

# Protein Design

- What is it ?
- Why ?
- Experimental methods
- What we need
- Computational Methods
  
- introduce
  - Monte Carlo
  - a pruning algorithm

# What is protein design ?

## Assumption

- you can write a protein sequence on a piece of paper
- a molecular biologist can produce it

## Most general

- you have a protein which is useful (enzyme, binding, ...)
- you want to make it more stable
  - temperature
  - solvents (tolerate organic solvents)
  - pH
- we concentrate on stability

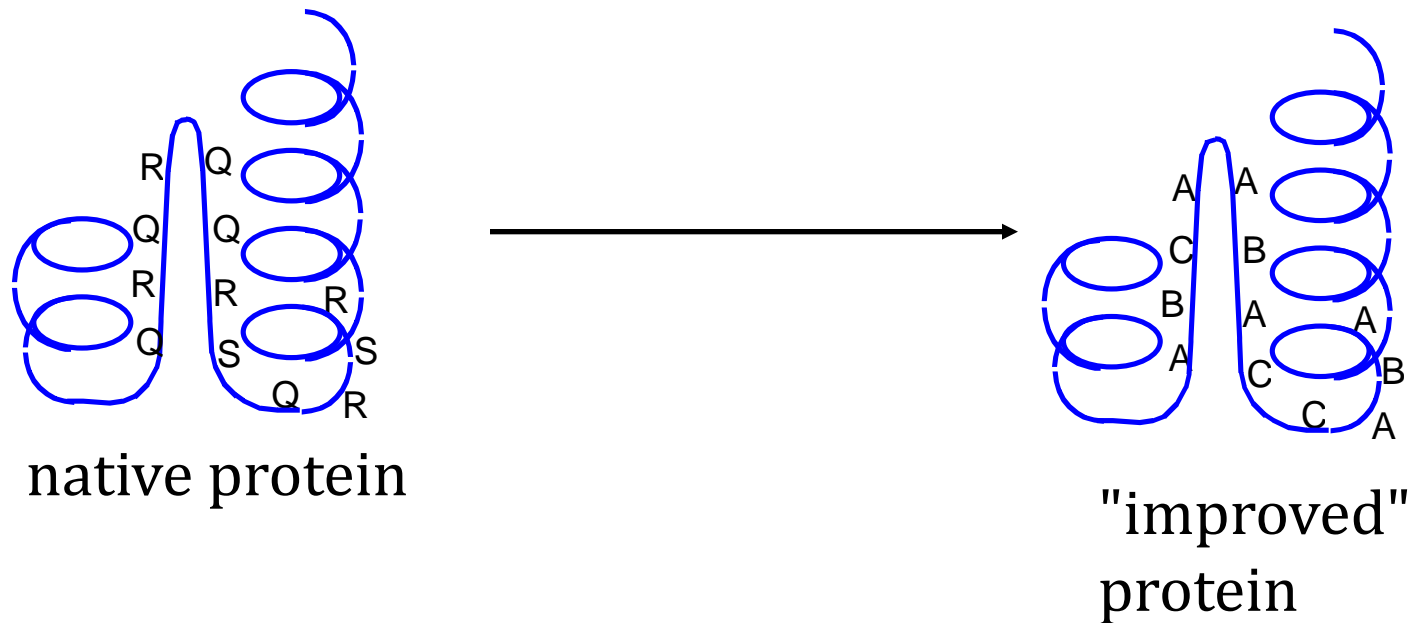
# Experimental approaches

- Bacteria / selection
- For binding
  - phage display
  - in vitro evolution
- stability – more difficult
- computational methods...

# Formalising the problem

We have a working structure

- want to make it more stable



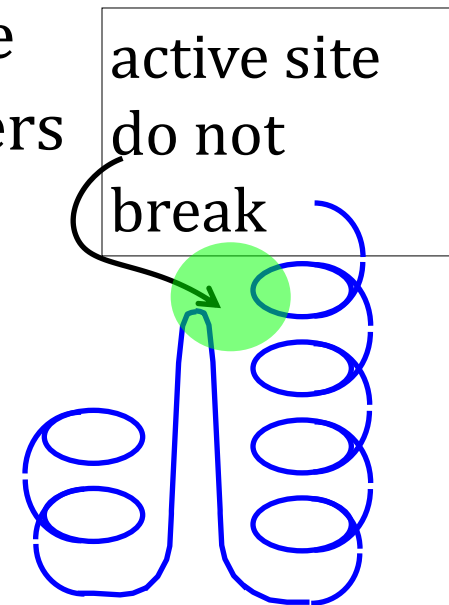
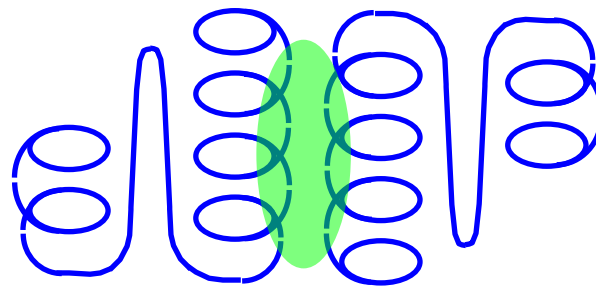
Rules

- structure should not change
- should be able to fix some residues (active site, important)..

