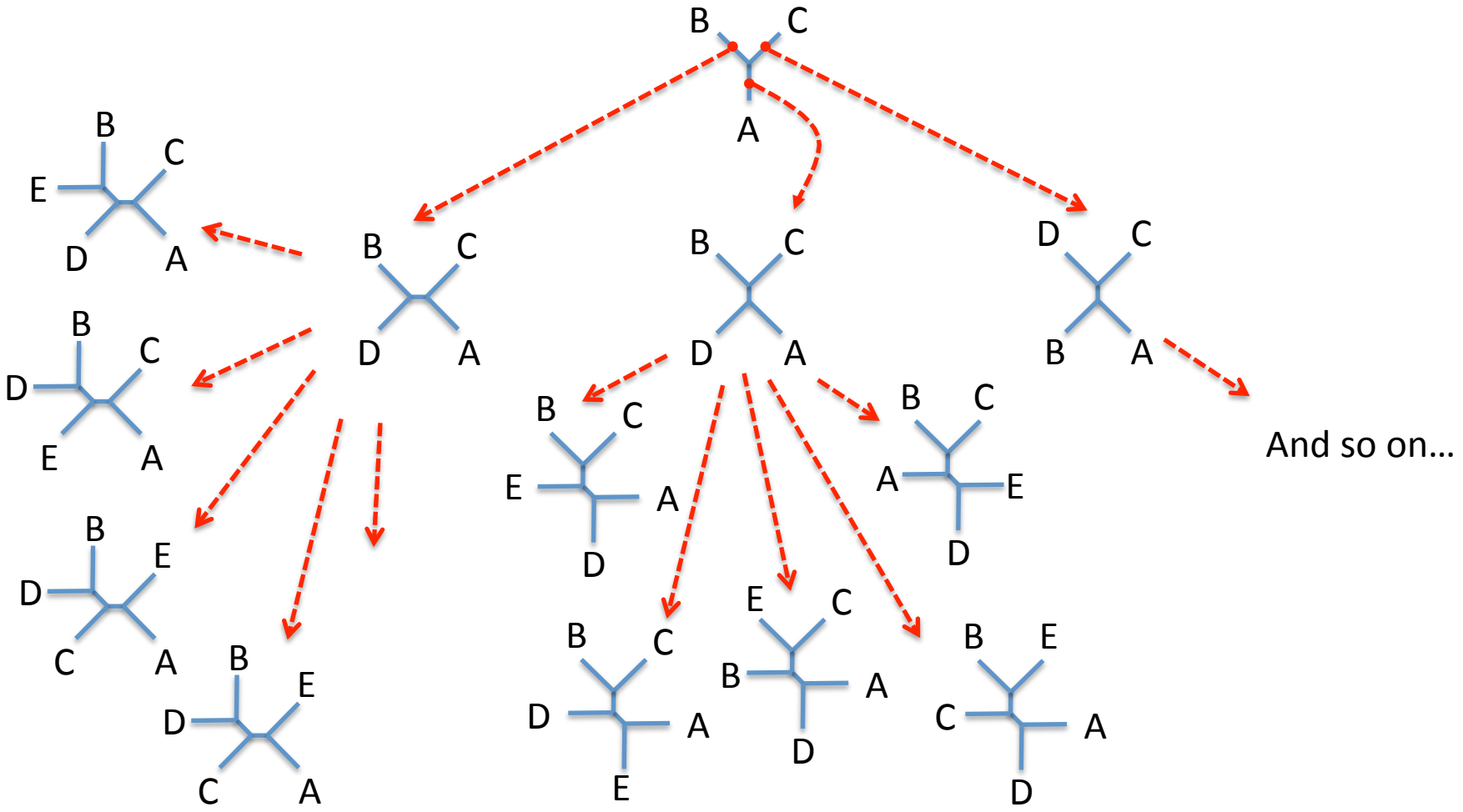
Branch-and-bound searching

- Add fourth taxon in all positions.
- Evaluate the length at each iteration



Branch-and-bound searching

- Add fourth taxon in all positions.
- Evaluate the length at each iteration



Branch-and-bound searching

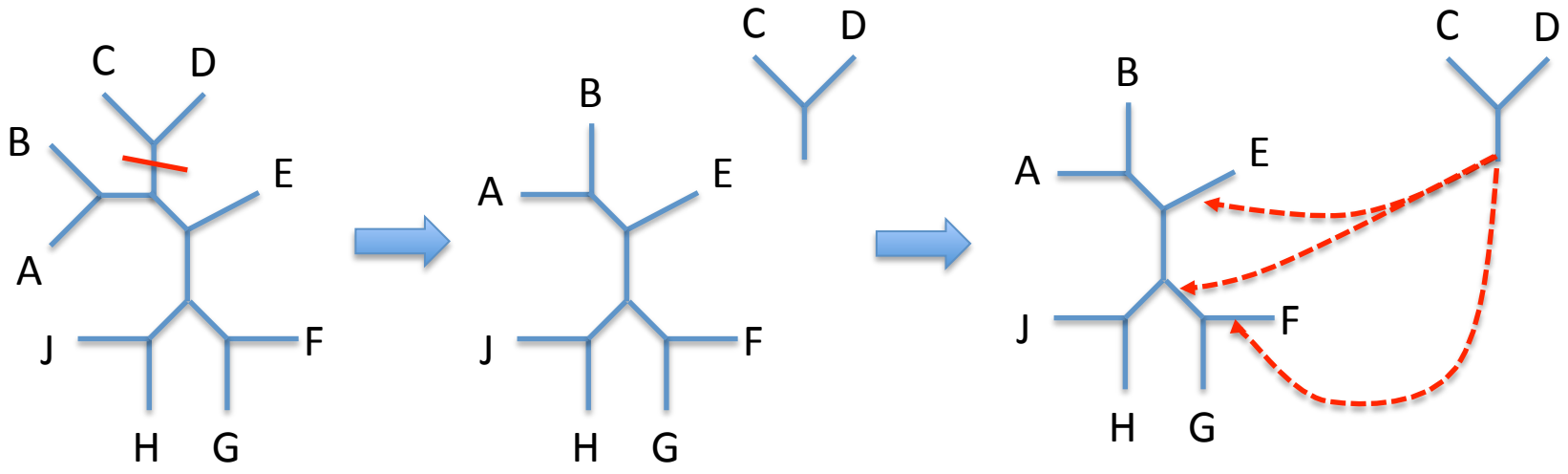
- Merely delays the inevitable:
- Tree numbers still grow exponentially with increasing numbers of terminal taxa

Heuristic methods

- Successive local rearrangements
- Trees are broken and rearranged systematically
- Also hill-climbing algorithms

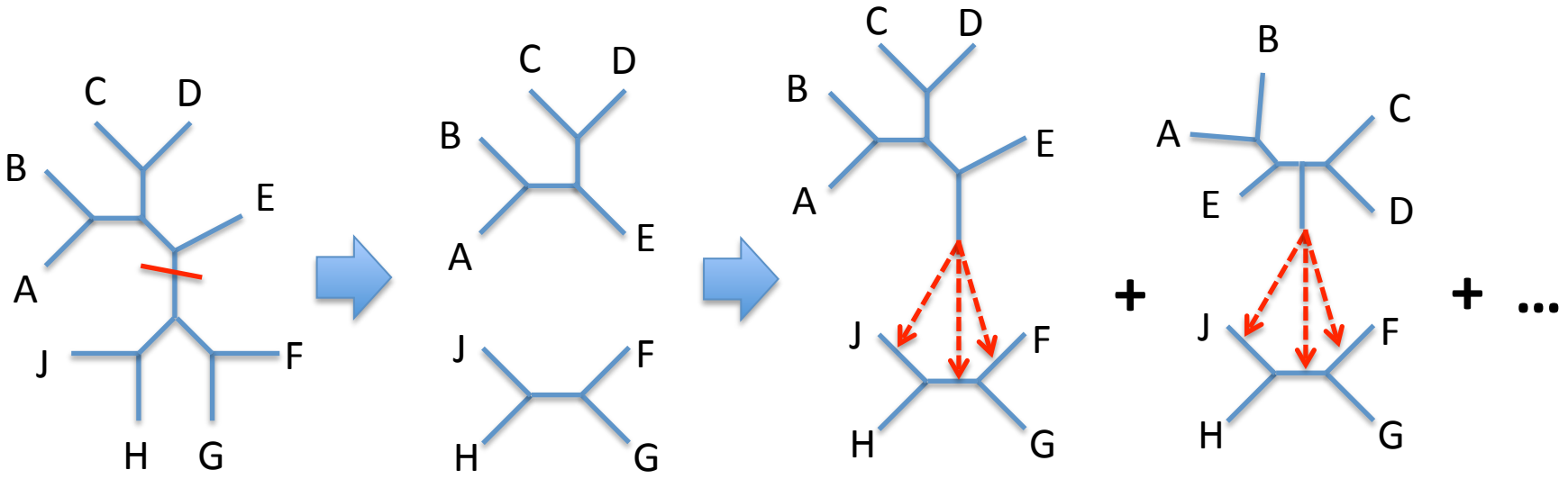
Finding trees by branch rearrangment

- E.g. the subtree prune and regraft (SPR)
- A subtree is cut from the main tree
- Reinserted at all possible positions

