S. Canzar[1] N. C. Toussaint[2] G. W. Klau[1]

An Exact Algorithm for Side-Chain Placement in Protein Design

[1]Centrum Wiskunde & Informatica, Amsterdam, The Netherlands

[2]University of Tübingen, Center for Bioinformatics, Tübingen, Germany

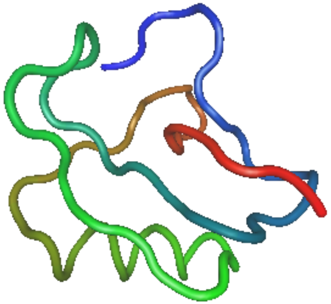
Life sciences seminar, 13 May 2011

Proteins

- key players in virtually all biological processes
- function mostly determined by its 3D structure

Proteins

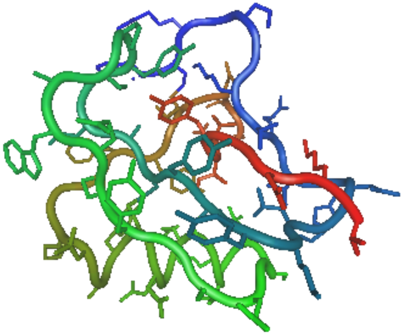
- key players in virtually all biological processes
- function mostly determined by its 3D structure



- sequence of amino acids (=residues) on *backbone*

Proteins

- key players in virtually all biological processes
- function mostly determined by its 3D structure



- sequence of amino acids (=residues) on *backbone*
- each amino acid has flexible side-chain

The Side-Chain Placement Problem

Side-Chain Placement (SCP)

Given a fixed backbone, place the amino acid side-chains on the backbone in the energetically most favorable conformation.

The Side-Chain Placement Problem

Side-Chain Placement (SCP)

Given a fixed backbone, place the amino acid side-chains on the backbone in the energetically most favorable conformation.

- Homology Modeling:
  - use known backbone of similar protein to predict protein structure
  - sequence of amino acids known

The Side-Chain Placement Problem

CWI

Side-Chain Placement (SCP)

Given a fixed backbone, place the amino acid side-chains on the backbone in the energetically most favorable conformation.

- Homology Modeling:
  - use known backbone of similar protein to predict protein structure
  - sequence of amino acids known
- Protein Design
  - find sequence of amino acids that will fold into given backbone
  - applications in pharmaceutical and biotechnological industry

The Side-Chain Placement Problem

Side-Chain Placement (SCP)

Given a fixed backbone, place the amino acid side-chains on the backbone in the energetically most favorable conformation.

- Homology Modeling:
  - use known backbone of similar protein to predict protein structure
  - sequence of amino acids known
- Protein Design
  - find sequence of amino acids that will fold into given backbone
  - applications in pharmaceutical and biotechnological industry

Same mathematical abstraction for both problems!

Discrete Search Space

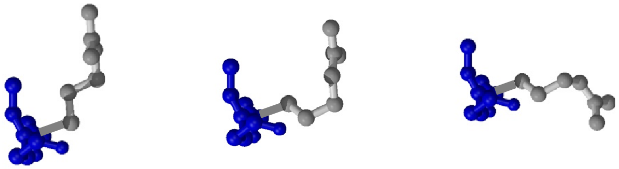
- The side-chain conformation of a residue is discretized into a finite number of states.

Discrete Search Space

- The side-chain conformation of a residue is discretized into a finite number of states.
- Each *rotamer* represents a set of similar, statistically preferred, side-chain conformations

Discrete Search Space

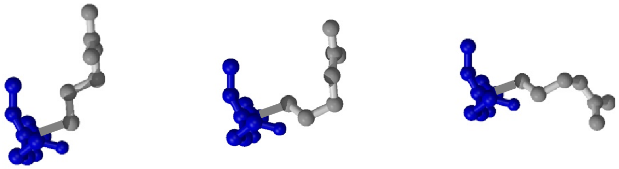
- The side-chain conformation of a residue is discretized into a finite number of states.
- Each *rotamer* represents a set of similar, statistically preferred, side-chain conformations
- Backbone-(in)dependent rotamer library (Dunbrack et al.)



(C. Kingsford)

Discrete Search Space

- The side-chain conformation of a residue is discretized into a finite number of states.
- Each *rotamer* represents a set of similar, statistically preferred, side-chain conformations
- Backbone-(in)dependent rotamer library (Dunbrack et al.)