# Fixing / specifying residues

## Examples

- lysine (K) often used for binding
  - change a residue to K and protein does not fold
  - mission:
    - adapt the rest of the residues to be stable
- change all residues, but not those in active site
- change some residues at surface to be soluble
- change some residues at surface to stop dimers



# Ingredients

- Score function (like energy)
- Search method

## Score function

- how does sequence fit to structure ?
- sequence  $S = \{s_1, s_2, \dots, s_N\}$
- coordinates  $R = \{ \mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_N \}$
- score =  $f(S, R)$  (different nomenclature soon)
- mission
  - adjust S to as to maximise score (minimise quasi-energy)

# Score function

How do amino acids

- suit structure ?
- suit each other ?

$$\begin{aligned} \text{score} &= \sum_{i=1}^{N_{res}} \text{score}_{struct}(s_i, R) \\ &+ \sum_{i=1}^{N_{res}} \sum_{j>i}^{N_{res}} \text{score}_{pair}(s_i, s_j, R) \end{aligned}$$

$\text{score}_{struct}$  might have

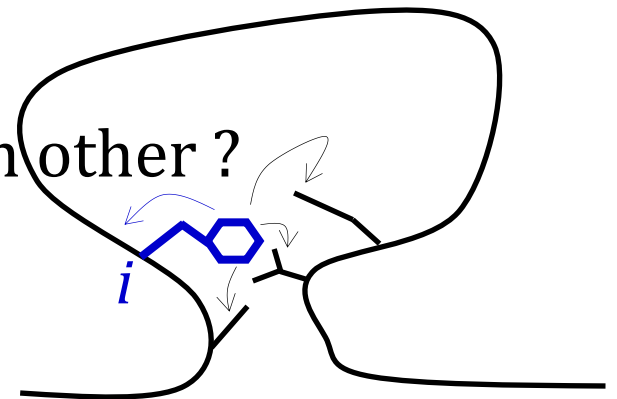
- backbone preferences (no proline in helices, ..)
- solvation (penalise hydrophobic at surface)

$\text{score}_{pair}$

- are residues too big (clashing)
- are there holes ? charges near each other ?

Messy functions

- lots of parameters



# Searching

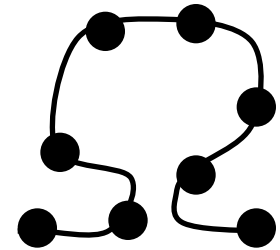
Systematic search – how long ?



- search space for  $N_{res} = 20 \times 20 \times \dots = 20^{N_{res}}$

Search space complex

- every time you change a residue, affects all neighbours
  - effects neighbours of neighbours
- brute force not a good idea
- two methods here
  1. Monte Carlo / simulated annealing
  2. Pruning / dead end elimination





# Monte Carlo

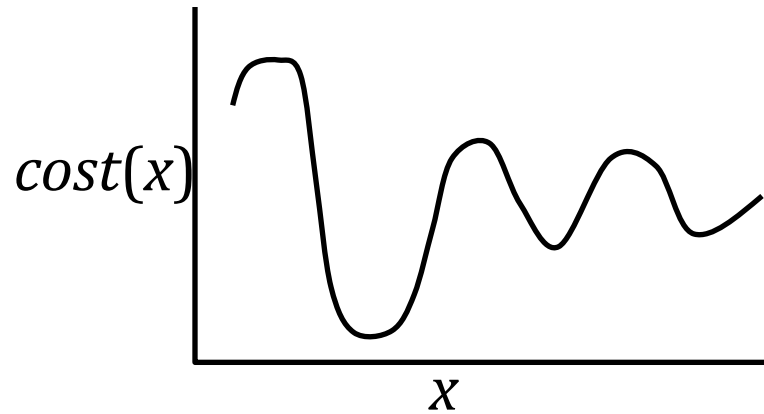
- more formally next semester
- first the problem

## The sequence optimisation problem

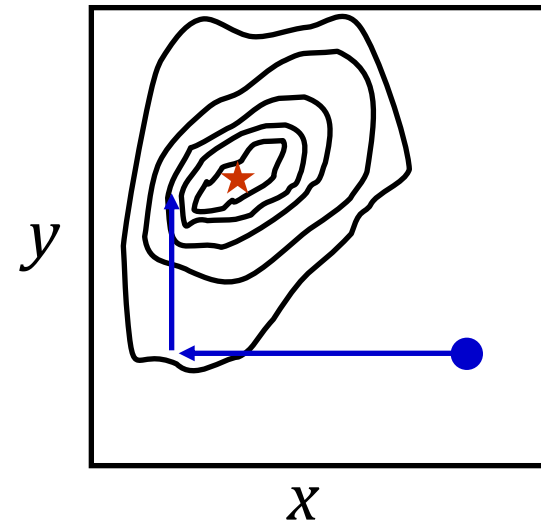
- discrete
- local minima / correlations in surface
- high dimensional

