Algorithms for Computational Protein Design

Thomas Schiex

▶ **To cite this version:**

Thomas Schiex. Algorithms for Computational Protein Design. 3rd cycle. International Winter School on "Algorithms in Structural Bioinformatics – Computational Protein Design" (Algorithms for Computational Protein Design), 2017. hal-02786907

HAL Id: hal-02786907
https://hal.inrae.fr/hal-02786907v1

Submitted on 5 Jun 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Advanced combinatorial optimization methods for protein design

The quest for Minimum Energy and Maximum Fitness Conformations

Thomas Schiex

November/December 2017

Cargèse, Corsica, France

Some preliminary words

Who am I ?

- a Computer Scientist specialized in discrete optimization,
- developing Artificial Intelligence techniques,
- applying them in bioinformatics (genetics, DNA, RNA).
- learning protein molecular modeling and biophysics.

Some preliminary words

Who am I ?

- a Computer Scientist specialized in discrete optimization,
- developing Artificial Intelligence techniques,
- applying them in bioinformatics (genetics, DNA, RNA).
- learning protein molecular modeling and biophysics.

I'm happy to learn more

- do tell me how to improve my understanding of bio-molecules
- during the presentation (if this will help others)
- after the presentation (otherwise)

The Computational Design Problem

Informal definition

Produce a sequence s of amino-acids that *spontaneously adopts* a conformation X that *performs some function*.

The Computational Design Problem

Informal definition

Produce a sequence s of amino-acids that *spontaneously adopts* a conformation X that *performs some function*.

Our assumptions on Fold and Function[3]

- The stability of a sequence s in a given conformation X can be estimated through a real valued energy function $E(s, X)$.

$$p_s(X) \propto e^{-\frac{E(s,X)}{k_B T}}$$

- The "fitness for purpose" of a sequence s in a given conformation X can be estimated through a real valued energy function $F(s, X)$.

What defines a conformation ?

Conformation

- quaternary structure: rigid body transforms ρ (docking)
- backbone structure: dihedral angles θ (backbone design)
- sequence: choice of amino-acid per position (sequence design)
- side-chains: torsion angles χ for each side-chain (packing)

$E(s, X)$ and $F(s, X)$ are inaccurate

Refined definition

We want to identify a set $\{(s, X)_i\}$ of designs that:

- have reasonably good fitness $F(s, X)$
- spontaneously adopt conformation X:

$$E(s, X) = \min_X(E(s, X))$$

- are diversified: $\forall i, j, \Delta((s, X)_i, (s, X)_j) > \delta$ Δ: distance

$E(s, X)$ and $F(s, X)$ are inaccurate

Refined definition

We want to identify a set $\{(s, X)_i\}$ of designs that:

- have reasonably good fitness $F(s, X)$
- spontaneously adopt conformation X:

$$E(s, X) = \min_X(E(s, X))$$

- are diversified: $\forall i, j, \Delta((s, X)_i, (s, X)_j) > \delta$ Δ: distance

Challenging optimization problem

- very high dimensionality, continuous variables (χ, ρ)
- non linearities in $E(X, s)$ and possibly $F(X, s)$ too
- discrete set of possible sequences s (size 20^n)

Several brain.decades later

The "rigid backbone, discrete rotamers" approach

1. backbones orientations and shapes (ρ, θ) are optimized using simplified energy and fitness functions (centroid-based).
   1. the initial backbones θ are selected among known backbones or designed *de novo* by assembling fragments of known backbones or...
   2. their relative positions ρ are optimized by a variety of docking algorithms.
2. sequence s is discrete, so χ is discretized too.

Many brain.decades later

Rotamer libraries

Each amino-acid is associated with a set of possible side-chain conformations that represent its most usual conformations (in the PDB, possibly conditional on SS or local backbone torsion angles).

Many brain.decades later

Rotamer libraries

Each amino-acid is associated with a set of possible side-chain conformations that represent its most usual conformations (in the PDB, possibly conditional on SS or local backbone torsion angles).

Existing rotamer libraries[12]

Tuffery,[61] Penultimate,[37] Dunbrack[54] (bb-dependent),...

Many brain.decades later

Rotamer libraries

Each amino-acid is associated with a set of possible side-chain conformations that represent its most usual conformations (in the PDB, possibly conditional on SS or local backbone torsion angles).

Existing rotamer libraries[12]

Tuffery,[61] Penultimate,[37] Dunbrack[54] (bb-dependent),...

A rotamer

A rotamer r defines both the amino-acid used (sequence) and its conformation (χ_i).

Final stages

The "rigid backbone, discrete rotamers" approach

1. Each position i has a set of possible rotamers. Together with ρ, θ, this defines all possible sequence·conformations.
2. A combination of rotamers that optimizes F and E is sought (discrete optimization).

It's an ill-posed problem

- We adjust (s, χ) to get minimum energy on (ρ, θ).
- Even with a perfect E and exact minimization, we have no guarantee that s should fold in (ρ, θ, χ):

$$E(s, \rho, \theta, \chi) = \min_{\rho, \theta, \chi} E(s, \rho, \theta, \chi)$$

- a better backbone configuration for s may exist.

It's an ill-posed problem

- We adjust (s, χ) to get minimum energy on (ρ, θ).
- Even with a perfect E and exact minimization, we have no guarantee that s should fold in (ρ, θ, χ):

$$E(s, \rho, \theta, \chi) = \min_{\rho, \theta, \chi} E(s, \rho, \theta, \chi)$$

- a better backbone configuration for s may exist.

Extra gesticulations

1. θ, ρ, χ may be post-adjusted by quasi-Newton optimization algorithms ("minimization") using a full-atom model followed by possible loops to previous stages.
2. Forward folding: large number of (ρ, θ, χ) predicted from s using protein structure prediction. Joint plot of RMSD to target and E (funnel expected).

Why this nevertheless seems to work

Few folds for many sequences

There are less than 2,000 folds for many more sequences. Secondary structure elements and hydrophobic packing constrain the space.

Why this nevertheless seems to work

Few folds for many sequences

There are less than 2,000 folds for many more sequences. Secondary structure elements and hydrophobic packing constrain the space.

There is a variety of side-chains, with different physical and chemical properties. It seems unlikely another fundamentally different stable structure will be more stable.

Why this nevertheless seems to work

Few folds for many sequences

There are less than 2,000 folds for many more sequences. Secondary structure elements and hydrophobic packing constrain the space.

There is a variety of side-chains, with different physical and chemical properties. It seems unlikely another fundamentally different stable structure will be more stable.

Remember: we are in control!

We are allowed to make designs very predictable. This is what "forward folding" checks. Probably far from a necessary and close to a sufficient condition.

Getting closer to our target problem

The "rigid backbone, discrete rotamers" optimization problem

- we are given (ρ, θ), spatially localized rigid backbone(s)
- we are given a set D_i of usable rotamers for each position (rigid, flexible, mutable).
- we are given a function $G(X, s)$ combining E and F (assume $G = -E$ for now).

We want to identify (χ, s) that maximizes G (minimize E).

Getting closer to our target problem

The "rigid backbone, discrete rotamers" optimization problem

- we are given (ρ, θ), spatially localized rigid backbone(s)
- we are given a set D_i of usable rotamers for each position (rigid, flexible, mutable).
- we are given a function $G(X, s)$ combining E and F (assume $G = -E$ for now).

We want to identify (χ, s) that maximizes G (minimize E).

(Free) energy function

- Bonded terms: dihedrals angles (θ, χ), angles, distances.
- Non bonded: electrostatics, Van der Waals (Lennard-Jones, H-bonds,...)
- Entropy: polar solvent (non trivial).

Decomposable energy and energy matrices

$$E(s, X) = E_{\varnothing} + \sum_{i=1}^{n} E_i(i_r) + \sum_{(i,j)\in I} E_{ij}(i_r, j_s)$$

Decomposable energy and energy matrices

$$E(s, X) = E_{\varnothing} + \sum_{i=1}^{n} E_i(i_r) + \sum_{(i,j)\in I} E_{ij}(i_r, j_s)$$

- i_r: the rotamer $r \in D_i$ used at position i.
- $E_{\varnothing}$: fixed contributions (backbones, rigid side-chains). Useless for optimization.
- $E_i(i_r)$: contributions that depend just on one position (internal rotamer energies, backbone-rotamer interactions, reference energies).
- $E_{ij}(i_r, j_s)$: all contributions that result from interacting rotamers (non bonded).
- I: cutoff (non bonded interactions ignored beyond some distance threshold - interaction type dependent).

Energy matrices

A large energy ("symmetric") matrix $E[i_r, j_s]$ can be precomputed.

- Size: $n.d + \frac{n \times (n-1)}{2}.d^2$, $d = \max_i |D_i|)$
- Can be "sparse" (cutoffs).

Energy matrices

A large energy ("symmetric") matrix $E[i_r, j_s]$ can be precomputed.

- Size: $n.d + \frac{n \times (n-1)}{2}.d^2$, $d = \max_i |D_i|$)
- Can be "sparse" (cutoffs).

Computing $E(s, X)$ becomes easy.

But energy minimization looks hard

- Protein with n residues.
- 20 amino-acids for each residues: 20^n possibilities
- up to one hundred conformations per amino-acid type (depends on density of rotamers and amino-acid of course).
- a minimum of $300 - 400$ sequences-conformations for fully mutable positions (typical).

But energy minimization looks hard

- Protein with n residues.
- 20 amino-acids for each residues: 20^n possibilities
- up to one hundred conformations per amino-acid type (depends on density of rotamers and amino-acid of course).
- a minimum of $300 - 400$ sequences-conformations for fully mutable positions (typical).

Exponentials grow quickly

Especially with base 400.

