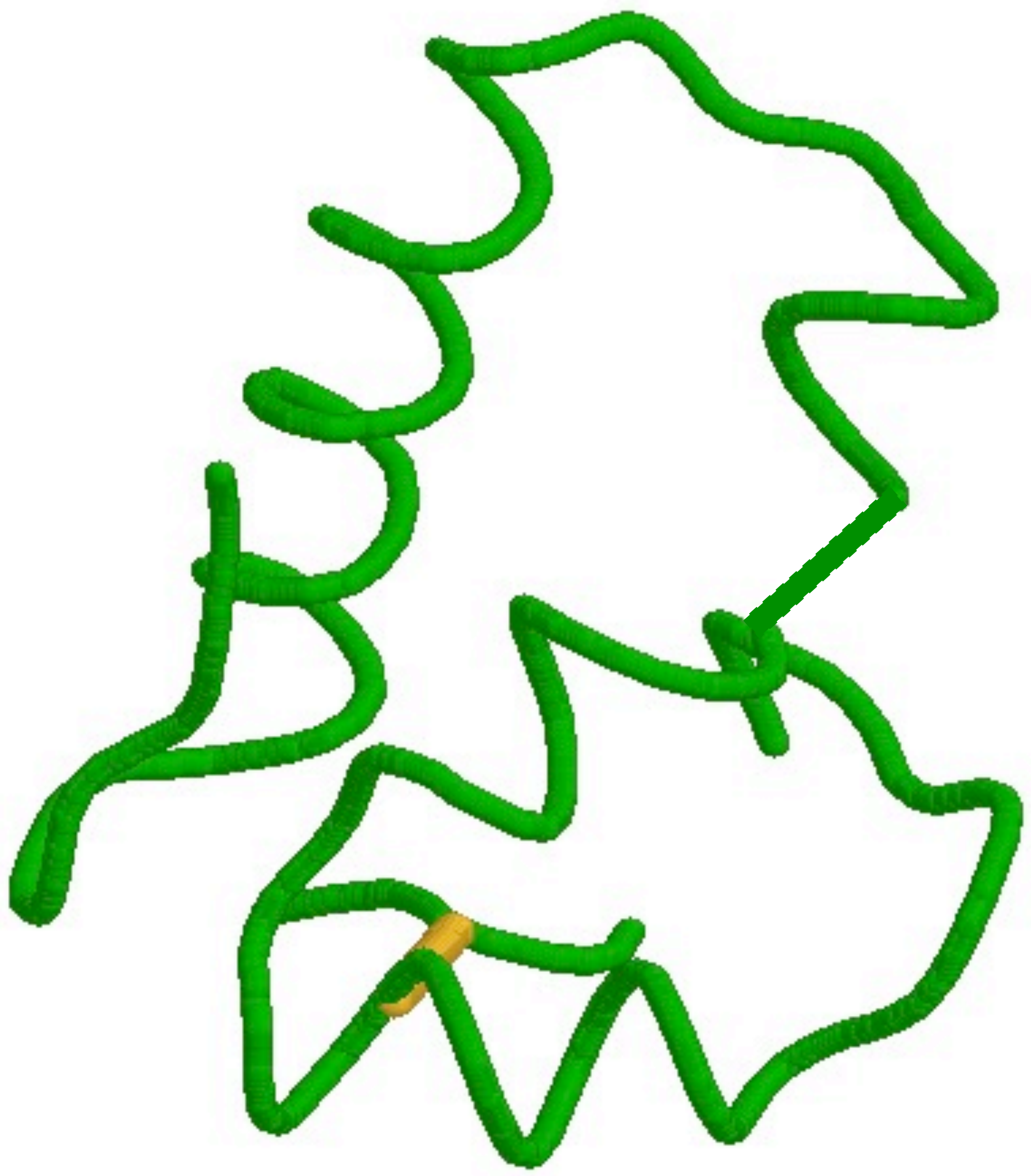
Side-Chain Positioning

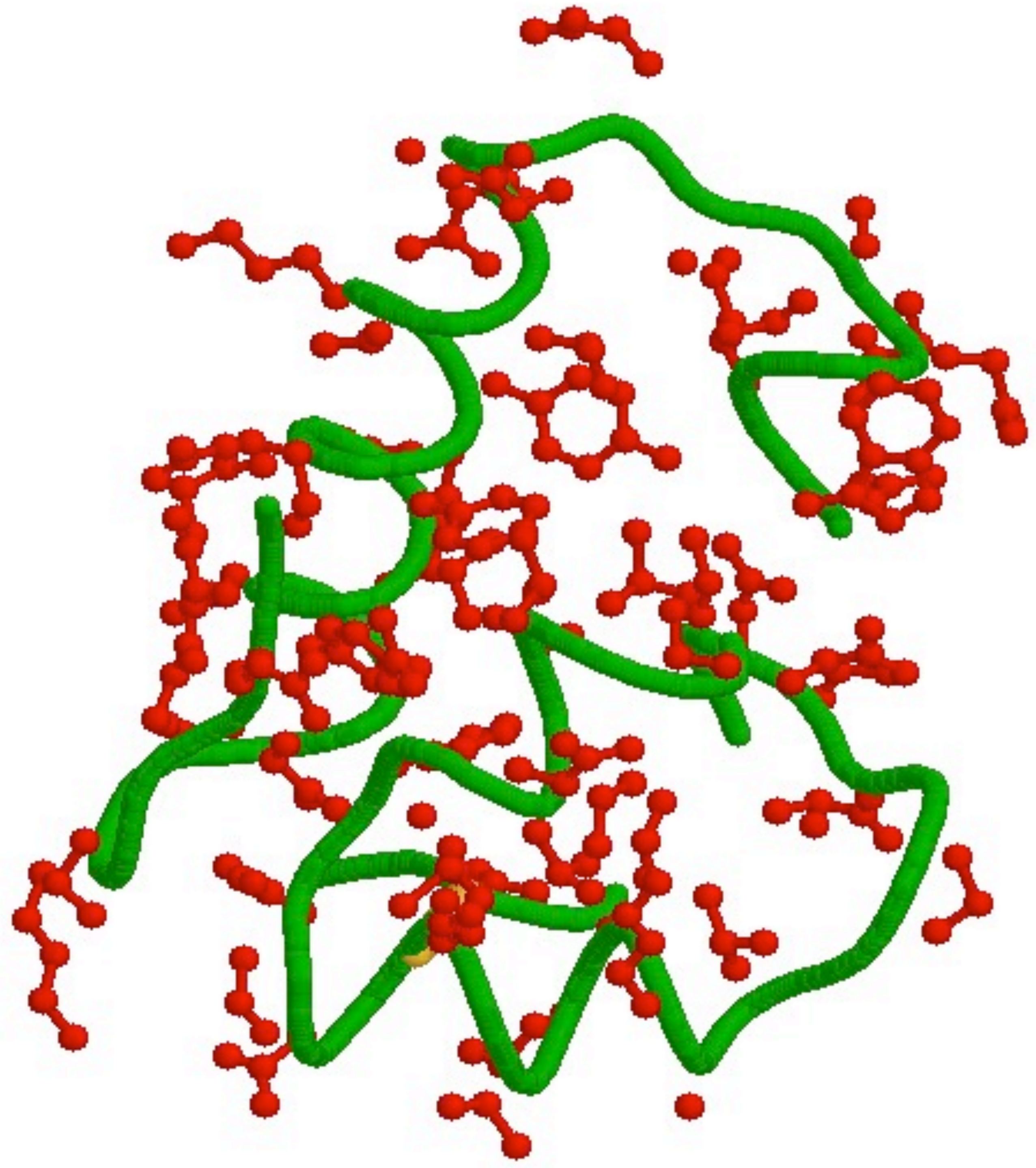
CMSC 423

Protein Structure