# dimensions and correlations

- a 1D problem



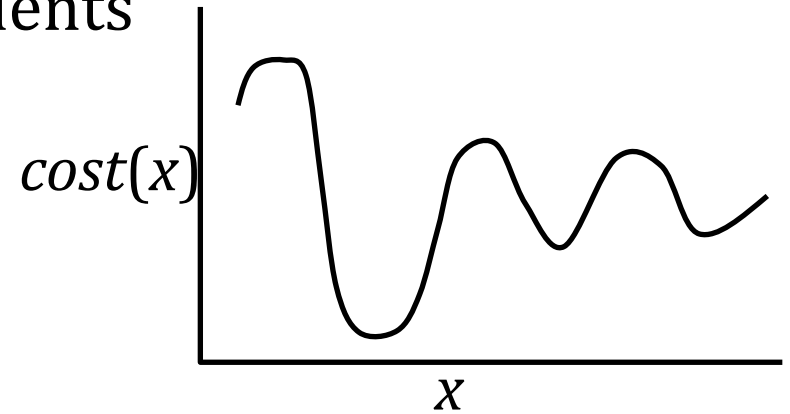
- local minima
- minimum of  $x$  depends on  $y$
- cannot optimize  $x$  and  $y$  independently
- what are correlations in this problem ?



# Discrete vs continuous problems

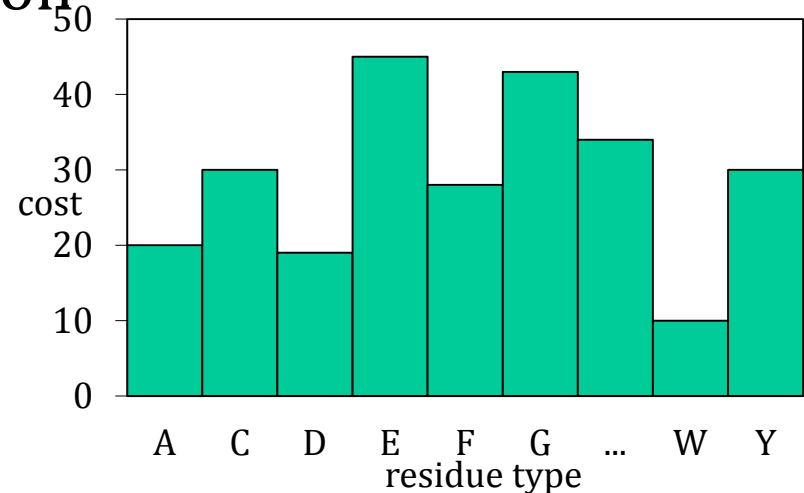
For a continuous function use gradients

- to optimise
- to recognise minima / maxima
- continuous functions
  - step in one direction is good
  - try another in same direction



With a discrete function

- no gradients
- order of labels arbitrary
  - ACDE or ECAD
- discrete
  - step in one direction may be no predictor of best direction



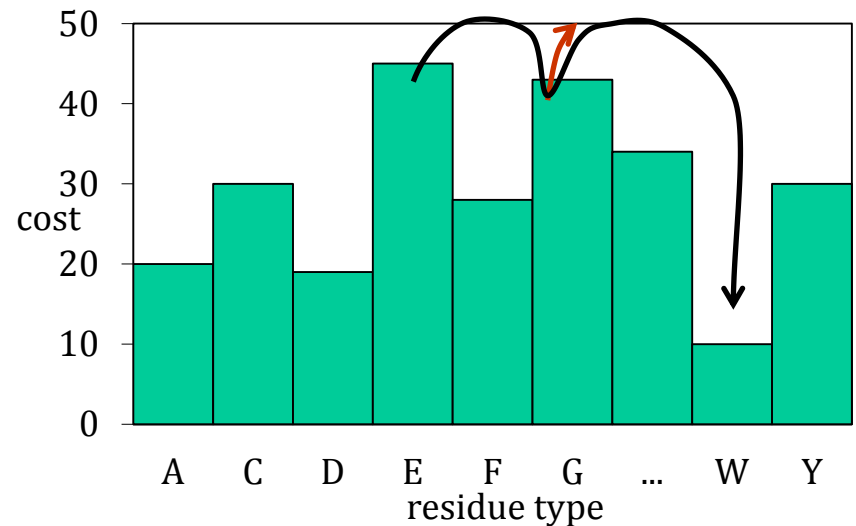
# what do we want ?

From step to step (sequence to sequence)

- be prepared to move in any direction
- if the system improves, try not to throw away good properties
- must be willing to go uphill sometimes

## Philosophy

- take a random move
- if it improves system
  - keep it
- if cost becomes worse
  - sometimes keep it
  - sometime reject



# Acceptance /rejection

- for convenience, write  $cost(S_n)$  - neglect coordinates R

Sign convention

- system (sequence) at step  $n$  is  $S_n$
- after a random step, cost changes from  $cost(S_n)$  to  $cost(S_{n+1})$
- $\Delta c = cost(S_{n+1}) - cost(S_n)$
- our sign convention: if  $\Delta c < 0$ , system is better

When to accept ?