(C. Kingsford)

⇒ Combinatorial search problem!

Energy Function

Quality of rotamer assignment by energy function:

- Singleton scores:
  - interaction between backbone and chosen rotamer
  - intrinsic energy of rotamer
- Pairwise scores:
  - van der Waals
  - electrostatic
  - hydrogen bonding
  - ...

Energy Function

Quality of rotamer assignment by energy function:

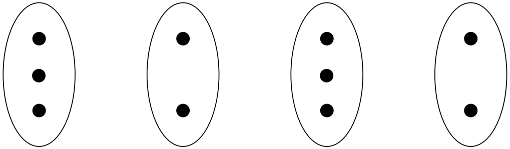
- Singleton scores:
  - interaction between backbone and chosen rotamer
  - intrinsic energy of rotamer
- Pairwise scores:
  - van der Waals
  - electrostatic
  - hydrogen bonding
  - ...

Goal: Find minimum energy solution!

Graph-Theoretic Formulation

Represent protein with k residues by k-partite graph $G = (V, E)$:

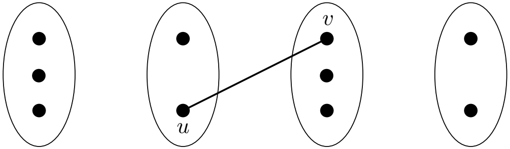
- part V_i for each residue i
- node $v \in V_i$ for each candidate rotamer of residue i



Graph-Theoretic Formulation

Represent protein with k residues by k-partite graph $G = (V, E)$:

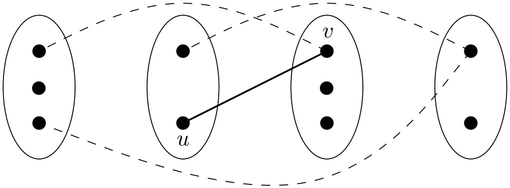
- part V_i for each residue i
- node $v \in V_i$ for each candidate rotamer of residue i
- edge uv denotes interaction between u and v



Graph-Theoretic Formulation

Represent protein with k residues by k-partite graph $G = (V, E)$:

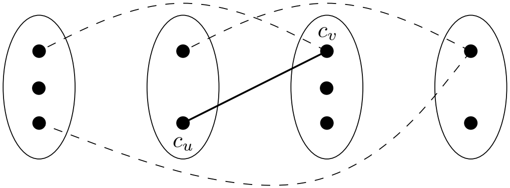
- part V_i for each residue i
- node $v \in V_i$ for each candidate rotamer of residue i
- edge uv denotes interaction between u and v



Graph-Theoretic Formulation

Represent protein with k residues by k-partite graph $G = (V, E)$:

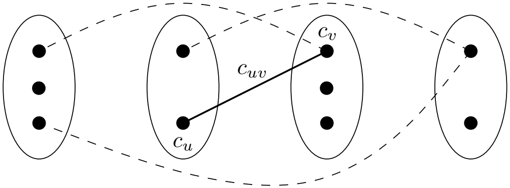
- part V_i for each residue i
- node $v \in V_i$ for each candidate rotamer of residue i
- edge uv denotes interaction between u and v
- node costs $c_v, v \in V$ = self-energy of rotamer v



Graph-Theoretic Formulation

Represent protein with k residues by k-partite graph $G = (V, E)$:

- part V_i for each residue i
- node $v \in V_i$ for each candidate rotamer of residue i
- edge uv denotes interaction between u and v
- node costs $c_v, v \in V$ = self-energy of rotamer v
- edge costs $c_{uv}, uv \in E$ = interaction energy of u and v



Problem SCP

Side-Chain Placement (SCP)

Given a k-partite graph $G = (V, E)$, $V = V_1, \cup \cdots \cup V_k$, with node costs $c_v, v \in V$, and edge costs $c_{uv}, uv \in E$, determine an assignment $a : [k] \mapsto V$ with $a(i) \in V_i$, such that cost

$$\sum_{i=1}^{k} c_{a(i)} + \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} c_{a(i)a(j)}$$

of induced subgraph is minimum.