But energy minimization looks hard

- Protein with n residues.
- 20 amino-acids for each residues: 20^n possibilities
- up to one hundred conformations per amino-acid type (depends on density of rotamers and amino-acid of course).
- a minimum of $300 - 400$ sequences-conformations for fully mutable positions (typical).

Exponentials grow quickly

Especially with base 400.

Does not prove it is hard (it's easy to find a shortest path in a graph even if the number of paths in it is horrendous).

Stronger hint

Assuming arbitrary energy terms

Discrete decomposable pairwise energy minimization is decision Non-deterministic Polynomial time complete.[43]

Stronger hint

Assuming arbitrary energy terms

Discrete decomposable pairwise energy minimization is decision Non-deterministic Polynomial time complete.[43]

Given e, $\exists ?(s, X)$ s.t. $E(s, X) \leq e$

Membership in NP is easy. Completeness too (MAX2SAT).

Stronger hint

Assuming arbitrary energy terms

Discrete decomposable pairwise energy minimization is decision Non-deterministic Polynomial time complete.[43]

Given e, $\exists ?(s, X)$ s.t. $E(s, X) \leq e$

Membership in NP is easy. Completeness too (MAX2SAT).

Ubiquity of stochastic local search/heuristics

- Bio-inspired: Genetic Algorithms (EGAD[44]).
- Physics-inspired: Monte Carlo biased for optimization[47]

Stronger hint

Assuming arbitrary energy terms

Discrete decomposable pairwise energy minimization is decision Non-deterministic Polynomial time complete.[43]

Given e, $\exists ?(s, X)$ s.t. $E(s, X) \leq e$

Membership in NP is easy. Completeness too (MAX2SAT).

Ubiquity of stochastic local search/heuristics

- Bio-inspired: Genetic Algorithms (EGAD[44]).
- Physics-inspired: Monte Carlo biased for optimization[47]

Does this mean there is no hope we can solve the problem exactly?

NP-complete $\equiv$ cannot be solved ?

Even if P $\neq$ NP, NP-completeness only implies there is an infinite family of energy minimization problems of arbitrary sizes that can only be solved in asymptotic exponential time.

Speaks of worst problems and does not say if this is 1.0000001^n.

NP-complete $\equiv$ cannot be solved ?

Even if P $\neq$ NP, NP-completeness only implies there is an infinite family of energy minimization problems of arbitrary sizes that can only be solved in asymptotic exponential time.

Speaks of worst problems and does not say if this is 1.0000001^n.

NP is the "New P" (Moshe Vardi, president of the ACM)

1. SAT solvers solve SAT instances with $> 10^6$ variables.
2. similar progress in other areas: constraint programming, integer linear programming, graphical model solving...
3. `toulbar2` can solve (find and prove minimum of) problems with $2^{1,000,000}$ states.

E is approximate. I do not need exact solving

Guaranteed solving: the good

- gives a real unbiased access to what minimum energy means (learning energy parameters, understanding misbehaviors)
- can be much faster than Monte Carlo (knows when to stop)
- may be asked for weaker guarantees (distance to optimum)
- can provide gap-less enumeration
- you exactly know what you get (no need to rerun).
- Monte Carlo trades time for quality, but you don't know the exchange rate, may get really stuck (ergodicity)

E is approximate. I do not need exact solving

Guaranteed solving: the good

- gives a real unbiased access to what minimum energy means (learning energy parameters, understanding misbehaviors)
- can be much faster than Monte Carlo (knows when to stop)
- may be asked for weaker guarantees (distance to optimum)
- can provide gap-less enumeration
- you exactly know what you get (no need to rerun).
- Monte Carlo trades time for quality, but you don't know the exchange rate, may get really stuck (ergodicity)

and the bad...

- has the possibility of being utterly slow (is 1 MY ok?)

Full redesign of 107 short proteins

Rosetta (Dunbrack, Talaris14), Fixbb MC protocol vs. `toulbar2`

Full redesign of 107 short proteins

Rosetta (Dunbrack, Talaris14), Fixbb MC protocol vs. `toulbar2`

Why full redesigns

1. Challenging
2. Used on $\beta 1$ domain of protein G to tune energy function parameters[2].

Full redesign of 107 short proteins

Rosetta (Dunbrack, Talaris14), Fixbb MC protocol vs. `toulbar2`

Why full redesigns

1. Challenging
2. Used on $\beta 1$ domain of protein G to tune energy function parameters[2].

The designs

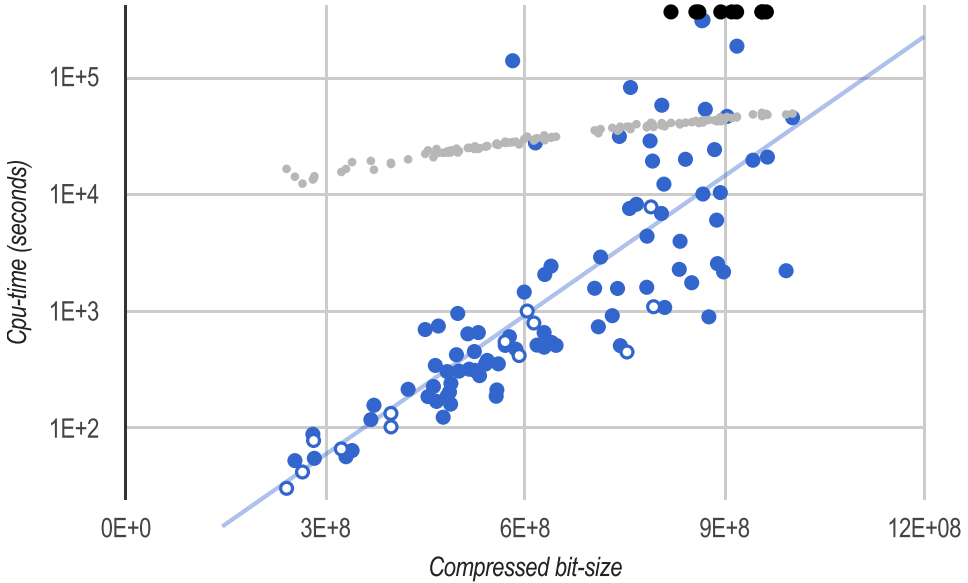
1. Structures extracted from the PDB (September 2014)
2. Length from 50 to 100 AA
3. Resolution better than 2 Å
4. Only representative at $< 30\%$ identity

Looking for the Global Minimum Energy Conformation

How

1. Intel Xeon E5-2690 2.9 GHz (Q1-2012 CPU)
2. `toulbar2`: 100 hours limit.

Looking for the GMEC



Looking for the GMEC and around

toulbar2 (CFN)

1. 98 problems solved to optimality
2. Largest problem solved: 10^{234}, energy matrix of 1.7 GB
3. Smallest unsolved: 10^{206}.
4. exact SCP with Talaris14 feasible even on big proteins.

Looking for the GMEC and around

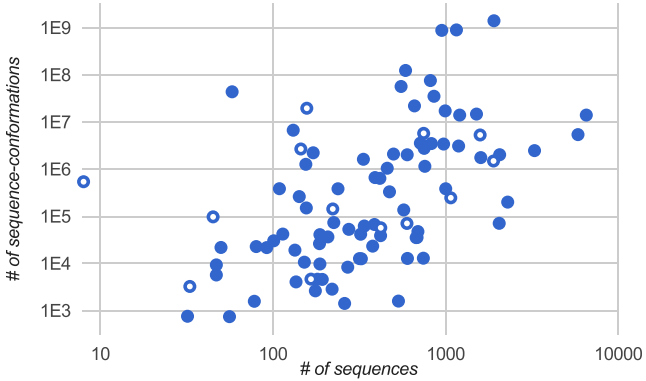
toulbar2 (CFN)

1. 98 problems solved to optimality
2. Largest problem solved: 10^{234}, energy matrix of 1.7 GB
3. Smallest unsolved: 10^{206}.
4. exact SCP with Talaris14 feasible even on big proteins.

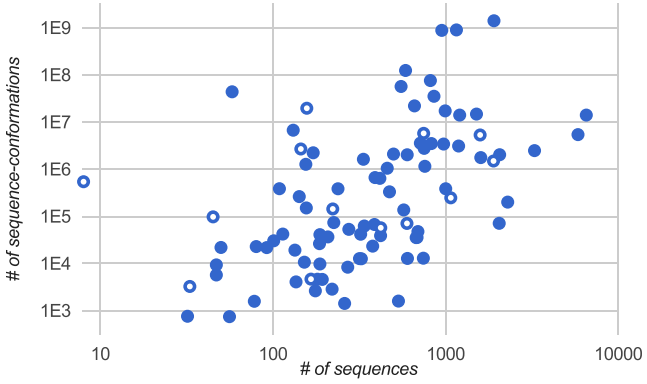
All sequence·conformations in 0.2 Rosetta unit (100h limit)

1. Gap-less list of sequence·conformation on 92/98 designs
2. Very fast sampling, huge spaces (up to 1.42 10^9)

Exploring Sequence·conformations around the GMEC



Exploring Sequence·conformations around the GMEC



"SCP branching" algorithm

- For each sequence, only finds a good enough conformation
- Exhaustively enumerates sequences in larger energy gaps

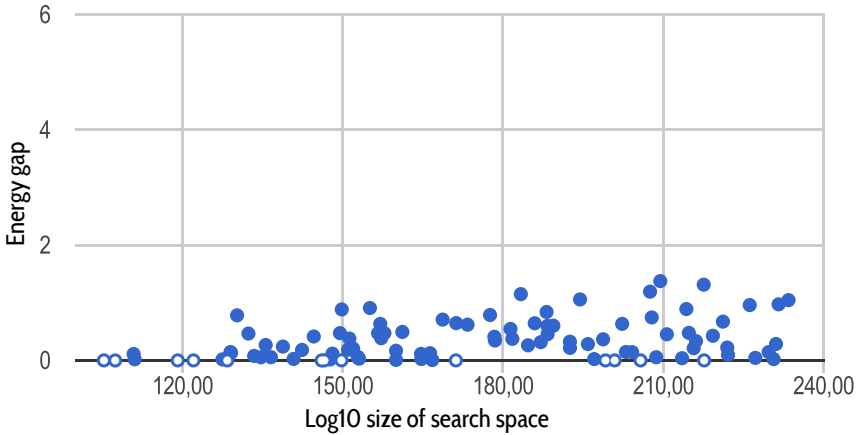
Rosetta FixBB protocol

Rosetta/fixbb

1. Best of 1000 runs of fixbb Rosetta protocol
2. Rosetta fixbb found the GMEC on 13 of these problems
3. These 13 problems took 90 hours for fixbb.
4. `toulbar2` solved them to optimality in 36 hours.