- if  $\Delta c$  is a bit  $< 0$  accept
- if  $\Delta c$  is a bit  $> 0$ , maybe OK
- if  $\Delta c \gg 0$ , do not accept

# Formal acceptance rule

- $-\Delta c < 0$ ,  $e^{-\Delta c}$  is between 0..1
- $-\Delta c \approx 0$  then  $e^{-\Delta c} \approx 1$  as  $\Delta c \rightarrow \infty$  then  $e^{-\Delta c} \rightarrow 0$
- formalise this rule

```
set up  $S=S_0$  and  $cost(S_0)$ 
```

```
while (not finished)
```

```
     $S_{\text{trial}}$  = random step from  $S$ 
```

```
     $\Delta c$  =  $cost(S_{\text{trial}}) - cost(S)$ 
```

```
    if ( $\Delta c < 0$ )
```

```
        /* accept */
```

```
         $S = S_{\text{trial}}$ 
```

```
    else
```

```
         $r = \text{rand}(0..1)$ 
```

```
        if ( $e^{-\Delta c} \geq r$ )
```

```
             $S = S_{\text{trial}}$ 
```

- vorsicht ! not the final method

# why we need temperature

As described

- system will run around
- try lots of new configurations
- sometimes accept bad moves
- always take good moves
- may never find best solution
  - imagine you are at a favourable state
  - most changes are uphill (unfavourable)
  - many of the smaller ones will be accepted
    - if we were to find the best sequence, the system would move away from it
- how to fix ?

# why we need temperature

- Initial sequence is not so good
  - let the system change a lot and explore new possibilities
- after some searching, make the system less likely to go uphill
- introduce the concept of temperature  $T$
- initially high  $T$  means you can go uphill (like a high energy state)
- as you cool the system down, it tends to find lowest energy state

- change acceptance criterion to  $e^{\frac{-\Delta c}{T}}$  as

$$T \rightarrow \infty, \quad e^{\frac{-\Delta c}{T}} \rightarrow 1$$

$$T \rightarrow 0, \quad e^{\frac{-\Delta c}{T}} \rightarrow 0$$

- put this into previous description



# why we need temperature

```
set up  $S = S_0$  and  $cost(S_0)$  set  $T = T_0$ 
```

```
while (not finished)
```

```
     $S_{\text{trial}}$  = random step from  $S$ 
```

```
     $T = \epsilon T$ 
```

/\*  $\epsilon$  bit smaller than 1 \*/

```
     $\Delta c = cost(S_{\text{trial}}) - cost(S)$ 
```

```
    if ( $\Delta c < 0$ )
```

```
         $S = S_{\text{trial}}$ 
```

```
    else
```

```
         $r = \text{rand}(0..1)$ 
```

```
        if ( $\exp(-\Delta c/T) \geq r$ )
```

```
             $S = S_{\text{trial}}$ 
```

Name of this procedure

- "simulated annealing"

# Final Monte Carlo / annealing

## History applications

- discrete problems – travelling salesman, circuit layout
- deterministic ? No
- convergence ? Unknown

## Practical issues

- what is a random step ?
  - change one amino acid ? change interacting pairs ?
- easy to program
- lots of trial and error
- statistical properties next semester
  
- can we reduce the search space ?

# Pruning

Are there elements of sequence which are impossible ?

- at position 35, no chance of Y, W, I, L, ...

Can one find impossible combinations

- reduce the search space so it can be searched systematically  
(brute force)

... dead end elimination method

- use an energy-like nomenclature

# Nomenclature

- we are not dealing with
  - free energy  $G$  or  $F$  or potential energy  $U$  or  $E$
- but let us pretend
  - score is  $E$
- rule : more negative  $E$  , better the system
- structure is fixed so neglect  $R / \mathbf{r}$  terms
- define a function  $s_i(a)$  as the residue type at site  $i$ 
  - can take on 20 values of "a" why ?  
**foreach (a in A, C, D, E..., W, Y)**  
**evaluate energy corresponding to a**
- our energies ?
  - two parts – pairwise and residue with backbone

# Nomenclature

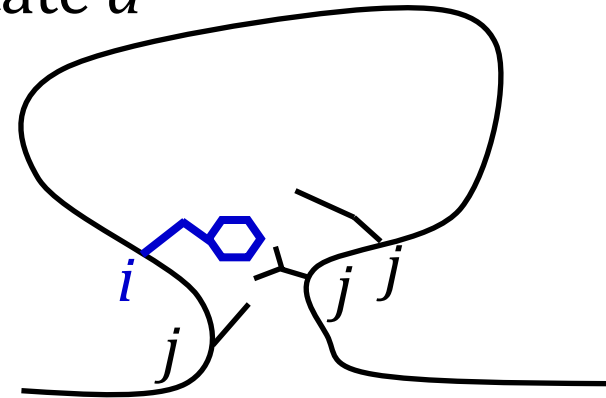
$E$  is (quasi-energy) of whole system

- label  $E_1$  as the terms that depend on residue + fixed environment
- $E_2$  as the energy terms that depend on pairs

$$E = \sum_{i=1}^{N_{res}} E_1(s_i) + \sum_{i=1}^{N_{res}} \sum_{j \neq i}^{N_{res}} E_2(s_i, s_j)$$

If we are interested in site  $i$  and being in state  $a$   
what do we have to look at ?

$$\sum_{i=1}^{N_{res}} E_1(s_i(a)) + \sum_{i=1}^{N_{res}} \sum_{j \neq i}^{N_{res}} E_2(s_i(a), s_j(b))$$