- $\mathcal{NP}$-hard [Pierce, Winfree, 2002]
- inapproximable [Chazelle *et al.*, 2004]

Previous Work

Heuristic:

- Simulated Annealing
- Monte Carlo
- Belief Propagation

Previous Work

Heuristic:

- Simulated Annealing
- Monte Carlo
- Belief Propagation

Less accurate with with increasing problem size! [Voigt *et al.* 2000]

Previous Work

Heuristic:

- Simulated Annealing
- Monte Carlo
- Belief Propagation

Less accurate with with increasing problem size! [Voigt *et al.* 2000]

Exact:

- Dead end elimination + A^*
- Branch and Bound
- Tree decomposition
- Integer linear programming

Overview of the Approach

- exact approach
- based on ILP formulation by [Althaus *et al.*], [Kingsford *et al.*]
- Branch & Bound framework
- Lagrangian relaxation:
  - lower bounds by shortest path computation
  - Lagrangian dual: Subgradient Optimization
  - primal feasible solutions
- initial primal bound by randomized local search

Overview of the Approach

- exact approach
- based on ILP formulation by [Althaus *et al.*], [Kingsford *et al.*]
- Branch & Bound framework
- Lagrangian relaxation:
  - **lower bounds by shortest path computation**
  - Lagrangian dual: Subgradient Optimization
  - primal feasible solutions
- initial primal bound by randomized local search

An ILP formulation

Variables:

- $x_u \in \{0,1\}, u \in V_i$, indicates wheter $a(i) = u$.
- $y_{uv} \in \{0,1\}$: edge uv is contained in induced subgraph

An ILP formulation

Variables:

- $x_u \in \{0,1\}, u \in V_i$, indicates wheter $a(i) = u$.
- $y_{uv} \in \{0,1\}$: edge uv is contained in induced subgraph

Constraints: (Let $r(v) = i$ iff $v \in V_i$)

- Pick one rotamer per residue:

$$\sum_{v \in V_i} x_v = 1 \quad \forall i \in [k]$$

An ILP formulation

Variables:

- $x_u \in \{0,1\}, u \in V_i$, indicates wheter $a(i) = u$.
- $y_{uv} \in \{0,1\}$: edge uv is contained in induced subgraph

Constraints: (Let $r(v) = i$ iff $v \in V_i$)

- Pick one rotamer per residue:

$$\sum_{v \in V_i} x_v = 1 \quad \forall i \in [k]$$

- Select induced edges:

$$\sum_{u \in V_i} y_{uv} = x_v \quad \forall v \in V, i \neq r(v)$$

Lagrangian Relaxation

$$
\begin{aligned}
\min \quad & \sum_{v \in V} c_v x_v + \sum_{uv \in E} c_{uv} y_{uv} & \\
\text{s.t.} \quad & \sum_{v \in V_i} x_v = 1 & \forall i \in [k] \\
& \sum_{u \in V_i} y_{uv} = x_v & \forall v \in V, i \neq r(v) \\
& x_v, y_{uv} \in \{0, 1\} & \forall v \in V, uv \in E
\end{aligned}
$$

Lagrangian Relaxation

$$
\begin{aligned}
\min \quad & \sum_{v\in V} c_v x_v + \sum_{uv\in E} c_{uv} y_{uv} & \\
\text{s.t.} \quad & \sum_{v\in V_i} x_v = 1 & \forall i \in [k] \\
& \sum_{u\in V_i} y_{uv} = x_v & \forall v \in V, i < r(v) \\
& \sum_{u\in V_i} y_{uv} = x_v & \forall v \in V, i > r(v) \\
& x_v, y_{uv} \in \{0,1\} & \forall v \in V, uv \in E
\end{aligned}
$$

Lagrangian Relaxation

$$
\begin{aligned}
\min \quad & \sum_{v \in V} c_v x_v + \sum_{uv \in E} c_{uv} y_{uv} & \\
\text{s.t.} \quad & \sum_{v \in V_i} x_v = 1 & \forall i \in [k] \\
& \sum_{u \in V_i} y_{uv} = x_v & \forall v \in V, i < r(v) \\
& \sum_{u \in V_i} y_{uv} = x_v & \forall v \in V, i = r(v) + 1 \\
& \sum_{u \in V_i} y_{uv} = x_v & \forall v \in V, i > r(v) + 1 \\
& x_v, y_{uv} \in \{0, 1\} & \forall v \in V, uv \in E
\end{aligned}
$$