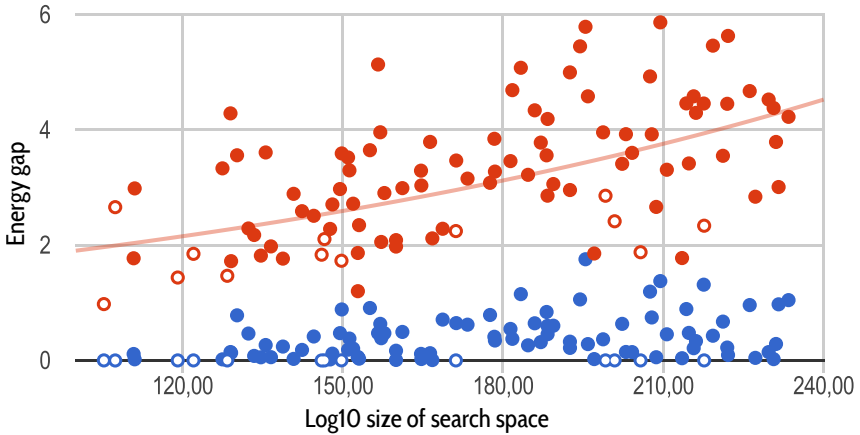
Distance to optimum as a function of space size

- Blue: best over 1 000 runs



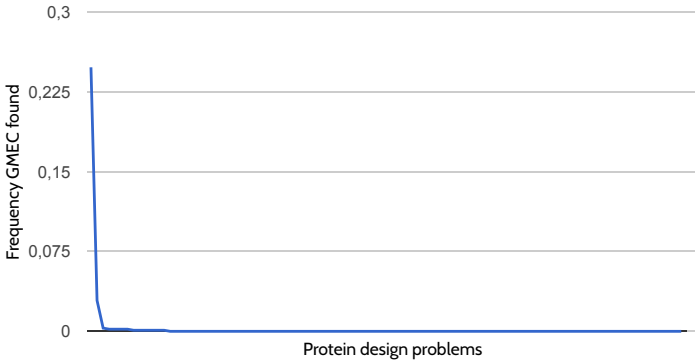
Distance to optimum as a function of space size

- Blue: best over 1 000 runs
- Red: average over 1 000 runs.



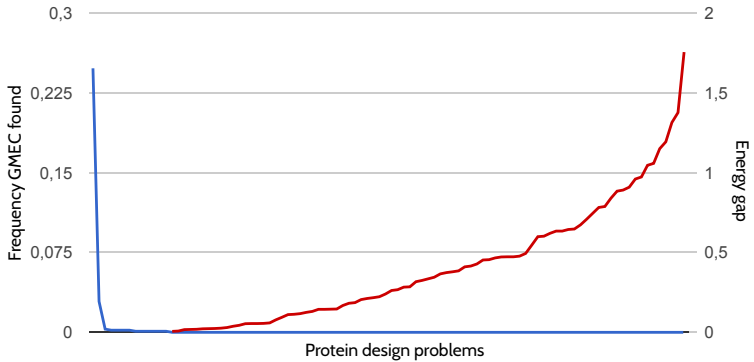
Reliability, distance to optimum and size

- Blue: probability of finding the GMEC (sorted)



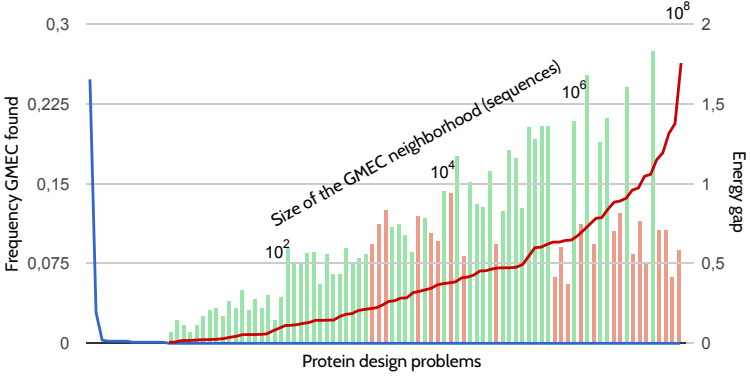
Reliability, distance to optimum and size

- Blue: probability of finding the GMEC (sorted)
- Red: energy gap to GMEC (sorted)



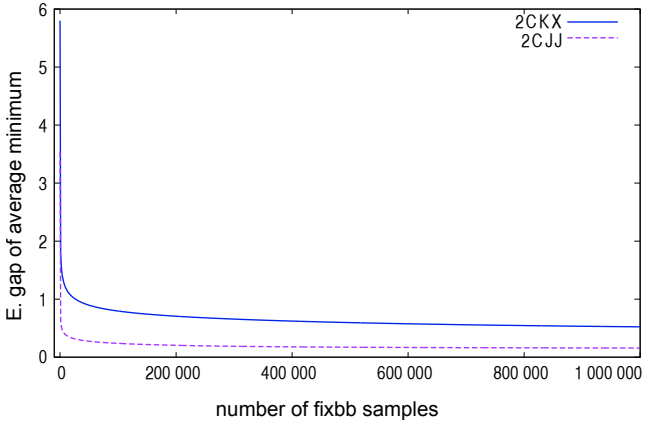
Reliability, distance to optimum and size

- Blue: probability of finding the GMEC (sorted)
- Red: energy gap to GMEC (sorted)
- Histogram: # of unique sequences between GMEC and fixbb best sequence (red: lower bound)



Getting closer to the GMEC with SA

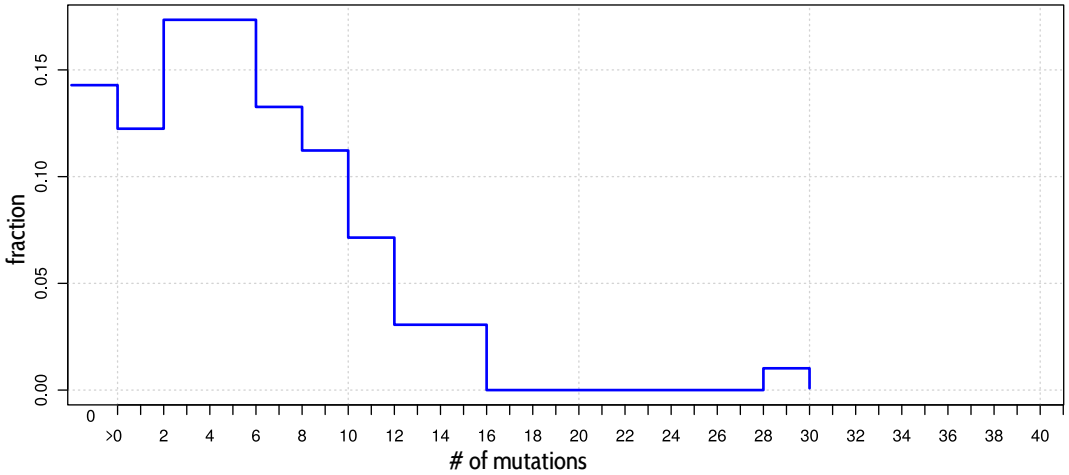
- Worst mean/median energy gap: 2CJJ and 2CKX
- 1 million runs of fixbb, 2 years of cpu-time each
- can estimate the expected gap as a function of time



What about sequences: Hamming dist. to GMEC

Best energy fixbb vs. GMEC

- 2.4% core, 7% boundary, 10% surface.



Distance to native as we get closer to the GMEC

Native sequence

- sort of reference (a protein's life is not just stability).
- used to tune energy[2,33]

Distance to native as we get closer to the GMEC

Native sequence

- sort of reference (a protein's life is not just stability).
- used to tune energy[2,33]

Type	native		fixbb best		GMEC
Hydrophobic	2,585	↘	2,440	↘	2,401
Charged	1,795	↗	1,996	↗	2,097
Polar	1,817	↘	1,730	↘	1,662
Aromatic	585	↗	616	↗	622

Cysteines in disulfide bridges: not counted.

Possible lessons

Monte Carlo sampling

- Fixbb becomes quickly unable to reach lowest energy areas
- Energy gap increases steadily with search space size
- What about unbiased (MC)MC sampling ?

Possible lessons

Monte Carlo sampling

- Fixbb becomes quickly unable to reach lowest energy areas
- Energy gap increases steadily with search space size
- What about unbiased (MC)MC sampling ?

Why guarantees are good?