# Nomenclature and rules

- there are 20 ( $N_{type}$ ) residues
- which fits best to the fixed environment?  $\min_a E_1(s_i(a))$
- implies testing each of the  $N_{type}$  for  $a$
- what is the best energy type  $a$  at site  $i$  could have, interacting with one site  $j$ ?

$$E_1(s_i(a)) + \min_b E_2(s_i(a), s_j(b))$$

- what is the best energy that type  $a$  at  $i$  could have considering all neighbours?

$$E_1(s_i(a)) + \sum_{j \neq i} \min_b E_2(s_i(a), s_j(b))$$

- for each  $a$  – can work out what is the best score it could yield
  - loop over  $b$
  - within loop over  $j$

# Dead-end elimination method

- worst energy that type  $c$  at  $i$  could have considering all neighbours ?

$$E_1(s_i(c)) + \sum_{j \neq i} \max_d E_2(s_i(c), s_j(d))$$

- when can one eliminate (rule out) residue type  $a$  at site  $i$  ?
- for any residues  $a, c$
- if the best energy for  $a$  is worse than the worst for  $c$ 
  - $a$  cannot be part of the optimal solution ... if

$$E_1(s_i(a)) + \sum_{j \neq i} \min_b E_2(s_i(a), s_j(b)) > E_1(s_i(c)) + \sum_{j \neq i} \max_d E_2(s_i(c), s_j(d))$$

# Dead-end elimination method

$$E_1(s_i(a)) + \sum_{j \neq i} \min_b E_2(s_i(a), s_j(b)) > E_1(s_i(c)) + \sum_{j \neq i} \max_d E_2(s_i(c), s_j(d))$$

using this approach

```
for (i = 0; i < N_res ; i++)
  foreach a in N_type
    calculate worst score for a
    calculate best score for a
for (i = 0; i < N_res ; i++)
  foreach a in N_type
    foreach b in N_type
      if best(a) > worst (b)
        remove a from candidates
```

How strong is this condition ?



# DEE condition

- much of the time
  - cannot really rule out type  $a$
- example ?
  - initial
    - $2 \times 10^{27}$
  - final
    - searchable in 90 cpu hr
- deterministic

Dahiyat, B.I, Mayo, S.L. (1997), Science 278, 82-87

## Combining ideas

- use DEE to get a list of candidate residues at each position
- search remaining space with Monte Carlo / simulated annealing
- not deterministic

# Success

## Method

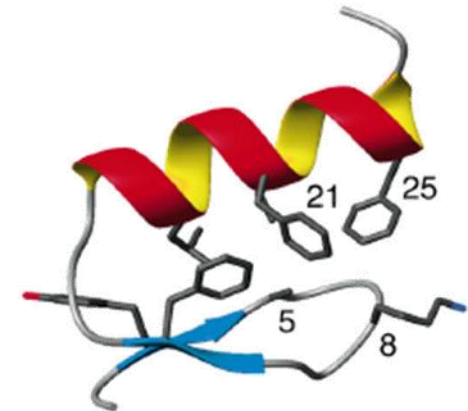
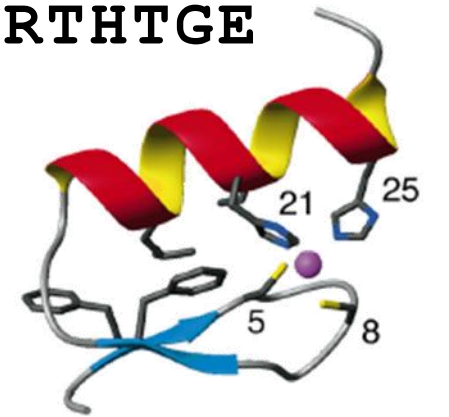
- Dead end elimination + systematic search

designed **QQYTAKIKGRTFRNEKELRDFIEKFKGR**

native **KPFQCRICMRNFSRSDHLTTHIRTHTGE**

## New sequence

- about 20 % similar to start
- not related to any known protein (still)
- Structure solved by NMR
- Problem solved ?
  - maybe not