Lagrangian Relaxation

$$
\begin{aligned}
\min \quad & \sum_{v \in V} c_v x_v + \sum_{uv \in E} c_{uv} y_{uv} && \\
\text{s.t.} \quad & \sum_{v \in V_i} x_v = 1 && \forall i \in [k] \\
& \sum_{u \in V_i} y_{uv} = x_v && \forall v \in V, i < r(v) \\
& \sum_{u \in V_i} y_{uv} = x_v && \forall v \in V, i = r(v) + 1 \\
& \cancelto{\text{dualize}}{\sum_{u \in V_i} y_{uv} = x_v} && \forall v \in V, i > r(v) + 1 \\
& x_v, y_{uv} \in \{0, 1\} && \forall v \in V, uv \in E
\end{aligned}
$$

Lagrangian Relaxation

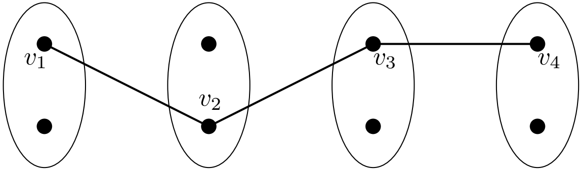
$$
\begin{aligned}
\min \quad & \sum_{v \in V} c_v x_v + \sum_{uv \in E} c_{uv} y_{uv} \quad + \sum_{v \in V} \sum_{i > r(v)+1} \lambda_v^i \cdot \Big(x_v - \sum_{u \in V_i} y_{uv}\Big) \\
\text{s.t.} \quad & \sum_{v \in V_i} x_v = 1 && \forall i \in [k] \\
& \sum_{u \in V_i} y_{uv} = x_v && \forall v \in V, i < r(v) \\
& \sum_{u \in V_i} y_{uv} = x_v && \forall v \in V, i = r(v) + 1 \\
& x_v, y_{uv} \in \{0, 1\} && \forall v \in V, uv \in E
\end{aligned}
$$

Lagrangian Subproblem

$$\sum_{v \in V_i} x_v = 1 \qquad \forall i \in [k]$$

$$\sum_{u \in V_i} y_{uv} = x_v \qquad \forall v \in V, i = r(v) - 1$$

$$\sum_{u \in V_i} y_{uv} = x_v \qquad \forall v \in V, i = r(v) + 1$$



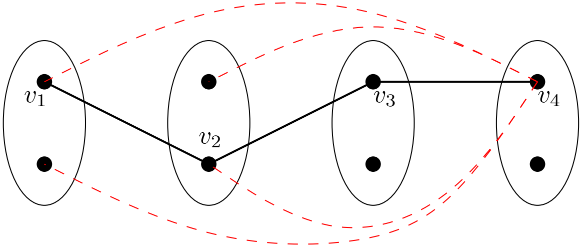
Lagrangian Subproblem

$$\sum_{v \in V_i} x_v = 1 \qquad \forall i \in [k]$$

$i < r(v)$

$$\sum_{u \in V_i} y_{uv} = x_v \qquad \forall v \in V, i = r(v) - 1$$

$$\sum_{u \in V_i} y_{uv} = x_v \qquad \forall v \in V, i = r(v) + 1$$

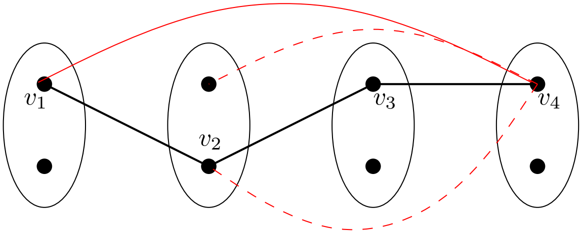


Lagrangian Subproblem

$$\sum_{v \in V_i} x_v = 1 \qquad \forall i \in [k]$$

$$\sum_{u \in V_i} y_{uv} = x_v \qquad \forall v \in V, \overset{i < r(v)}{\cancel{i = r(v) - 1}}$$

$$\sum_{u \in V_i} y_{uv} = x_v \qquad \forall v \in V, i = r(v) + 1$$

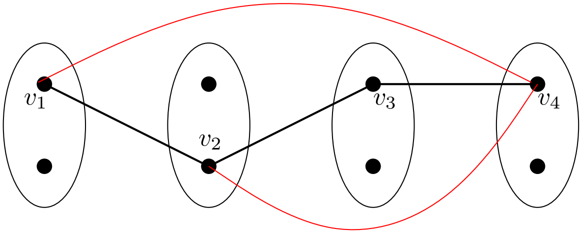


Lagrangian Subproblem

$$\sum_{v \in V_i} x_v = 1 \qquad \forall i \in [k]$$

$$\sum_{u \in V_i} y_{uv} = x_v \qquad \forall v \in V, \cancel{i = r(v) - 1} \xrightarrow{} i < r(v)$$