- GMEC not crucial, but an *upper bound on error*?
- Guaranteed optima have different composition
- Talaris favorable for guaranteed optimization (but exponential)
- Exhaustive enumeration can be very fast (exponential output)

Some exact NP-complete optimization frameworks

- 0/1 Linear programming: optimize a linear function under linear constraints (CPLEX and Gurobi free for academics).
- Quadratic Programming: optimize a quadratic criteria under linear constraints (SDP based, BiqMac, open source).
- PWMaxSat: maximize the weighted # of satisfied clauses under hard clauses (MaxHS, bincd, akmaxsat,...).
- Graphical models: minimize a decomposable function

Advanced optimization methods for CPD

Some exact NP-complete optimization frameworks

- 0/1 Linear programming: optimize a linear function under linear constraints (CPLEX and Gurobi free for academics).
- Quadratic Programming: optimize a quadratic criteria under linear constraints (SDP based, BiqMac, open source).
- PWMaxSat: maximize the weighted # of satisfied clauses under hard clauses (MaxHS, bincd, akmaxsat,...).
- Graphical models: minimize a decomposable function

Focus on Graphical Model optimization

- All NP-complete: polytime transform from any to the others
- ILP used eg. in CLASSY[38]
- Graphical model optimization: direct, fastest exact method

What is a Graphical Model ?

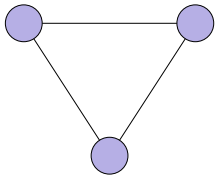
1. A set of variables, each with a domain
2. We define a joint function on all variables
3. By combining ($\bigotimes$) functions involving few variables

What is a Graphical Model ?

1. A set of variables, each with a domain
2. We define a joint function on all variables
3. By combining ($\bigotimes$) functions involving few variables

Why "graphical" ?

- One vertex per variable
- One edge if two variables participate together in a function
- Can describe a function on many variables concisely
- Hard to manipulate (NP-hard queries)

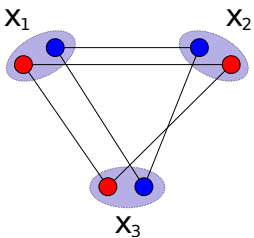


What is a Graphical Model ?

1. A set of variables, each with a domain
2. We define a joint function on all variables
3. By combining ($\bigotimes$) functions involving few variables

Why "graphical" ?

- One vertex per variable
- One edge if two variables participate together in a function
- Can describe a function on many variables concisely
- Hard to manipulate (NP-hard queries)



The many faces of graphical models

Which $\otimes$, which query ($\oplus$), which cost domain?

1. $(\min, \wedge)$, $\mathbb{B}$	Constraint Networks, CSP/SAT
2. $(\min, +)$, $\mathbb{N}^+$, $\mathbb{Q}^+$	Cost Function Networks, WCSP
3. $(\min, +)$, $\mathbb{R}$	Minimum energy
4. $(\max, \times)$, $\mathbb{R}^+$	Maximum probability
5. $(+, \times)$, $\mathbb{R}^+$	Weighted counting, Z

$(\oplus, \otimes)$ should define a semi-ring.

The many faces of graphical models

Which $\otimes$, which query ($\oplus$), which cost domain?

1. $(\min, \wedge)$, $\mathbb{B}$	Constraint Networks, CSP/SAT
2. $(\min, +)$, $\mathbb{N}^+$, $\mathbb{Q}^+$	Cost Function Networks, WCSP
3. $(\min, +)$, $\mathbb{R}$	Minimum energy
4. $(\max, \times)$, $\mathbb{R}^+$	Maximum probability
5. $(+, \times)$, $\mathbb{R}^+$	Weighted counting, Z

$(\oplus, \otimes)$ should define a semi-ring.

- Different algebras. We will stick to $(\min, +)$ and $(\sum, \prod)$
- Often closely related/equivalent algorithms[45,52,55].
- All problems NP-hard (or worse !)

A CFN (X, D, C, k)

1. Set $X = \{1, 2, \ldots, n\}$ of variables. i with finite domain $D_i \in D$.
2. Set C of cost functions c_S, each depending on some variables $S \subset X$, $c_S : D^S \to \{0, \ldots, k\}$ (k finite or not)
3. Cost combined by (bounded) addition[8]

$$C(x) = \sum_{c_S \in C} c_S(x[S]) \qquad c_\varnothing : \text{lower bound}$$

A CFN (X, D, C, k)

1. Set $X = \{1, 2, \ldots, n\}$ of variables. i with finite domain $D_i \in D$.
2. Set C of cost functions c_S, each depending on some variables $S \subset X$, $c_S : D^S \to \{0, \ldots, k\}$ (k finite or not)
3. Cost combined by (bounded) addition[8]

$$C(x) = \sum_{c_S \in C} c_S(x[S]) \qquad c_\varnothing : \text{lower bound}$$

The Weighted Constraint Satisfaction Problem

Find an assignment x of all variables s.t. $C(x) = \min_{y \in D^X} C(y)$.

A CFN (X, D, C, k)

1. Set $X = \{1, 2, \ldots, n\}$ of variables. i with finite domain $D_i \in D$.
2. Set C of cost functions c_S, each depending on some variables $S \subset X$, $c_S : D^S \to \{0, \ldots, k\}$ (k finite or not)
3. Cost combined by (bounded) addition[8]

$$C(x) = \sum_{c_S \in C} c_S(x[S]) \qquad c_\varnothing : \text{lower bound}$$

The Weighted Constraint Satisfaction Problem

Find an assignment x of all variables s.t. $C(x) = \min_{y \in D^X} C(y)$.

If $k = 1$, this is the "Constraint Satisfaction Problem" (CSP).

You Shift, Scale, Round

- Variables: one per residue, domain of available rotamers for the residue.
- Cost functions: each energy function $E_\varnothing$, E_i, E_{ij} defines a cost function with a precision:

$$c_S(x) = \left\lceil (E_S(x) - \min_{y \in D^S} E_S(y)) \times 10^{precision} \right\rceil$$

You Shift, Scale, Round

- Variables: one per residue, domain of available rotamers for the residue.
- Cost functions: each energy function $E_\varnothing$, E_i, E_{ij} defines a cost function with a precision:

$$c_S(x) = \left\lceil (E_S(x) - \min_{y \in D^S} E_S(y)) \times 10^{precision} \right\rceil$$

Just fixed decimal point representation. Adjustable maximum error.

You Shift, Scale, Round

- Variables: one per residue, domain of available rotamers for the residue.
- Cost functions: each energy function $E_\varnothing$, E_i, E_{ij} defines a cost function with a precision:

$$c_S(x) = \left\lceil (E_S(x) - \min_{y \in D^S} E_S(y)) \times 10^{precision} \right\rceil$$

Just fixed decimal point representation. Adjustable maximum error.

In the rest of the talk, I (often) confound energies E_S and costs c_S.

ILP, QP, Graphical model, all NP-complete

All equivalent

Let's look (or skip) a few slides to check this.

Pairwise CFN as 01LP (infinite k, finite costs)

The local polytope [27, 53, 64]

$$\text{Minimize} \sum_{i,a} c_i(a) \cdot x_{ia} + \sum_{\substack{c_{ij} \in C \\ a \in D^i, b \in D^j}} c_{ij}(a,b) \cdot y_{iajb} \quad \text{subject to}$$

$$\sum_{a \in D^i} x_{ia} = 1 \qquad \forall i \in \{1, \ldots, n\}$$

$$\sum_{b \in D^j} y_{iajb} = x_{ia} \qquad \forall c_{ij} \in C, \forall a \in D^i$$

$$\sum_{a \in D^i} y_{iajb} = x_{jb} \qquad \forall c_{ij} \in C, \forall b \in D^j$$

$$x_{ia} \in \{0,1\} \qquad \forall i \in \{1, \ldots, n\}$$

$nd + e.d^r$ variables. $n + 2ed$ contraintes.

Binary CFN as 01QP (infinite k, finite costs)

Only nd variables

$$\min \sum_{i,a} c_i(a).x_{ia} + \sum_{\substack{c_{ij} \in C \\ a \in D^i, b \in D^j}} c_{ij}(a,b) \cdot x_{ia} \cdot x_{jb} \quad \text{subject to}$$

$$\sum_{a} x_{ia} = 1 \quad (\forall i \in \{1, \ldots, n\})$$

Fastest exact method?

Benchmark

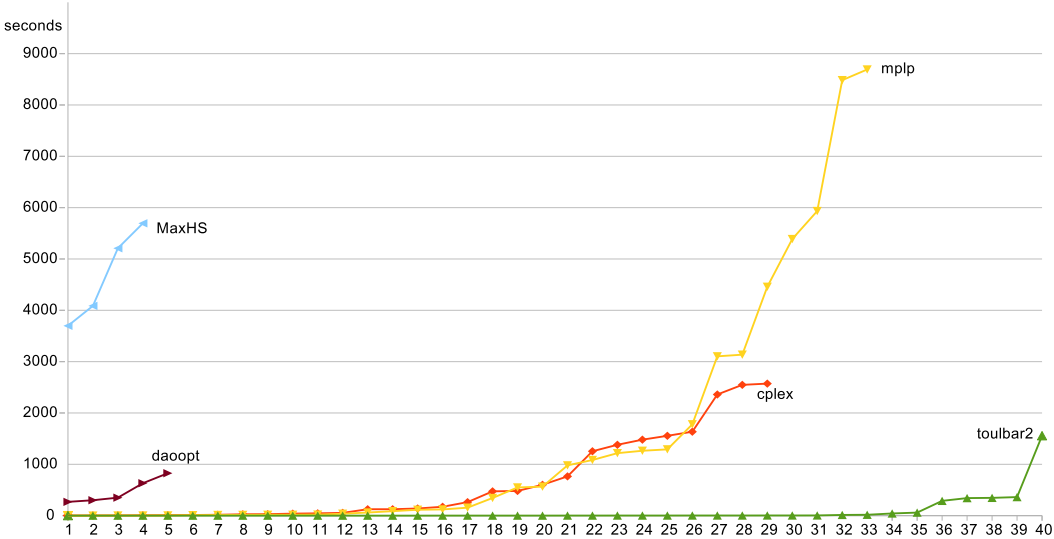
- 47 high quality backbones extracted from the protein design literature.
- AMBER energy matrices computed using OSPREY (Penultimate rotamer library).
- used to compare exact optimization methods for ILP, QP, PWMaxSAT, OSPREY, Graphical model solvers.