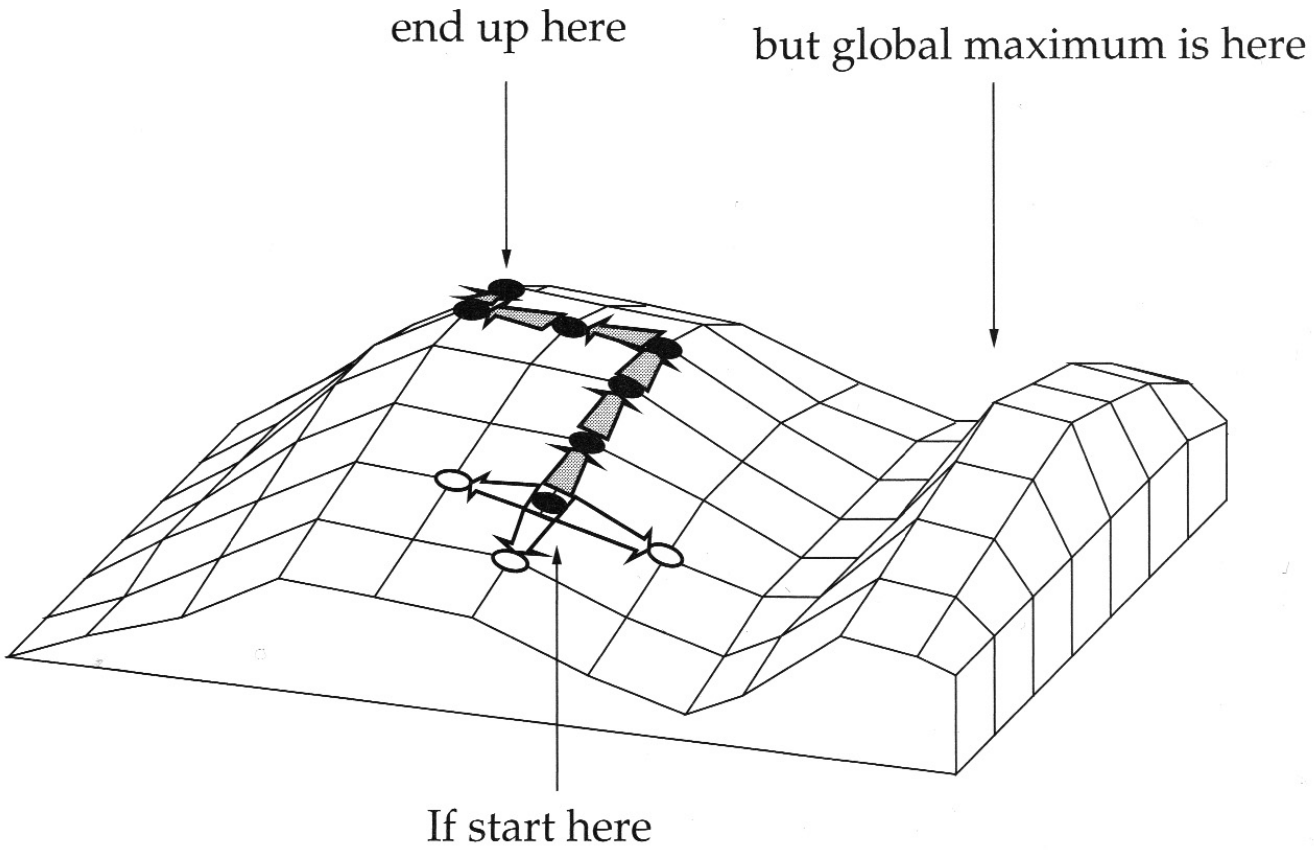


Tree bisection and reconnection (TBR)

- Like SPR, but the pruned subtree is iteratively *rerooted* at each possible reinsertion point.
- *Very* thorough and effective as a search strategy



Greedy hill-climbing algorithms



Felsenstein (2004)

Conclusions

- We can spend a long time solving network optimality problems
- Phylogenies are but one example
- Sensible algorithms are key
- Computational speed is limiting