# Success

## Methods

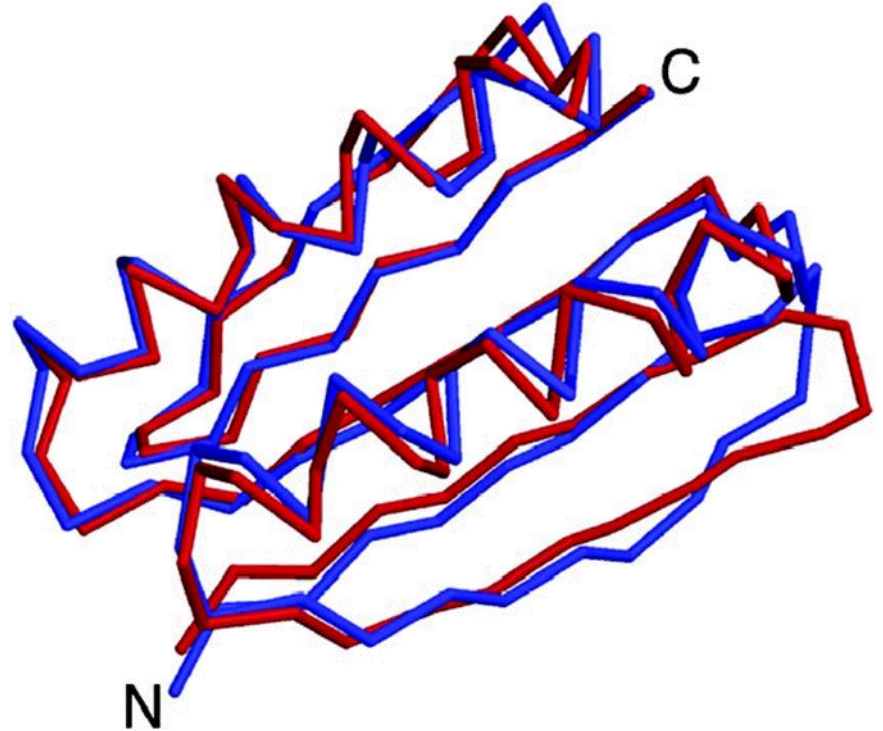
- pure Monte Carlo

## Result

- apparently new sequence

## Structure

- as predicted
- solved by X-ray
  - phasing story
- Problem solved
  - unclear (how many failures ?)



# Methods so far

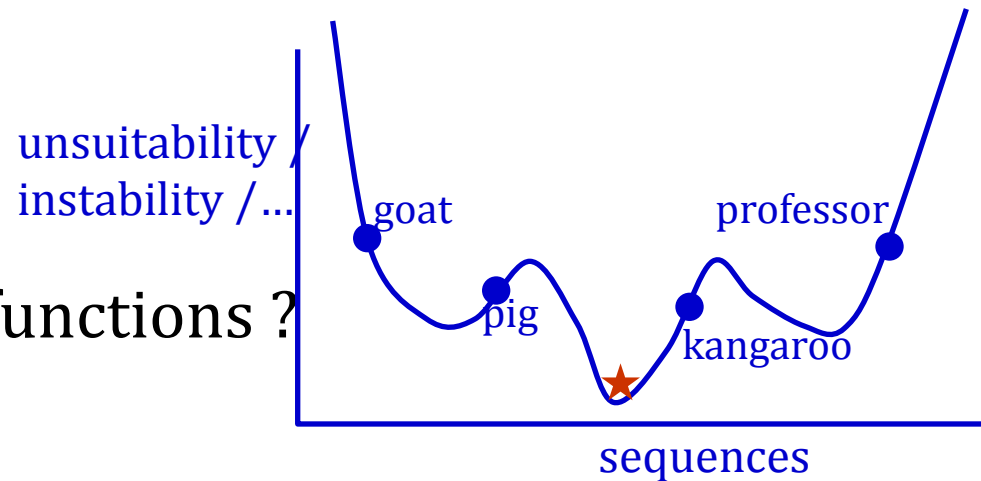
	Monte Carlo	Dead-end elimination
guaranteed global optimum	no	does not try
deterministic	no	yes

# Only one answer ?

May not matter

- consider real proteins – compare human, goat, ...
  - all stable – all slightly different
- implication
  - there may be many solutions which are equally good

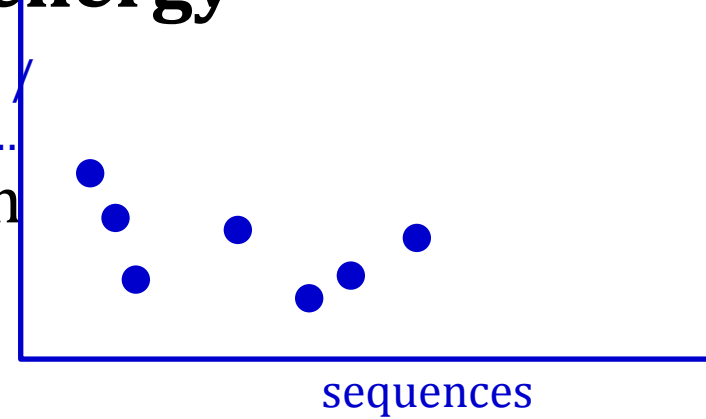
- How good are our energy functions ?



# Determinism and energy

I have a perfect score / energy function

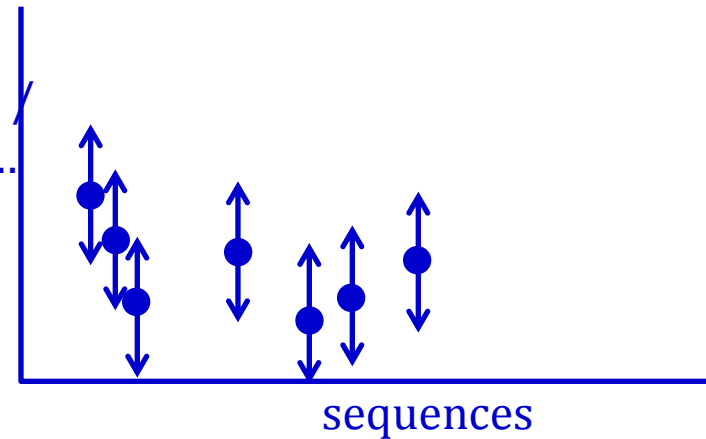
unsuitability /  
instability / ...