Backbone

Protein Structure



Backbone

Side-chains

Side-chain Positioning

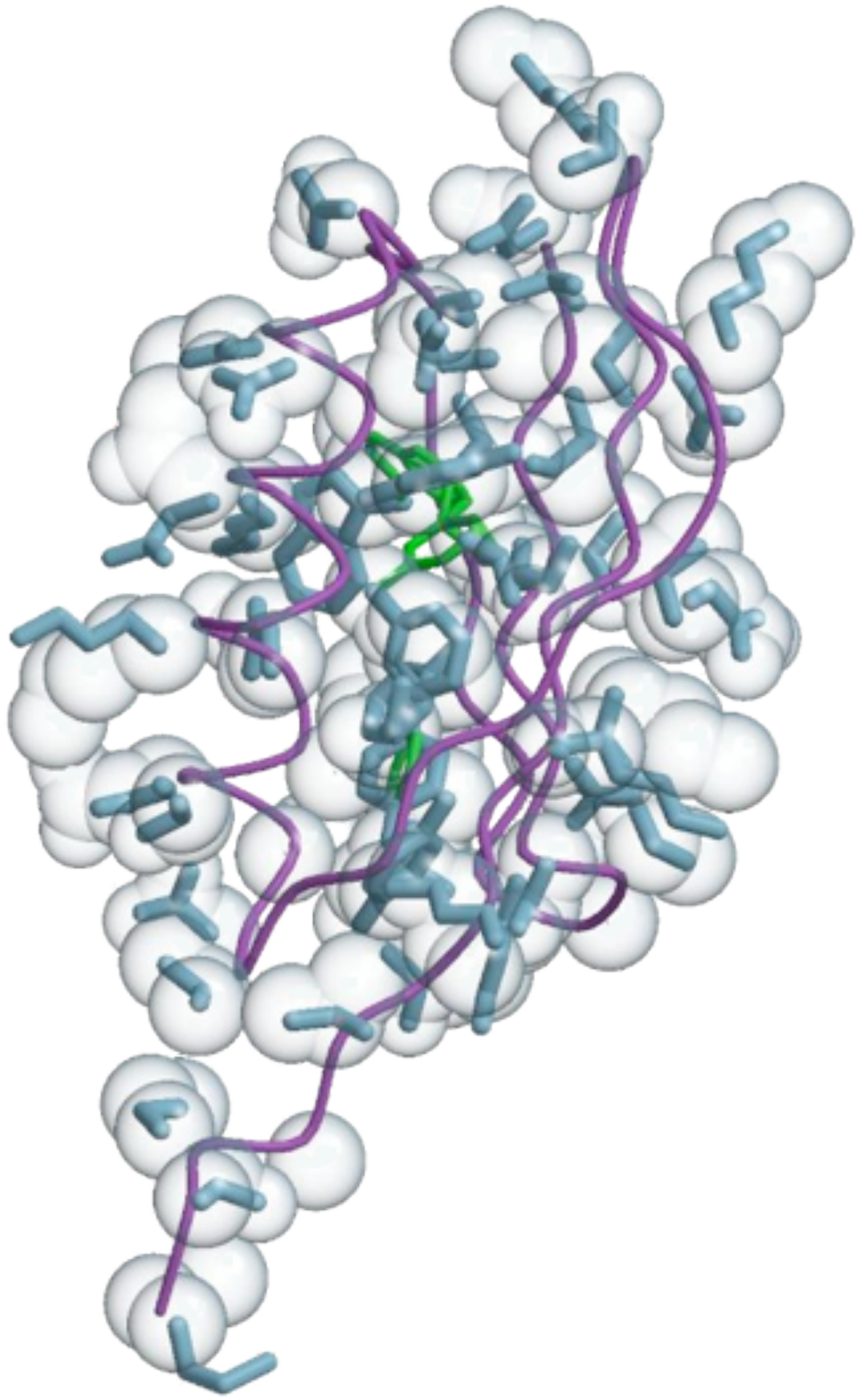
Given:

- amino acid sequence
- position of backbone in space

Find best 3D positions for side chains

"Best" = lowest-energy

Discrete formulation reasonable using *rotamers*

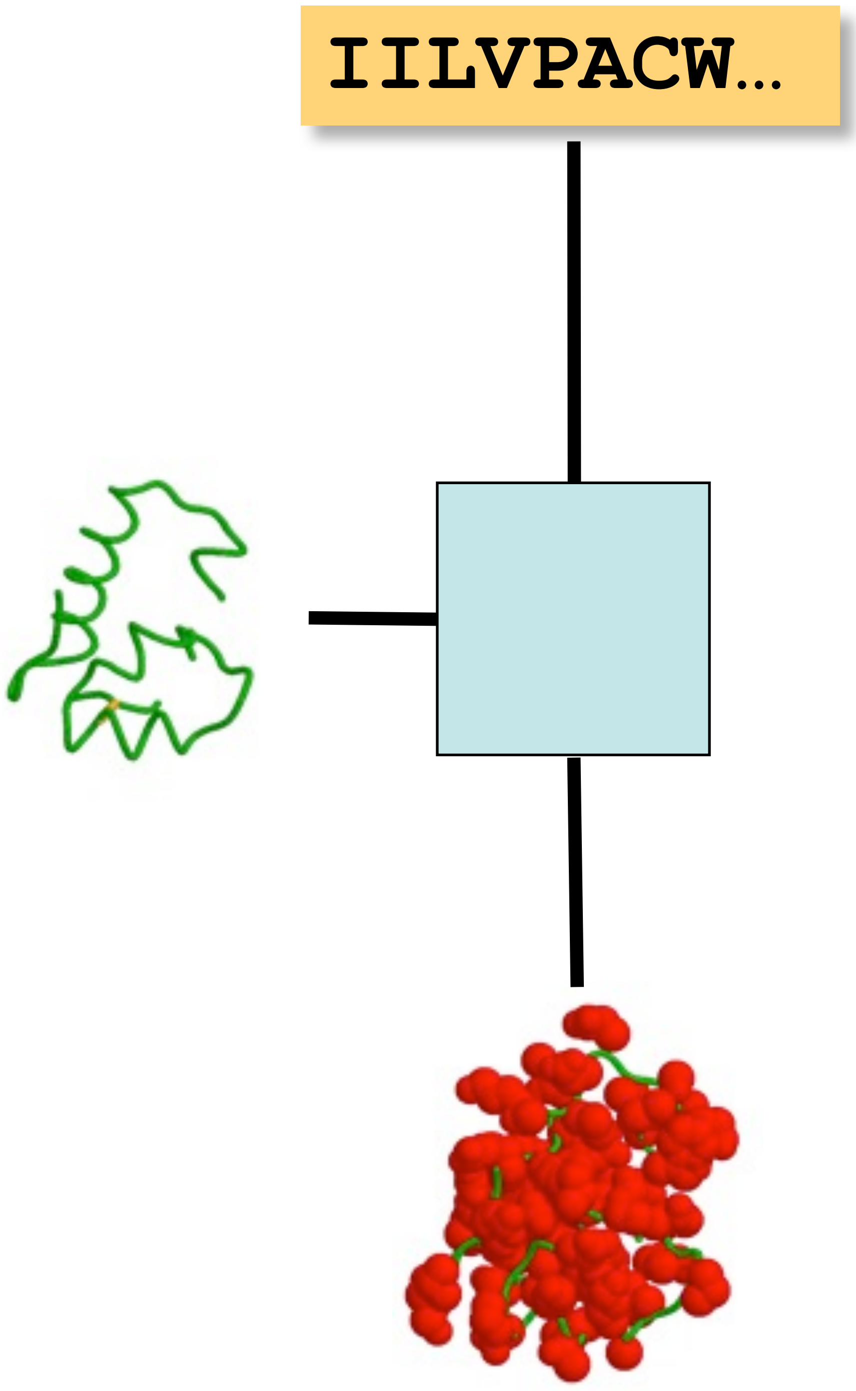


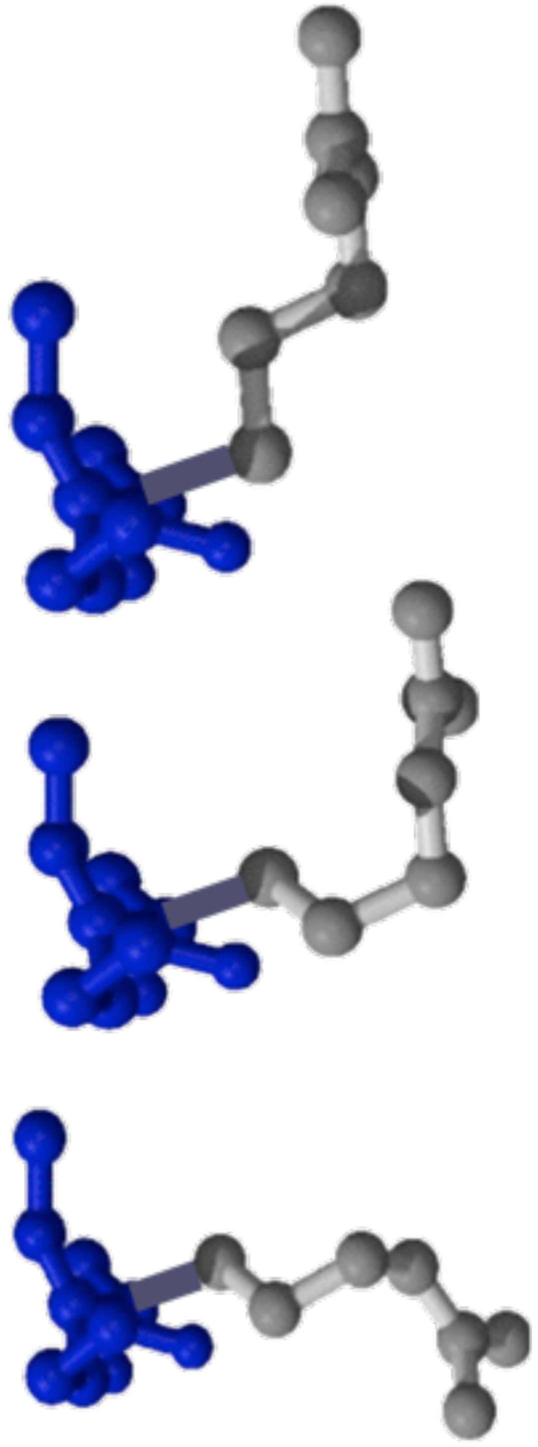
Side-chain Positioning Problem

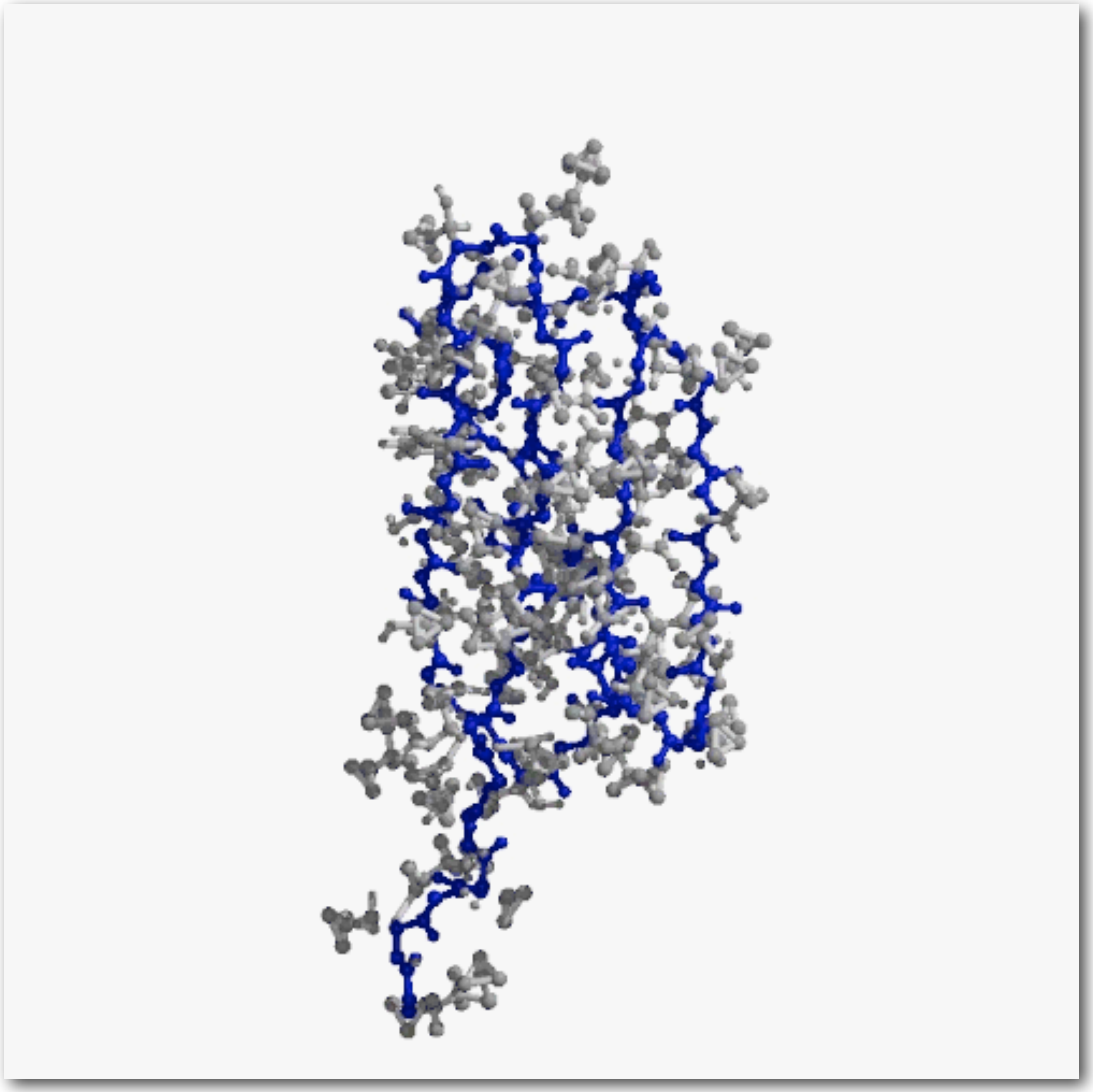
Given:
- fixed backbone
- amino acid sequence

Find the 3D positions for the side-chains that minimize the energy of the structure