$$\sum_{u \in V_i} y_{uv} = x_v \qquad \forall v \in V, i = r(v) + 1$$



Solving the Lagrangian Subproblem

$$\text{minimize} \sum_{v\in V}(c_v + \sum_{i>r(v)+1} \lambda_v^i)x_v + \sum_{\substack{uv\in E\\ r(u)<r(v)}} (c_{uv} - \lambda_u^{r(v)})y_{uv}$$

Consider the *profit* δ of a node v:

$$\delta(v) = (c_v + \sum_{i>r(v)+1} \lambda_v^i) + \sum_{i=1}^{r(v)-2} \min_{u\in V_i}(c_{uv} - \lambda_u^{r(v)})$$

Solving the Lagrangian Subproblem

$$\text{minimize} \sum_{v\in V}(c_v + \sum_{i>r(v)+1} \lambda_v^i)x_v + \sum_{\substack{uv\in E\\ r(u)<r(v)}} (c_{uv} - \lambda_u^{r(v)})y_{uv}$$

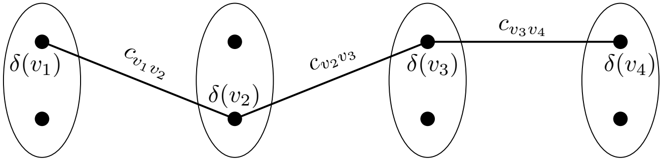
Consider the *profit* δ of a node v:

$$\delta(v) = (c_v + \sum_{i>r(v)+1} \lambda_v^i) + \sum_{i=1}^{r(v)-2} \min_{u\in V_i}(c_{uv} - \lambda_u^{r(v)})$$

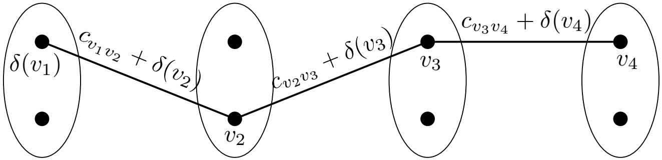
Then the score of a feasible path $p = (v_1, v_2, \dots, v_k)$ is:

$$\sum_{i=1}^{k} \delta(v_i) + \sum_{i=1}^{k-1} c_{v_i v_{i+1}}$$

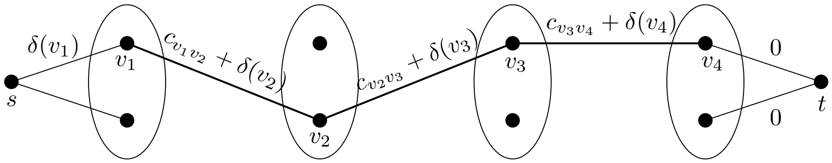
Lagrangian Bound by Shortest Path



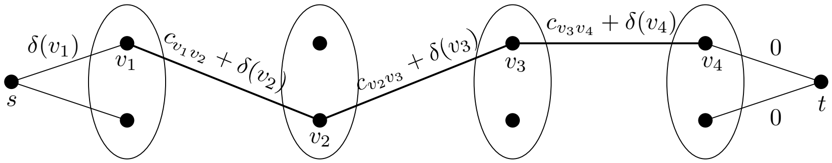
Lagrangian Bound by Shortest Path


$\delta(v_1)$
$c_{v_1 v_2} + \delta(v_2)$
v_2
$c_{v_2 v_3} + \delta(v_3)$
v_3
$c_{v_3 v_4} + \delta(v_4)$
v_4

Lagrangian Bound by Shortest Path

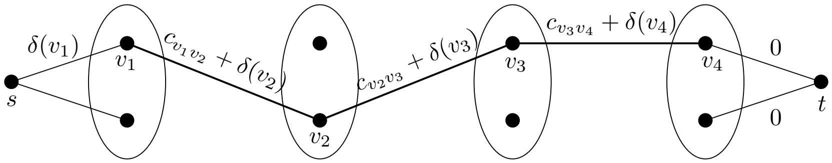

$\delta(v_1)$
v_1
s
$c_{v_1 v_2} + \delta(v_2)$
v_2
$c_{v_2 v_3} + \delta(v_3)$
v_3
$c_{v_3 v_4} + \delta(v_4)$
v_4
0
0
t