I have errors / approximations

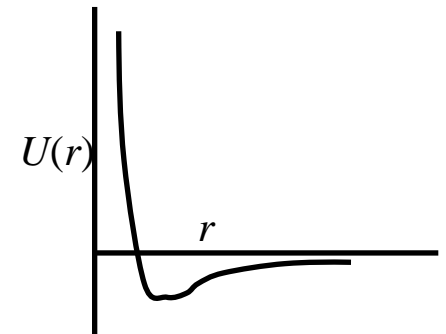
- best answer could be any one

unsuitability /  
instability / ...





# Problems - stability / energy

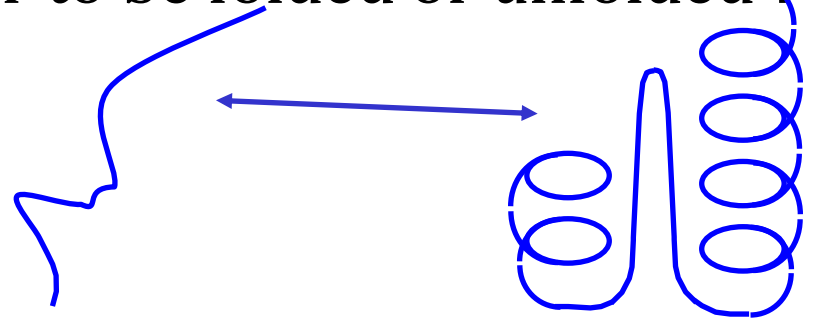
- energy functions
- what do we mean by energy ?
- example - two charges  $U(r) = \frac{q_1 q_2}{Dr}$
- example - two argon atoms  $U(r) = 4\varepsilon(\sigma^{12}r^{-12} - \sigma^6r^{-6})$
- make energy better ?
  - replace every amino acid by a larger one (more contacts - more negative energy)
  - silly - proteins are not full of large amino acids
- what determines stability ?







# Problems - stability / energy

- stability – does a molecule prefer to be folded or unfolded?
- what is unfolded?  or  ?



My energy function tells me to change "X" to "Y"

- it affects both the good  and bad 
- has it affected the energy difference?
  - no guarantee

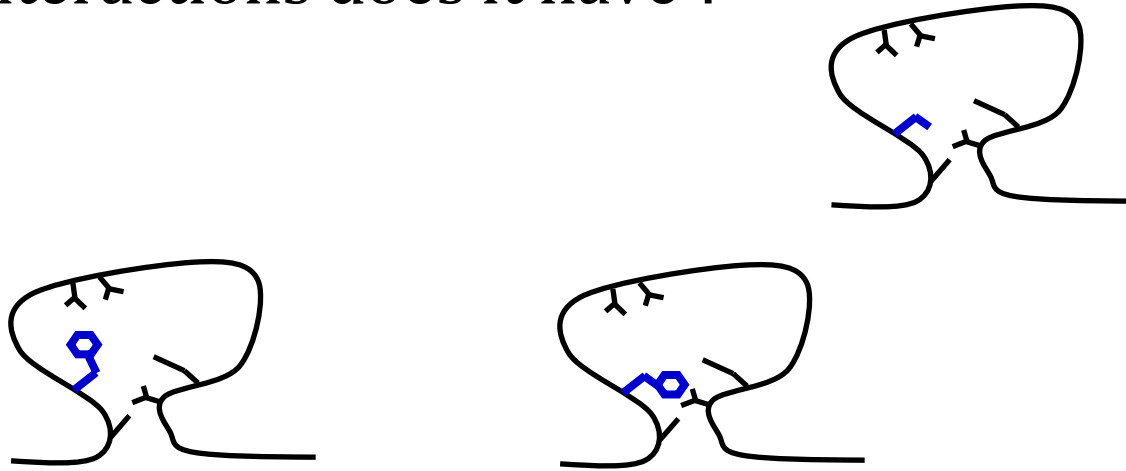
Current score functions ?

- some pure potential energy
- very difficult to estimate  $\Delta G$

# Problems - sidechains

## Side chain positions

- can I ever calculate the energy if I change X to Y ?
- insert a phe into this structure
- what interactions does it have ?



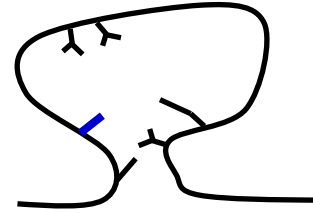
## How to cope with side chain positions in a practical way

- optimise location of sidechains
- use average
- explicit rotamers

# Sidechains - optimise at each step

- I start with known protein

- change A  $\rightarrow$  F



- use an energy minimiser / optimiser to find best position for F

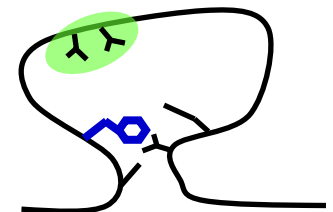
- sensible ?

- we have a gigantic search space
- explicit optimisation of one side chain would be expensive

- silly ?