Assume lowest energy is best




3 rotamers of Arg



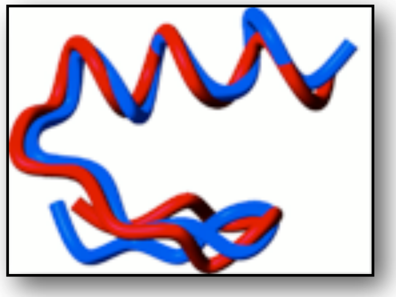
Applications

Homology modeling

- Rapid, low-cost structure determination

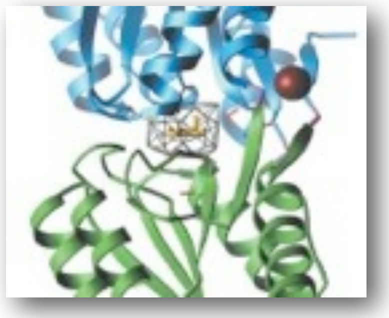
Protein design

- Find sequence that folds into a given shape
- e.g. redesign of zinc finger that folds without zinc, (Dahiyat+97)



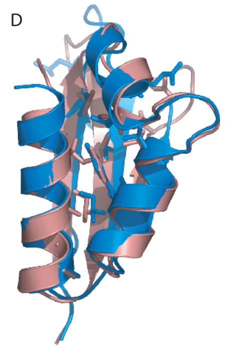
Ligand binding

- e.g. novel binding pockets (Looger+03)