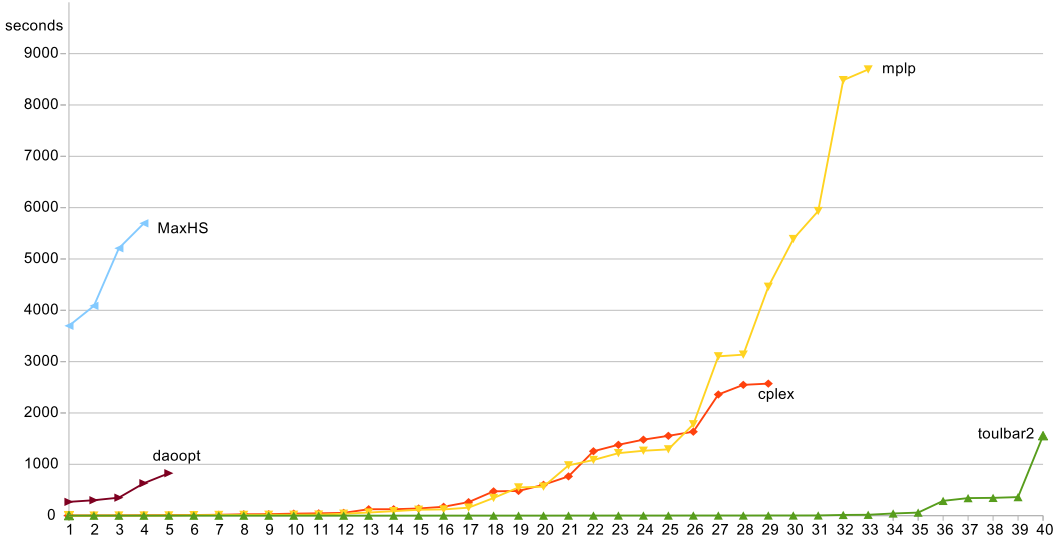
Fastest exact method?

Benchmark

- 47 high quality backbones extracted from the protein design literature.
- AMBER energy matrices computed using OSPREY (Penultimate rotamer library).
- used to compare exact optimization methods for ILP, QP, PWMaxSAT, OSPREY, Graphical model solvers.

All encodings (polytime transformations) described in David Allouche et al. "Computational protein design as an optimization problem". In: *Artificial Intelligence* 212 (2014), pp. 59–79.


seconds
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
mplp
MaxHS
cplex
toulbar2
daoopt



QP and other PWMaxSAT solvers do not solve any instance.

Dead End Elimination & Neighborhood Substitutability

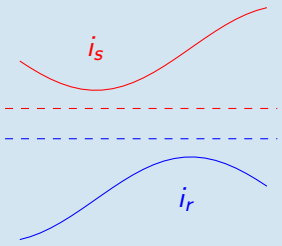
DEE - Nature (Desmet, de Maeyer et al., 1992) - 791 cites

- two rotamers i_r, i_s of the residue i
- if the best arrangement of neighbors for i_s has a larger energy than the worst arrangement of neighbors for i_r, remove i_s.

$$E(i_r) + \sum_{j \neq i} \max_t E(i_r, j_t) < E(i_s) + \sum_{j \neq i} \min_t E(i_s, j_t)$$

$$E_i(i_s) + \sum_{j=1}^{n} \min_t E_{ij}(i_s, j_t)$$

$$E_i(i_r) + \sum_{j=1}^{n} \max_t E_{ij}(i_r, j_t)$$



Dead End Elimination & Neighborhood Substitutability

DEE - Nature (Desmet, de Maeyer et al., 1992) - 791 cites

- two rotamers i_r, i_s of the residue i
- if the best arrangement of neighbors for i_s has a larger energy than the worst arrangement of neighbors for i_r, remove i_s.

$$E(i_r) + \sum_{j \neq i} \max_t E(i_r, j_t) < E(i_s) + \sum_{j \neq i} \min_t E(i_s, j_t)$$

Neigh. Substitutability - AAAI, (Gene Freuder, 1991) - 290 cites

- two values i_r, i_s of the variable i
- if the best arrangement of neighbors of i_r is false whenever the worst arrangement of the neighbors of i_s is false, remove i_r.

$$E(i_r) \bigwedge_{j \neq i} \bigwedge_t E(i_r, j_t) \leq E(i_s) \bigwedge_{j \neq i} \bigvee_t E(i_s, j_t)$$

Dead End Elimination

Sufficient condition for suboptimality

- Also known as a dominance rule, persistency (Boolean case[5,20]), substitutability,[13] partial optimality,[57]...
- Polynomial space/time complexity
- More or less effective sufficient conditions exist on values,[1,17,34] on pairs and beyond.[35]

Dead End Elimination

Sufficient condition for suboptimality

- Also known as a dominance rule, persistency (Boolean case[5,20]), substitutability,[13] partial optimality,[57]...
- Polynomial space/time complexity
- More or less effective sufficient conditions exist on values,[1,17,34] on pairs and beyond.[35]

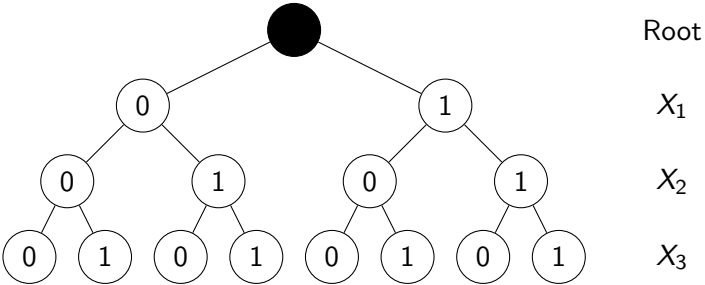
Not a panacea

- polynomial time, so cannot solve the fixed backbone protein design problem in all cases.
- may remove "close to optimal" solutions.
- usually solves only a fraction of nowadays "small problems".

Needs to be combined with tree search

Which trees ?

- the root node is the graphical model to solve (or a preprocessed one).
- the sons of a node are obtained by splitting the domain (how) of a chosen (how) variable in at least two sets (value/variable heuristics).
- the leaves are totally assigned graphical models.



Here: used only for search in trees of bounded depth.

A^*: exact shortest path finding in implicit graphs

Here: used only for search in trees of bounded depth.

open: list of nodes yet to explore (only the root initially).
$h(n)$: lower bound on the best leaf cost under n.

1. Extract $n = \arg\min_{open} h(n)$ from *open* (admissible heuristics)
2. If n is a leaf, we have an optimal solution
3. Else, we insert all sons of n in *open* and loop to 1

A^*: exact shortest path finding in implicit graphs

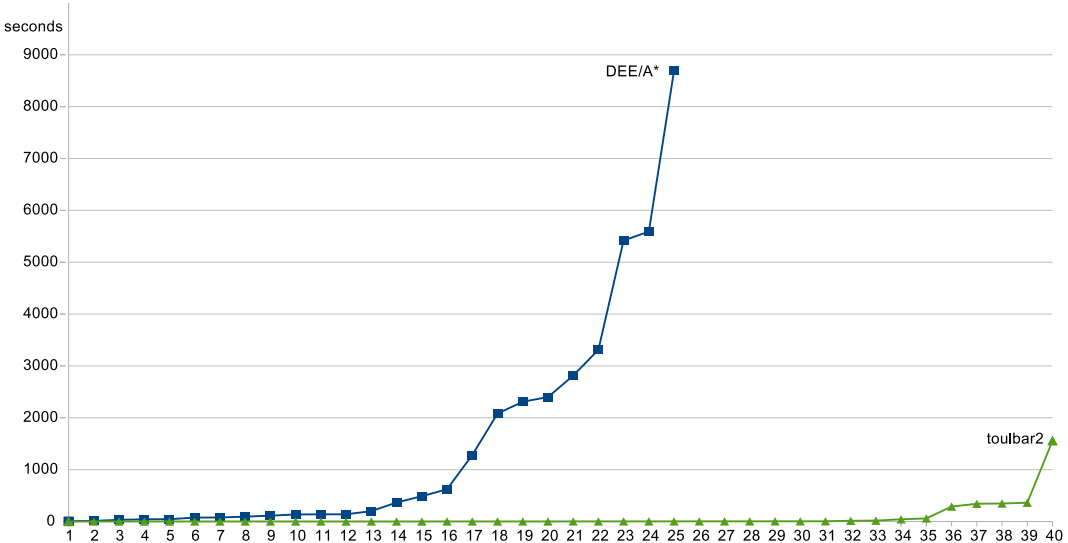
Here: used only for search in trees of bounded depth.

open: list of nodes yet to explore (only the root initially).
$h(n)$: lower bound on the best leaf cost under n.

1. Extract $n = \arg\min_{open} h(n)$ from *open* (admissible heuristics)
2. If n is a leaf, we have an optimal solution
3. Else, we insert all sons of n in *open* and loop to 1

- worst case exponential time and space
- $h(n)$ quality is crucial (tight/fast)
- anytime lower bound (not upper bound)
- produces solutions with increasing energies

DEE/A* (OSPREY) vs. CFN (toulbar2)



Should we forget DEE/A*?

No, DEE/A* powers OSPREY

- Open source successful Java tool with AMBER force field.[14]
- Exact methods for continuous rotamers,[18] flexible backbone or both[19] (harder to solve).
- Recent versions use `toulbar2` as a subroutine.

Should we forget DEE/A*?

No, DEE/A* powers OSPREY

- Open source successful Java tool with AMBER force field.[14]
- Exact methods for continuous rotamers,[18] flexible backbone or both[19] (harder to solve).
- Recent versions use `toulbar2` as a subroutine.