- I change A $\rightarrow$ F, but the rest of the side chains may move

- bad idea



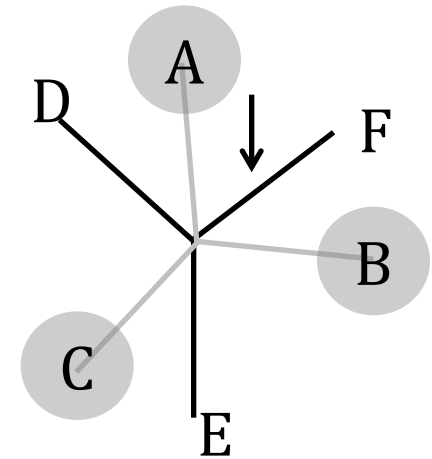
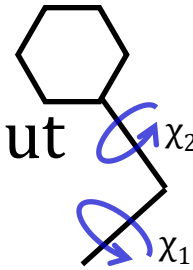
# Sidechains - use averaging

Ignore the problem of sidechain geometry

- at room temperature, side chains move
  - small (middle of protein) to big (surface)
- we cannot expect Å accuracy anyway
  
- rather fast
  
- what if we want to worry about atoms ?

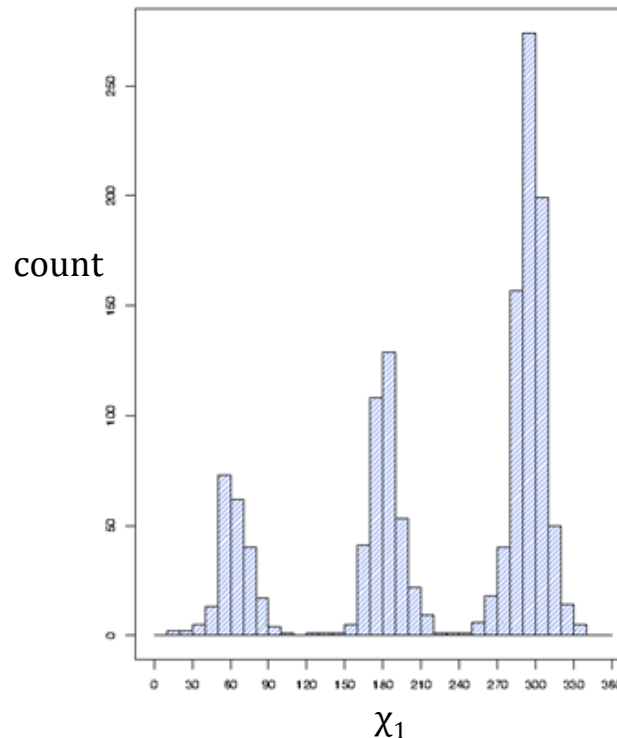
# Sidechains - use rotamers

- sidechains can move anywhere but
  - there are preferences in diagram - three more likely states



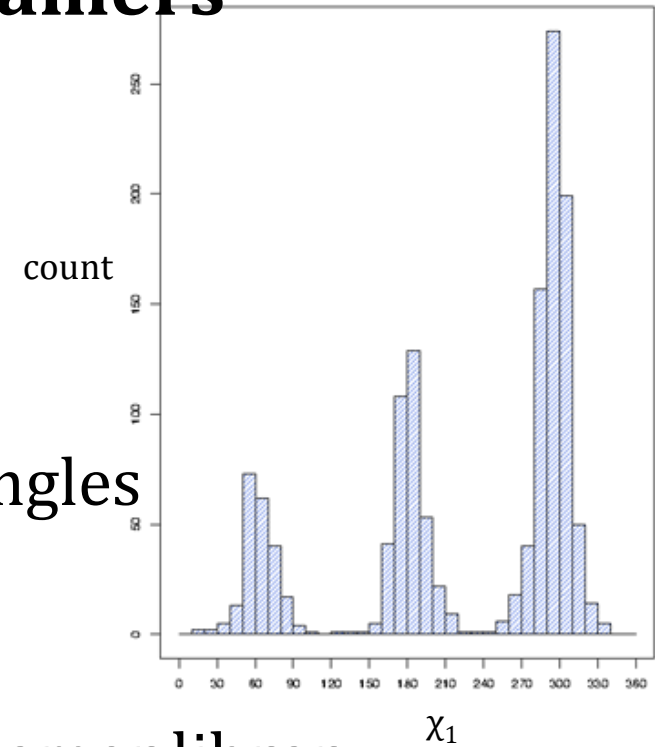
- how many times is the first angle ( $\chi_1$ ) seen at each angle ?

- how to use this ?
  - look for most popular angles (60, 180, 300)



# Sidechains - use rotamers

- For this example
  - do not have 1 cys residue
  - replace with cys1, cys2, cys3
  - treat all amino acids similarly
- more complicated because of more angles
- consequence
  - $N_{type}$  of amino acids  $\gg 20$
- requires that you have a pre-built rotamer library
- fits to
  - Monte Carlo (random moves between residues or rotamers)
  - dead end elimination (will remove impossible rotamers)



# Problems - viability

Designed sequences must

- fold
- be expressed + produced

# Summary

- Nature of the problem - discrete (not continuous)
- Optimisation methods (MC, DEE)
- Score functions
  - not energy, not free energy, not potential energy
- Success / state of the art
  - not many examples from literature
  - failure rate ?
  - cost
- Definitely not a routine method