Subroutine in flexible backbone prediction

- e.g. (Bradley+,2005)



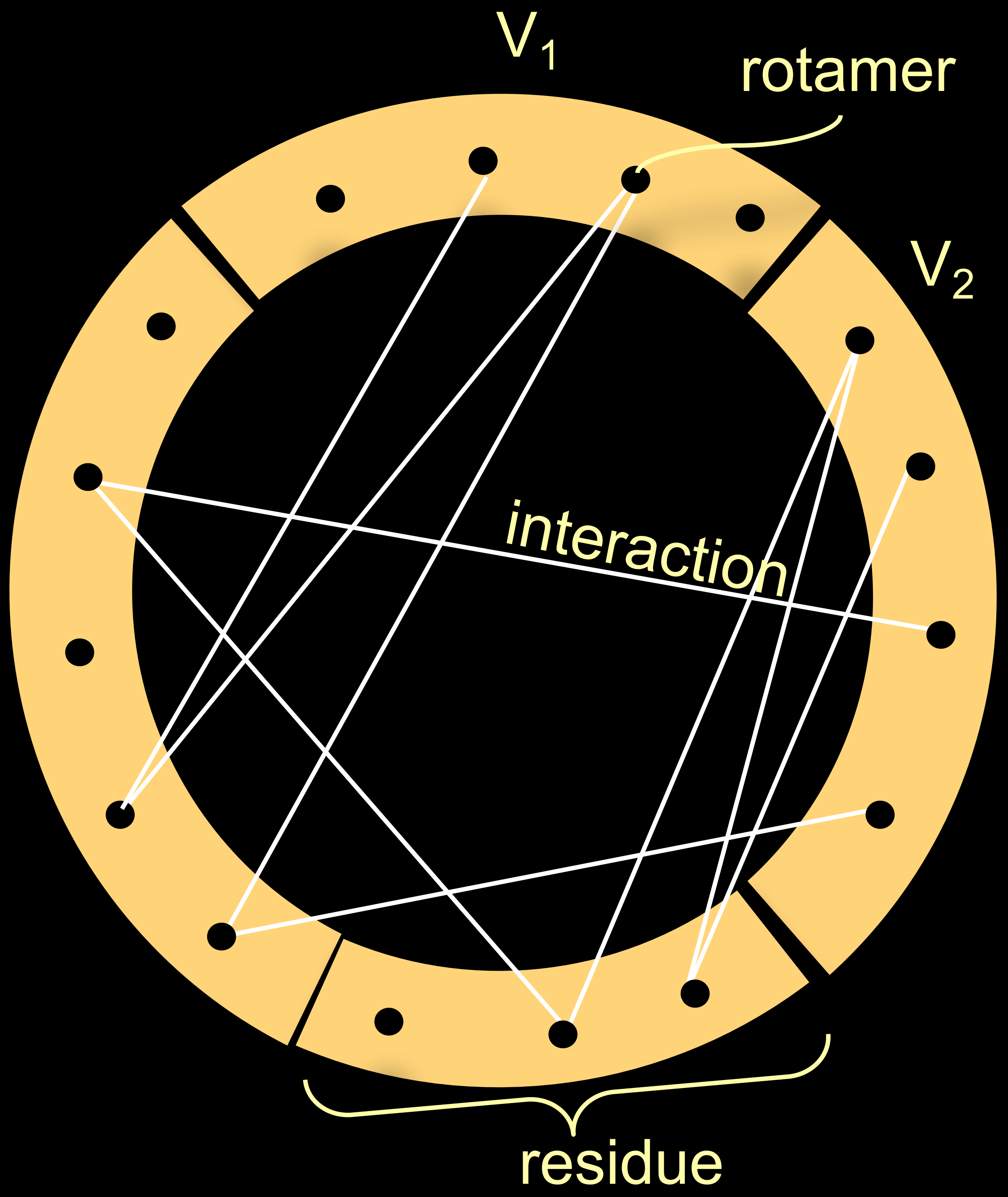
Graph Problem

Graph with part V_i for each side chain:

- node for each rotamer
- edge {u,v} represents the interaction between u and v

Weights:

- E(u) = self-energy
- E(u,v) = interaction energy

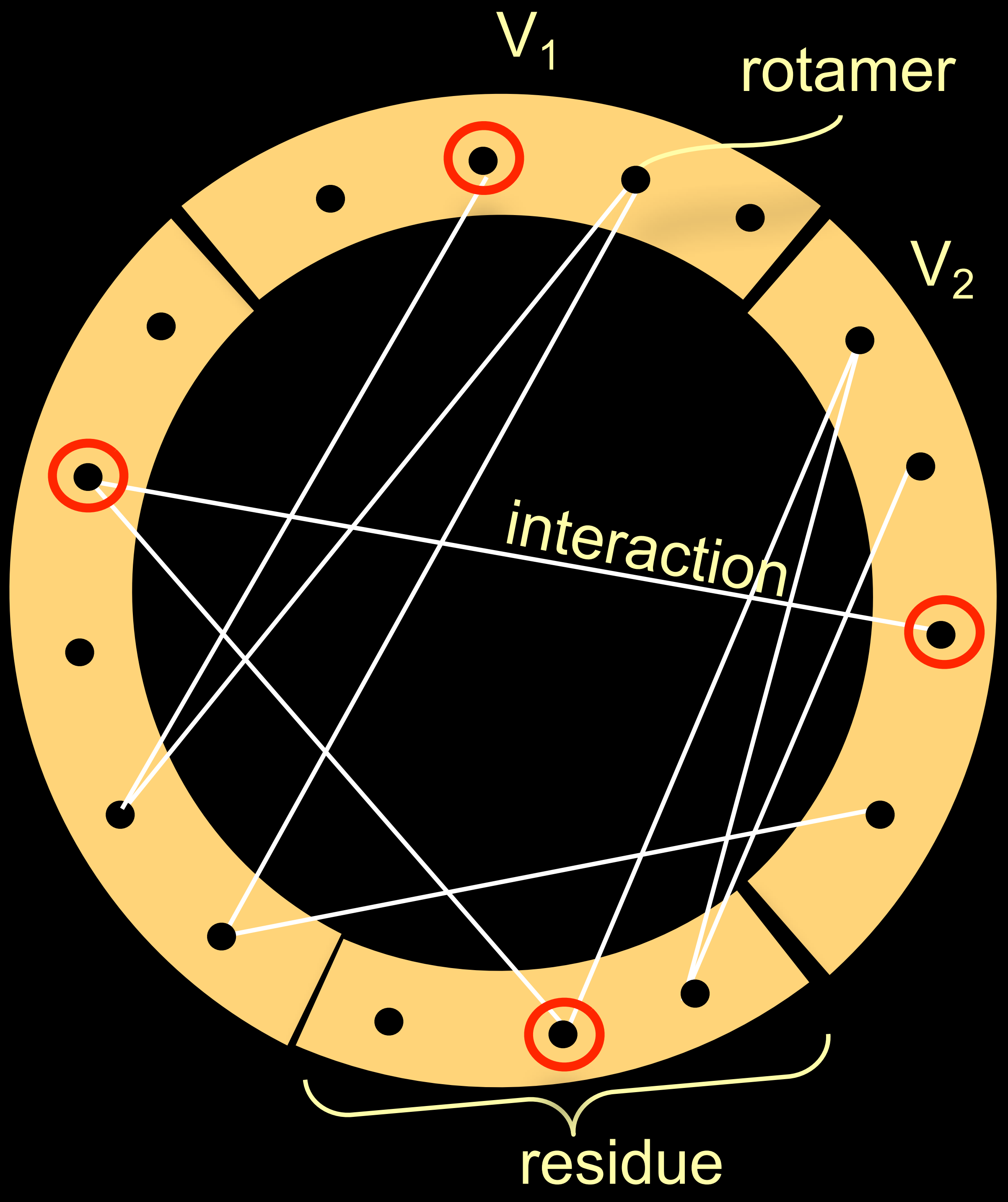


Graph Problem

Solution is one node from each part

Cost of solution is cost of induced subgraph

Goal: pick one node from each position to minimize the cost of the induced subgraph



Hardness

NP-hard to approximate the minimum energy within a factor of cn where $c > 0$ and n = # of rotamers (CKS04)

$\Rightarrow$ Little hope for a fast algorithm that guarantees good solutions

Proposed Solutions

Local search	
– Monte Carlo	(Xiang+01)
– Simulated annealing	(Lee+91, Kuhlman+00)
– Many others	
Graph heuristics	
– Scwrl	(Bower+97, Canutescu+03)
– Dead-end elimination	(Desmet+92,...)
– & others	(Samudrala+98, Bahadur+04)
Mathematical programming	
– Semidefinite	**(Chazelle, K, Singh, 04)**
– Linear/integer	(Althaus+00; Eriksson+01; **KCS, 05**)

⇒ Flexible, practical framework to find optimal solutions.

Integer Programming

- General optimization framework:
  - Describe system by set of variables

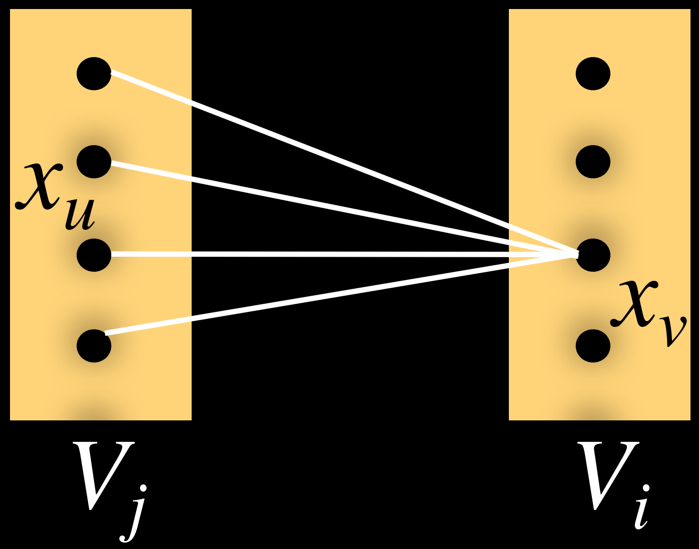
IP :=
- Minimize a linear function.
- Subject to linear constraints (= or $\geq$).
- While requiring the variables to be {0,1}.

- Computationally hard, but many advanced solver packages:
  - CPLEX, COIN-OR, ABACUS, FortMP, LINGO, …

Integer Programming Formulation

Binary variables x_u for each node

Binary variables x_{uv} for each edge



Minimize $$\sum_u E_u x_u + \sum_{u,v} E_{uv} x_{uv}$$

subject to:

1. $$\sum_{u \in V_j} x_u = 1$$ for every residue j

2. $$\sum_{u \in V_j} x_{uv} = x_v$$ for every residue j, node v

Why Integer Programming?

Optimal solutions

- Eliminate any effect of local search
- Help to improve energy functions
- Assess quality of heuristic methods

Very good IP solvers available

Ensemble of near-optimal solutions

- Several design candidates
- Confidence in solution

Linear Programming Relaxation

$$x_u, x_{uv} \in \{0, 1\}$$

Integer Program

Enforcing binary constraints is hard.

Guarantees finding an optimal choice of rotamers.

$$0 \leq x_u, x_{uv} \leq 1$$

Linear Program

Computationally easier.

May return fractional solution.

If integral, done.