Lagrangian Bound by Shortest Path


$\delta(v_1)$
v_1
$c_{v_1 v_2} + \delta(v_2)$
v_2
$c_{v_2 v_3} + \delta(v_3)$
v_3
$c_{v_3 v_4} + \delta(v_4)$
v_4
0
0
s
t

$\Rightarrow$ Shortest path in time linear in the number of edges!

Lagrangian Bound by Shortest Path



$\delta(v_1)$
v_1
$c_{v_1 v_2} + \delta(v_2)$
v_2
$c_{v_2 v_3} + \delta(v_3)$
v_3
$c_{v_3 v_4} + \delta(v_4)$
v_4
0
0
s
t

$\Rightarrow$ Shortest path in time linear in the number of edges!

$\Rightarrow$ Optimal solution in time $\mathcal{O}(|V|^2)$

Experimental Setting

- C++, LEDA, BALL
- compare to CPLEX [Kingsford *et al.*]
  - DEE, TreePack, R3 do not allow multiple candidate amino acids
  - treewidth $\approx 10 - 20$ for small instances
  - reduced instances too large
- 2.26 GHz Intel Quad Core processors, 4 GB RAM, 64 bit Linux
- time limit 12 hours, memory limit 16 GB
- suboptimal rotamers eliminated in preprocessing
- 2 different benchmark sets

Experimental Results (I)

Protein design energy files by Kingsford *et al.*

- 25 proteins, 11-124 flexible residue positions
- surface residues fixed, $\leq$ 6 amino acids at core positions

Instance			Lagrangian B&B			CPLEX	
Name	#res	#rot	N	H	time/s	time/s	S
1c9o	66	1130	2	1	0.33	1.96	5.9
1cex	197	2556	9	2	13.37	33.25	2.5
1cz9	139	2332	1	0	3.7	18.10	4.9
1czp	98	1170	1	0	0.54	4.32	8.0
1d4t	104	1636	1	0	0.37	2.36	6.4
1mfm	153	2134	25	5	21.89	145.63	6.7
1plc	99	1156	2	1	1.50	6.08	4.1
1qj4	256	4080	313	10	8,424.56	31,636.40	3.8
1qq4	198	2045	16	4	32.56	38.89	1.2
1rcf	169	2396	2	1	4.76	12.85	2.7
2pth	193	3077	66	6	322.28	518.51	1.6
3lzt	129	2074	7	2	3.20	10.64	3.3
5p21	166	2874	52	4	106.09	115.01	1.1
7rsa	124	1958	1	0	0.78	3.31	4.2

Experimental Results (II)

Protein design instances from Yanover *et al.*

- 97 proteins, 40-180 flexible residue positions
- at each position all 20 amino acids allowed
- Rosetta energy function

Instance			Lagrangian B&B			CPLEX	
Name	#res	#rot	N	H	time/s	time/s	S
1brf	44	3524	9	4	293.97	469.87	1.6
1bx7	25	1048	1	0	0.54	5.77	10.7
1d3b	66	5732	1	0	530.37	9,577.68	18.1
1en2	59	2689	1	0	19.41	39.94	2.1
1ezg	58	1653	2	1	185.11	441.23	2.4
1g6x	51	3190	1	0	23.96	160.64	6.7
1gcq	65	5442	4	2	903.82	5,270.08	9.8
1i07	52	3186	4	1	187.45	166.20	0.9
1kth	49	3330	18	4	798.57	642.42	0.8
1rb9	43	3307	7	2	127.93	9,535.72	74.5
1sem	54	4348	192	8	5,020.55	6,470.37	1.3
4rxn	45	3636	1	0	220.33	3,034.57	13.8

Conclusion and Outlook

- Combinatorial relaxation outperforms LP relaxation
- Performance depends on energy function and number of allowed amino acids
- Large real-world instances solved optimally in reasonable time
- Strong heuristics on specific problem classes [Sontag *et al.*]
- Wide range of applications:
  - image understanding
  - error correcting codes
  - frequency assignment in telecommunication