S. Canzar\textsuperscript{1}  N. C. Toussaint\textsuperscript{2}  G. W. Klau\textsuperscript{1}

An Exact Algorithm for Side-Chain Placement in Protein Design

\textsuperscript{1}Centrum Wiskunde & Informatica, Amsterdam, The Netherlands

\textsuperscript{2}University of Tübingen, Center for Bioinformatics, Tübingen, Germany

SEA’11
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.
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.)

⇒ 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
  - ...

Goal: Find minimum energy solution!
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 $c_v, v \in V = \text{self-energy of rotamer } v$
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 $c_v$, $v \in V = \text{self-energy of rotamer } v$
- edge costs $c_{uv}$, $uv \in E = \text{interaction energy of } u \text{ and } v$
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

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

Exact:
- Dead end elimination + A* Branch and Bound
- Tree decomposition
- Integer linear programming
Previous Work

Heuristic:
- Simulated Annealing
- Monte Carlo
- Belief Propagation

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

Heuristic:
- Simulated Annealing
- Monte Carlo
- Belief Propagation

Less accurate 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 whether $a(i) = u$.
- $y_{uv} \in \{0, 1\}$: edge $uv$ is contained in induced subgraph
An ILP formulation

Variables:

- $x_u \in \{0, 1\}, u \in V_i$, indicates whether $a(i) = u$.
- $y_{uv} \in \{0, 1\}$: 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 whether \( a(i) = u \).
- \( y_{uv} \in \{0, 1\} \): 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

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

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

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

$$\min \sum_{v \in V} c_v x_v + \sum_{uv \in E} c_{uv} y_{uv}$$

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

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

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

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

$$x_v, y_{uv} \in \{0, 1\} \quad \forall v \in V, uv \in E$$
Lagrangian Relaxation

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

\[ \sum_{v \in V_i} x_v = 1 \quad \forall i \in [k] \]
\[ \sum_{u \in V_i} y_{uv} = x_v \quad \forall v \in V, i = r(v) - 1 \]
\[ \sum_{u \in V_i} y_{uv} = x_v \quad \forall v \in V, i = r(v) + 1 \]
Lagrangian Subproblem

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

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

\[ \sum_{u \in V_i} y_{uv} = x_v \quad \forall v \in V, i = r(v) + 1 \]
Lagrangian Subproblem

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

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

\[
\sum_{u \in V_i} y_{uv} = x_v \quad \forall v \in V, i = r(v) + 1
\]
Lagrangian Subproblem

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

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

\[ \sum_{u \in V_i} y_{uv} = x_v \quad \forall v \in V, i = r(v) + 1 \]
Solving the Lagrangian Subproblem

\[
\text{minimize} \quad \sum_{v \in V} (c_v + \sum_{i > r(v)+1} \lambda_i^v x_v) + \sum_{uv \in E} (c_{uv} - \lambda_r^v y_{uv})
\]

Consider the profit \( \delta \) of a node \( v \):

\[
\delta(v) = (c_v + \sum_{i > r(v)+1} \lambda_i^v) + \sum_{i=1}^{r(v)-2} \min_{u \in V_i} (c_{uv} - \lambda_u^{r(v)})
\]
Solving the Lagrangian Subproblem

\[
\text{minimize } \sum_{v \in V} (c_v + \sum_{i > r(v) + 1} \lambda^i_v)x_v + \sum_{uv \in E_{r(u) < r(v)}} (c_{uv} - \lambda^{r(v)}_u)y_{uv}
\]

Consider the profit \( \delta \) of a node \( v \):

\[
\delta(v) = (c_v + \sum_{i > r(v) + 1} \lambda^i_v) + \sum_{i=1}^{r(v)-2} \min_{u \in V_i} (c_{uv} - \lambda^{r(v)}_u)
\]

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

\[
\sum_{i=1}^{k} \delta(v_i) + \sum_{i=1}^{k-1} c_{v_i v_{i+1}}
\]
Lagrangian Bound by Shortest Path

\[ \delta(v_1) \delta(v_2) \delta(v_3) \delta(v_4) c_{v_1 v_2} c_{v_2 v_3} c_{v_3 v_4} \]

\[ \Rightarrow \text{Shortest path in time linear in the number of edges!} \]

\[ \Rightarrow \text{Optimal solution in time } O(|V|^2) \]
Lagrangian Bound by Shortest Path

\[ \delta(v_1) c_{v_1v_2} + \delta(v_2) c_{v_2v_3} + \delta(v_3) c_{v_3v_4} + \delta(v_4) \]

⇒ Shortest path in time linear in the number of edges!

⇒ Optimal solution in time \( O(|V|^2) \)
Lagrangian Bound by Shortest Path

\[ \delta(v_1) \]
\[ c_{v_1v_2} + \delta(v_2) \]
\[ v_2 \]
\[ c_{v_2v_3} + \delta(v_3) \]
\[ v_3 \]
\[ c_{v_3v_4} + \delta(v_4) \]
\[ v_4 \]
\[ 0 \]
\[ t \]

⇒ Shortest path in time linear in the number of edges!

Optimal solution in time \( O(|V|^2) \)

S. Canzar
An Exact Algorithm for Side-Chain Placement in Protein Design
Lagrangian Bound by Shortest Path

$cv_1v_2 + \delta(v_2)$

$c_{v_2v_3} + \delta(v_3)$

$c_{v_3v_4} + \delta(v_4)$

$\delta(v_1)$

$s$

$v_1$

$v_2$

$v_3$

$v_4$

$t$

$\Rightarrow$ Shortest path in time linear in the number of edges!
Lagrangian Bound by Shortest Path

\[ \delta(v_1) \quad c v_1 v_2 + \delta(v_2) \quad c v_2 v_3 + \delta(v_3) \quad c v_3 v_4 + \delta(v_4) \quad 0 \quad 0 \]

\[ \delta(v_1) \quad v_1 \quad v_2 \quad v_3 \quad v_4 \quad 0 \quad 0 \quad t \]

⇒ Shortest path in time linear in the number of edges!
⇒ 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

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 | Name  | #res | #rot | N | H | time/s | Lagrangian B&B | CPLEX | 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