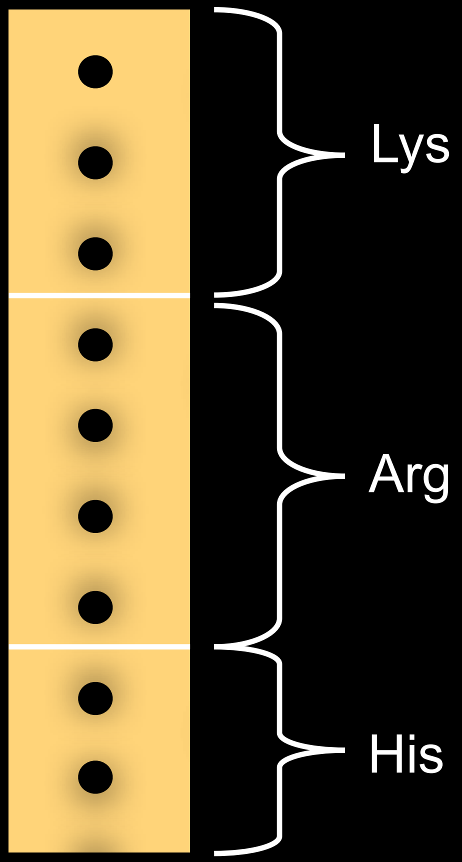
If not, either round or add new constraints

Design Problems

Want to design a sequence that will fold into a given backbone
- Output is an amino acid sequence

Assumption: a sequence that fits well onto this backbone will fold into it

Put rotamers for several amino acids into each graph part



Redesign Tests

- Redesigned 25 protein cores
  - Energy function best suited to solvent inaccessible residues

  $\Rightarrow$ Fixed surface residues

- Group amino acids into classes:

  AVILMF / HKR / DE / TQNS / WY / P / C / G

- Problem sizes:
  - 11 to 124 residues
  - 552 to 6,655 rotamers

Design Results

Redesigned 25 protein cores

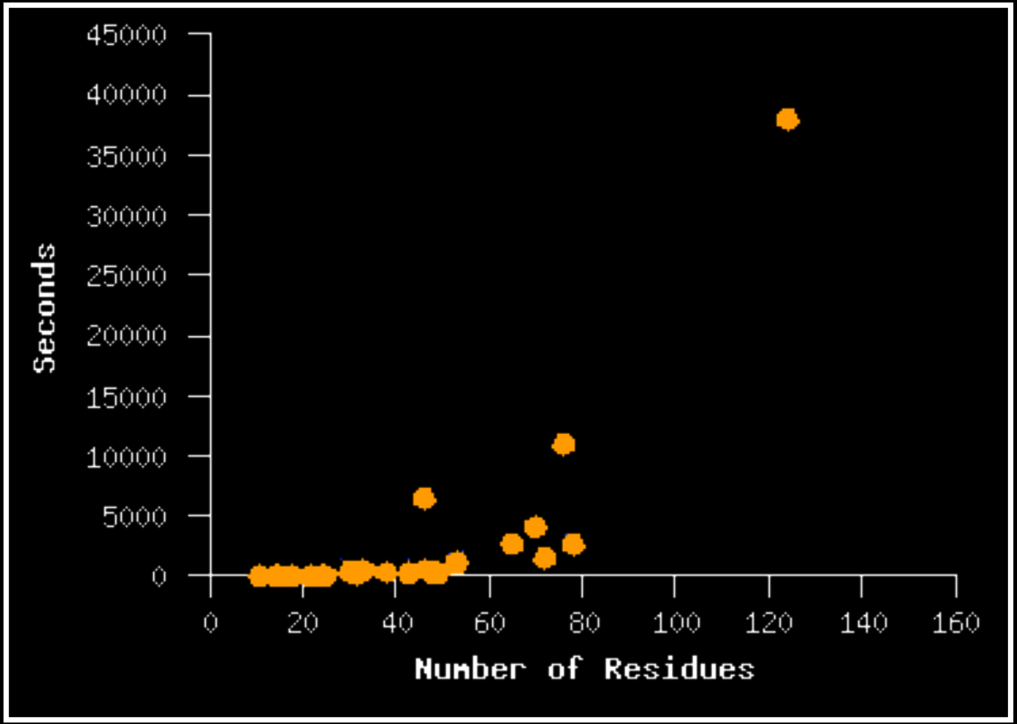
- 11 to 124 residues
- 552 to 6,655 nodes

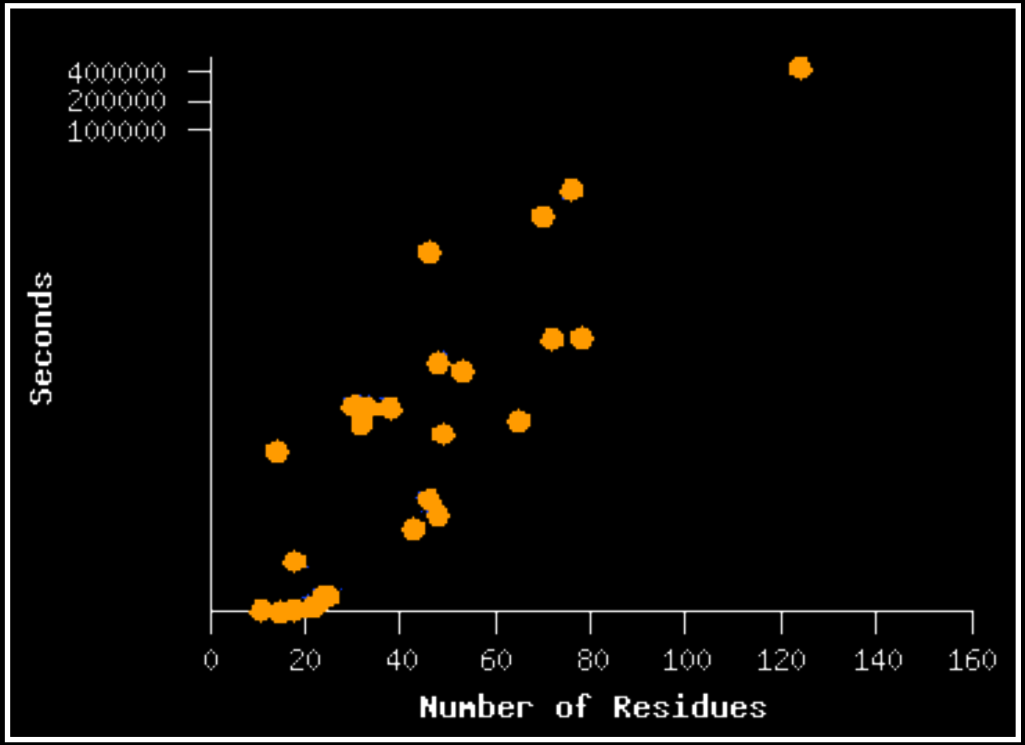
LP much slower (20 hours)