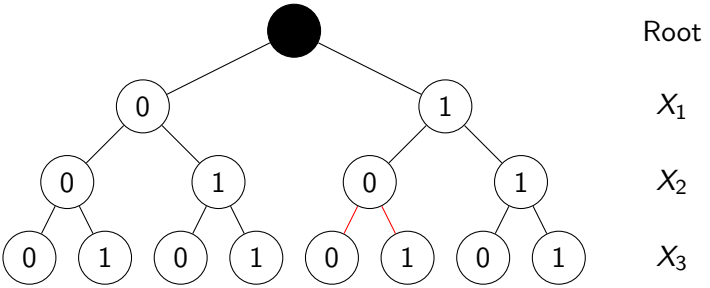
How: 20 years of open source research/evaluation

- poly/any space tree search exploration (DFS, HBFS)
- strong/fast bounds (preprocessing and during search), crucial
- variable/value ordering (activity/bound guided): crucial
- variable elimination (preprocessing and during search)
- exploiting conditional independencies (tree decompositions)
- DEE (preprocessing and during search): minor effects

Depth First Branch and Bound (DFS)

Dive and backtrack

- the tree is "ordered" by the branching heuristics
- ordered search: a current position suffices (linear space)
- quick first solution, k set to the energy of the best known
- if $h(n) \geq k$, no better solution below: backtrack



Enumerating sequences/conformation

Within a δ energy threshold of the GMEC

1. Assume you know the GMEC energy E^*
2. Set k to $E^* + \delta$ and never update k during search
3. The energy E of each attained leaf $E^* \leq E < E^* + \delta$

Enumerating sequences/conformation

Within a δ energy threshold of the GMEC

1. Assume you know the GMEC energy E^*
2. Set k to $E^* + \delta$ and never update k during search
3. The energy E of each attained leaf $E^* \leq E < E^* + \delta$

- Empirically fast sequence·conformation enumeration
- Estimation of side-chain conformational entropy

Enumerating sequences, not conformations

SCP Branching[60]

1. Same as above plus...
2. Branch first on variables with 2 or more AAs in their domain
3. Split on AA identity if possible
4. When a leaf is reached, backtrack to an AA split node.

Enumerating sequences, not conformations

SCP Branching[60]

1. Same as above plus...
2. Branch first on variables with 2 or more AAs in their domain
3. Split on AA identity if possible
4. When a leaf is reached, backtrack to an AA split node.

- Exponential speedups.
- Enumerations of sequence on larger energy gaps
- Sequence libraries!

Hybrid Best First Search

A^* mixed with DFS

- extract a favorite node n from *open*
- explore the sub-tree below n with DFS (k updated)
- stop when a maximum # of nodes have been explored
- put all "dangling branches" as new nodes in *open*
- filter *open* (remove n if $h(n) \geq k$)

Hybrid Best First Search

A^* mixed with DFS

- extract a favorite node n from *open*
- explore the sub-tree below n with DFS (k updated)
- stop when a maximum # of nodes have been explored
- put all "dangling branches" as new nodes in *open*
- filter *open* (remove n if $h(n) \geq k$)

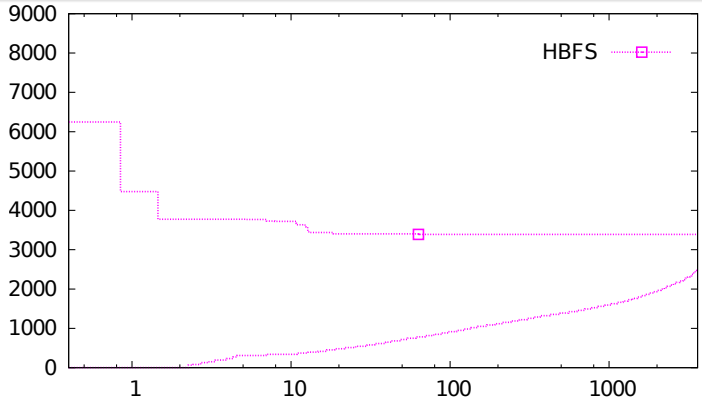
Benefits

- anyspace: if *open* gets too large, just do more DFS
- anytime solutions: we get increasingly good solutions
- anytime lower bound: $\min_{n \in open} h(n)$
- unordered search (a path to diversity?).

Avenues for time/quality trade-offs

Anytime profiles - HBFS

Increasingly tighter gap: (a solution/upper bound, a lower bound).
`-timer=b`: stop after b seconds. Trades time for quality with a bound on what you risk loosing.



Bounds and Equivalence Preserving Transformations

Naive lower bound $h(n)$

Costs are non negative thus $c_{\varnothing}$ is a lower bound of the optimum.

Bounds and Equivalence Preserving Transformations

Naive lower bound $h(n)$

Costs are non negative thus $c_\varnothing$ is a lower bound of the optimum.

Main idea

Transform a CFN into an equivalent CFN with a larger $c_\varnothing$
Equivalent: same joint cost $C(x)$ for any assignment $x \in D^X$

Bounds and Equivalence Preserving Transformations

Naive lower bound $h(n)$

Costs are non negative thus $c_\varnothing$ is a lower bound of the optimum.

Main idea

Transform a CFN into an equivalent CFN with a larger $c_\varnothing$
Equivalent: same joint cost $C(x)$ for any assignment $x \in D^X$

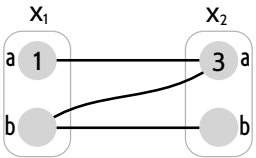
Equivalent Preserving Transformations (EPTs)

Shift costs between different scopes: 2-bodies terms to 1-body terms, 2-bodies terms to constant term, or the converse.

- never change the joint cost distribution $C(\cdot)$
- never create negative costs
- incrementality! Ideal for DFS/HBFS.

Tightening $c_\varnothing$ by EPTs

Assume that initially $c_\varnothing = 0, k = 4$



Tightening $c_\varnothing$ by EPTs

Assume that initially $c_\varnothing = 0, k = 4$

Shift 1 to right *a*

Tightening $c_\varnothing$ by EPTs

Assume that initially $c_\varnothing = 0, k = 4$

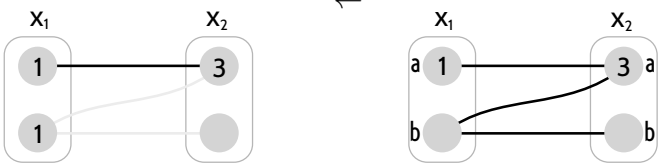
$\leftarrow$

Shift 1 from right a

Tightening $c_\varnothing$ by EPTs

Assume that initially $c_\varnothing = 0, k = 4$

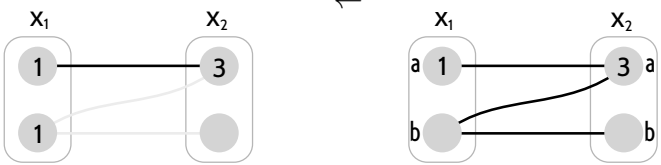
Shift 1 to left b



Tightening $c_\varnothing$ by EPTs

Assume that initially $c_\varnothing = 0, k = 4$

Shift 1 to left b

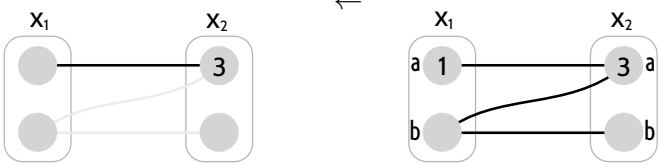


$\Downarrow$ Shift 1 from x_1 to $c_\varnothing$

Tightening $c_\varnothing$ by EPTs

Assume that initially $c_\varnothing = 0, k = 4$

Shift 1 to left b



$\Downarrow$

$c_\varnothing = 1$

Shift 1 from x_1 to $c_\varnothing$

Tightening $c_\varnothing$ by EPTs

Assume that initially $c_\varnothing = 0, k = 4$

Shift 1 to left b

$\leftarrow$

$\Downarrow$

$c_\varnothing = 1$

Shift 1 from x_1 to $c_\varnothing$

Preserves global energy below k.

Tightening $c_\varnothing$ by EPTs

Assume that initially $c_\varnothing = 0, k = 4$

Shift 1 to left b

$\leftarrow$

$\Downarrow$

$c_\varnothing = 1$

Shift 1 from x_1 to $c_\varnothing$

Preserves global energy below k.

Dead End Elimination does not

Local consistencies

Properties of CFN such that a well defined set of obvious local deductions have been performed leading to a fixpoint (closure).

Properties of CFN such that a well defined set of obvious local deductions have been performed leading to a fixpoint (closure).

Node consistency (NC)

For any variable i, there exists i_r s.t. $c_i(i_r) = 0$ and no value i_s such that $c_\varnothing + c_i(i_s) \geq k$.

Properties of CFN such that a well defined set of obvious local deductions have been performed leading to a fixpoint (closure).

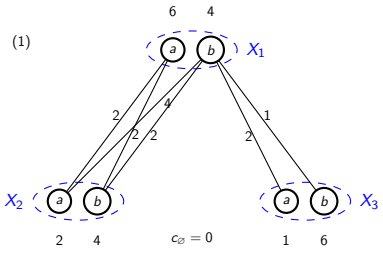
Node consistency (NC)

For any variable i, there exists i_r s.t. $c_i(i_r) = 0$ and no value i_s such that $c_\varnothing + c_i(i_s) \geq k$.

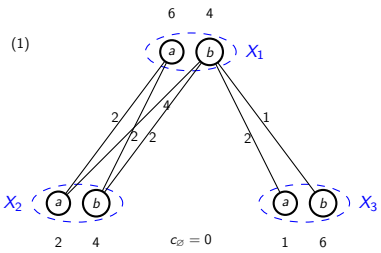
Arc consistency (AC*)

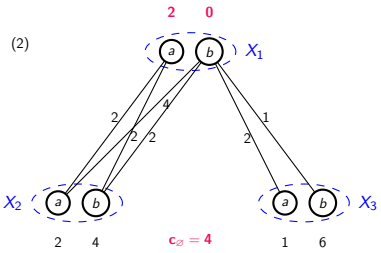
NC + for any variable i and value i_r and cost function c_{ij}, there exists $j_s \in D_j$ s.t. $c_{ij}(i_r, j_s) = 0$.