Only 6 integral out of 25

After DEE, can solve IP for remaining problems:

- one took 125 hours
- remaining 18 took 13 hours



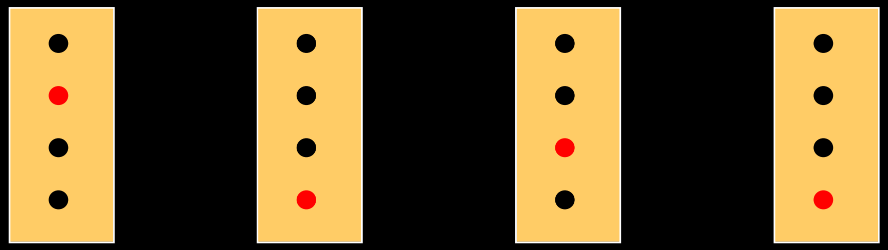


Near-Optimal Solutions

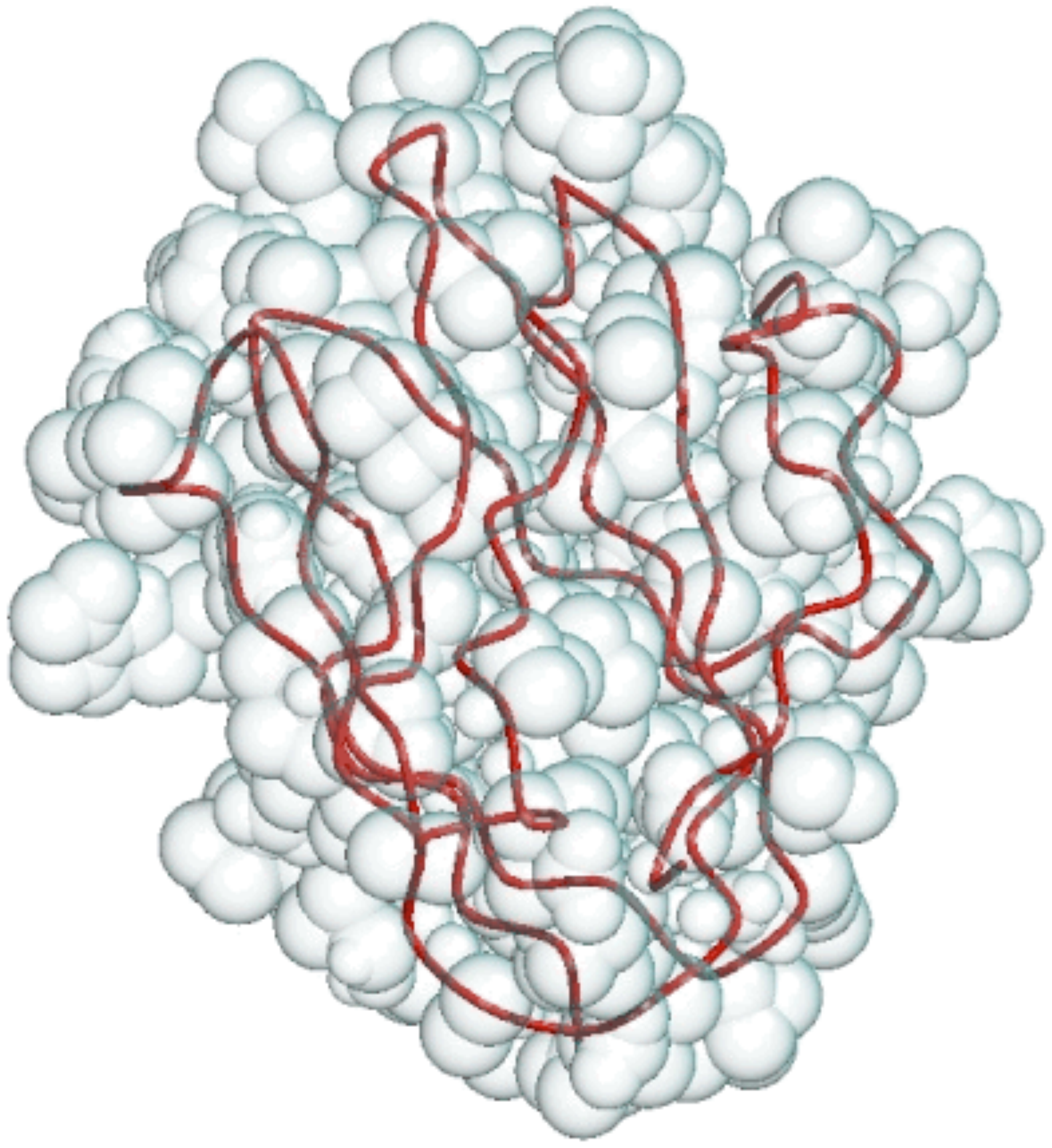
- Near-optimal solutions are useful:
  - Several candidates for protein design
  - Confidence in solution
- Can be found with integer program formulation
- To exclude m previously found solutions, add constraints:

$$\sum_{u \in S_k} x_u \leq p - 1 \quad \text{for } k = 1, \ldots, m$$

where S_k is set of chosen nodes for solution k



Near-Optimal Solutions


1aac - best 597 solutions.

⟸ Required only that some residue change

- Can also require, say, core residue change
- Or force several residues to move at once

Thus,

- Side-chain positioning is a biologically useful problem with a nice combinatorial problem behind it
- Linear / integer programming effective method for finding optimal side-chain positions
- Empirical difficulty ≠ theoretical hardness
- Design problems yield harder search problems than homology modeling