EPTs for Node consistency

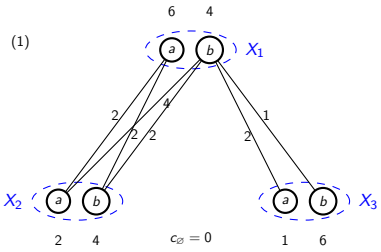


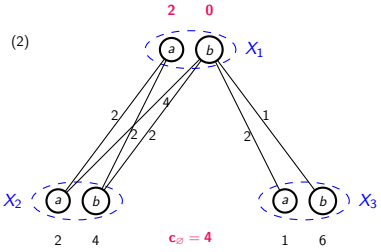
EPTs for Node consistency

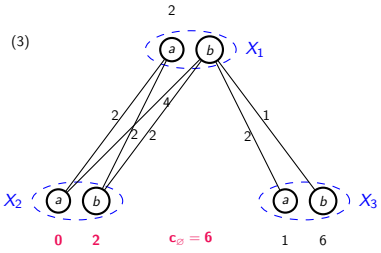




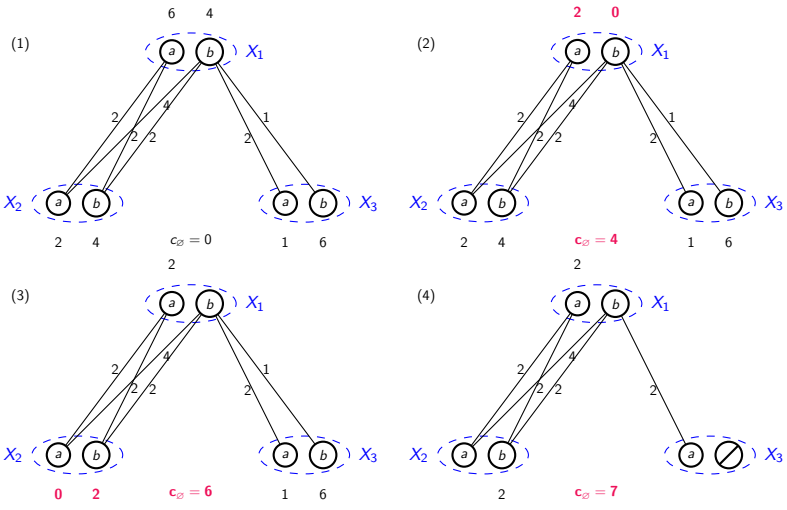
EPTs for Node consistency



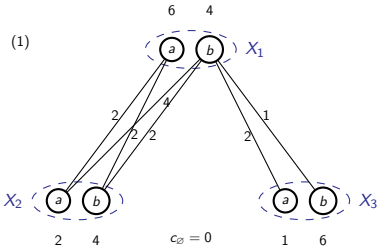




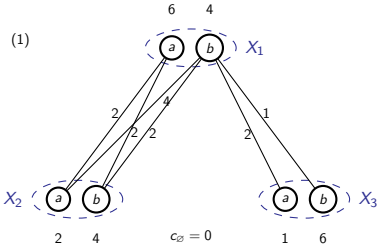
EPTs for Node consistency

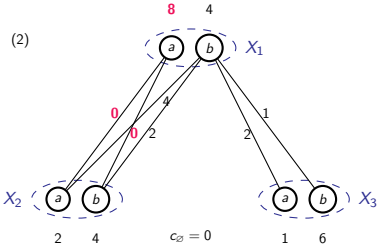


EPTs for Arc consistency

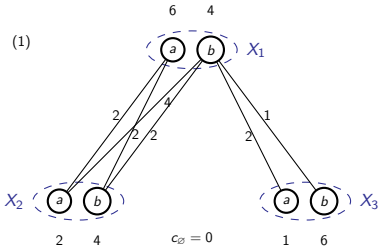


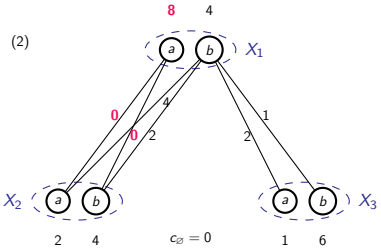
EPTs for Arc consistency

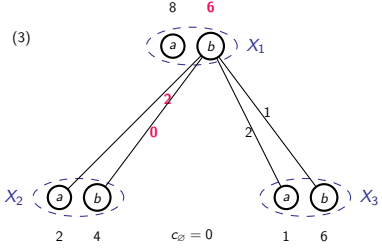




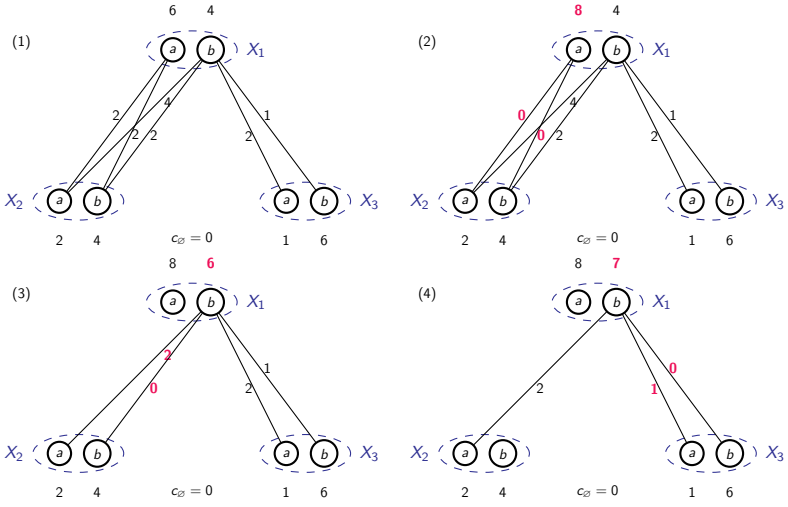
EPTs for Arc consistency



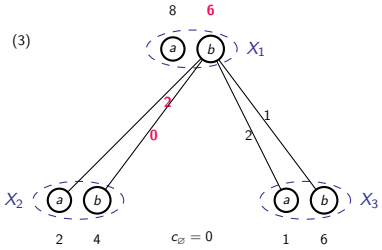


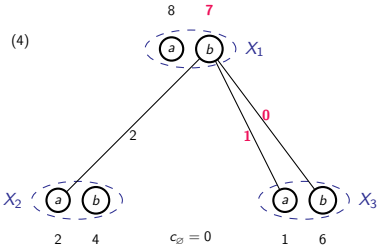


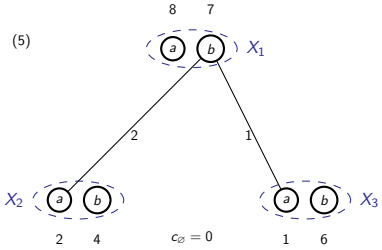
EPTs for Arc consistency



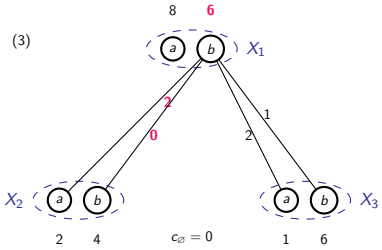
EPTs for Arc consistency

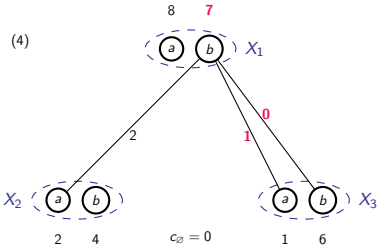


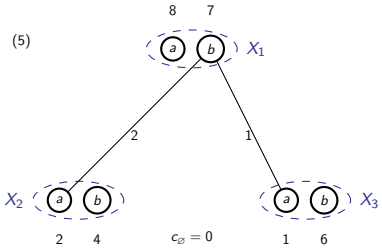


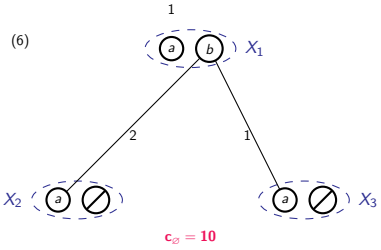


EPTs for Arc consistency



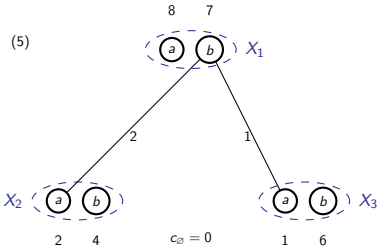






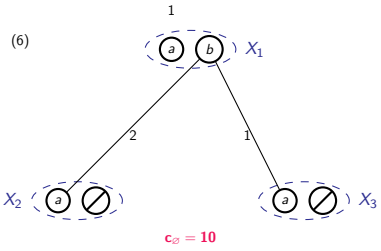
EPTs for Arc consistency

(5)



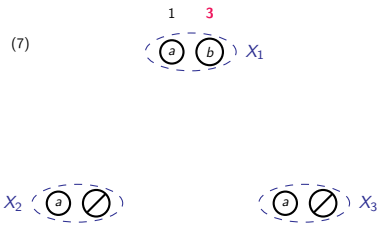
$c_\varnothing = 0$

(6)



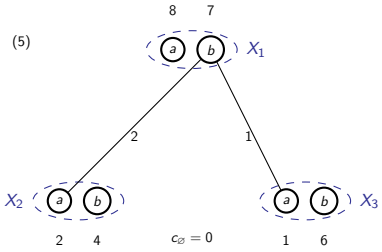
$\mathbf{c_\varnothing = 10}$

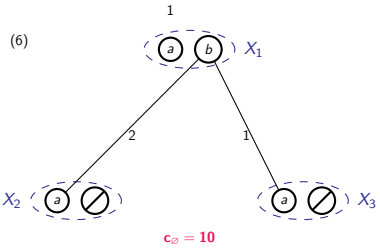
(7)

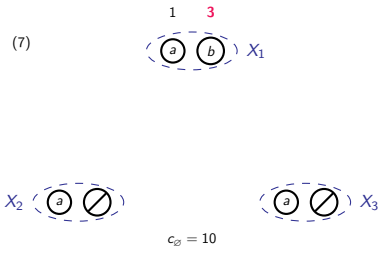


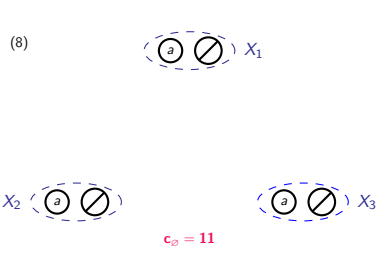
$c_\varnothing = 10$

EPTs for Arc consistency









Increasingly strong local consistencies

1. 2000: Arc Consistency[51]
2. 2003: (Full) Directional Arc Consistency[31]
3. 2005: Full Existential Directional Arc Consistency[32]
4. 2008: Virtual Arc Consistency[9,10,25,64]
5. also in image processing. Look for TRWS,[25,64] OpenGM.[23]

Increasingly strong local consistencies

1. 2000: Arc Consistency[51]
2. 2003: (Full) Directional Arc Consistency[31]
3. 2005: Full Existential Directional Arc Consistency[32]
4. 2008: Virtual Arc Consistency[9,10,25,64]
5. also in image processing. Look for TRWS,[25,64] OpenGM.[23]

Local consistency enforcing

- only remove solutions that have energy $\geq k$ (adjustable).
- does not change the problem (below k)
- gives an incremental lower bound

Bool(P) [9]

Given a CFN $P = (X, D, C, k)$, Bool(P) is the CSP $(X, D, C - \{c_\varnothing\}, 1)$.

Bool(P) forbids all positive cost assignments, ignoring $c_\varnothing$.

Bool(P) [9]

Given a CFN $P = (X, D, C, k)$, Bool(P) is the CSP $(X, D, C - \{c_\varnothing\}, 1)$.

Bool(P) forbids all positive cost assignments, ignoring $c_\varnothing$.

Virtual AC

A CFN P is Virtual AC iff Bool(P) has a non empty AC closure.

Bool(P) [9]

Given a CFN $P = (X, D, C, k)$, Bool(P) is the CSP $(X, D, C - \{c_\varnothing\}, 1)$.

Bool(P) forbids all positive cost assignments, ignoring $c_\varnothing$.

Virtual AC

A CFN P is Virtual AC iff Bool(P) has a non empty AC closure.

Virtual AC

Same fixpoints as TRW-S [26], MPLP1[56], SRMP [24], Max-Sum diffusion [10, 29], Aug-DAG[28]...

Connection with Constraint Satisfaction

Virtual AC[10]

1. solves tree-structured problems,
2. solves CFNs with submodular cost functions (Monge matrices)
3. solves CFNs for which AC is a decision procedure in Bool(P).

Virtual AC[10]

1. solves tree-structured problems,
2. solves CFNs with submodular cost functions (Monge matrices)
3. solves CFNs for which AC is a decision procedure in Bool(P).

1. Any solution of Bool(P) has cost $c_\varnothing$ and is therefore optimal.
2. A problem which is VAC and has only one value a in each domain such that $c_i(a) = 0$ is solved.
3. There is always at least one such value (or else not VAC).

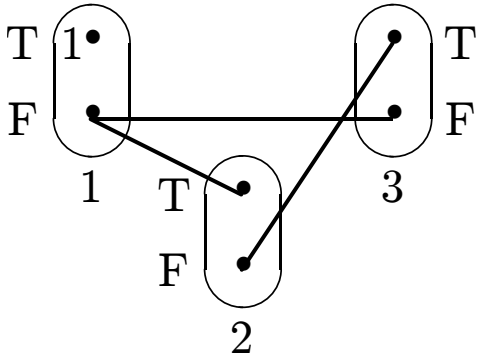
How do we enforce VAC ?

How do we enforce VAC ?

Iterative procedure

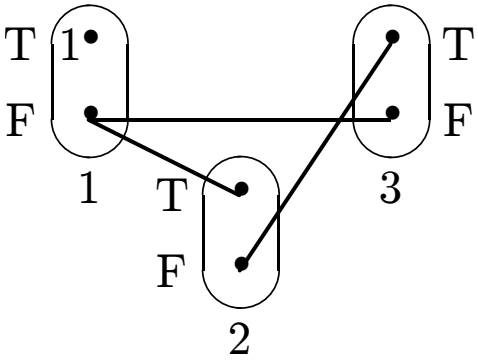
1. Enforce AC in Bool(P) until a wipe-out occurs (record EPTs)
2. Extract a minimal set of EPTs sufficient for the wipe-out
3. Apply cost EPTs on P using suitable cost moves

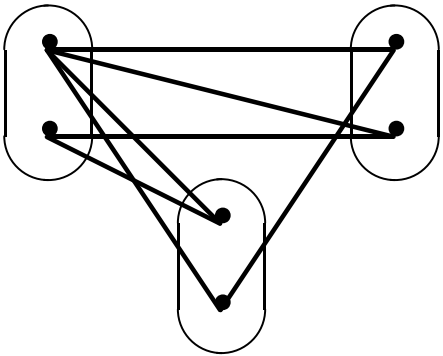
A “simple” example



Original problem

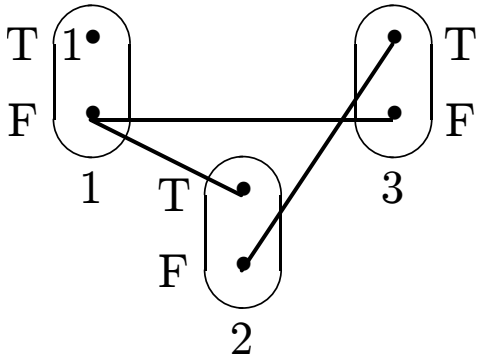
A “simple” example

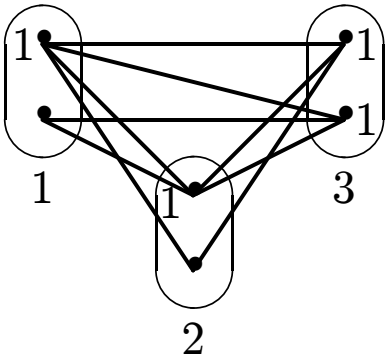




AC: deleting $(3, F)$ and $(2, T)$

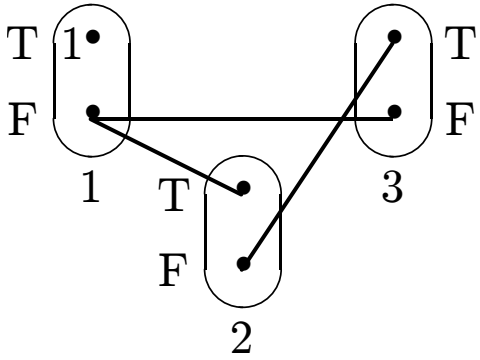
A “simple” example

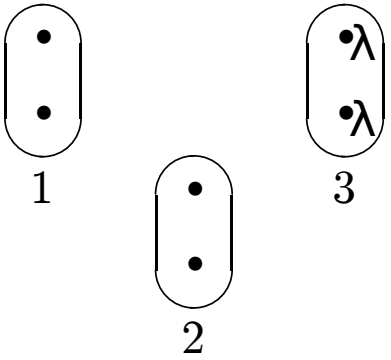




AC: deleting $(3, T)$: wipe out with 3 EPTs !

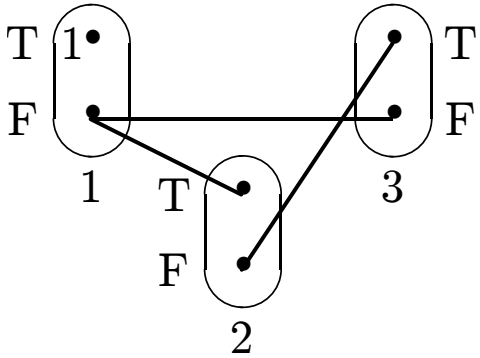
A “simple” example

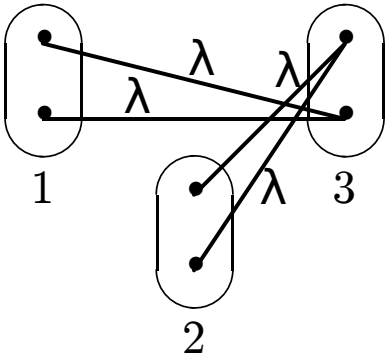




We want to bring λ cost unit to x_3, λ unknown.

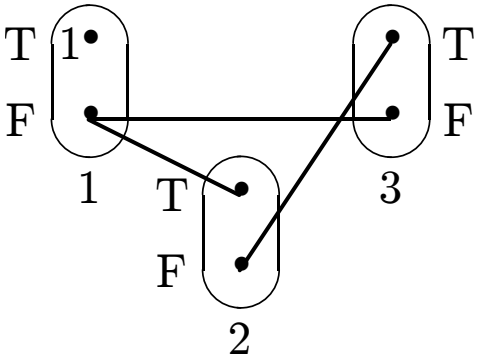
A "simple" example

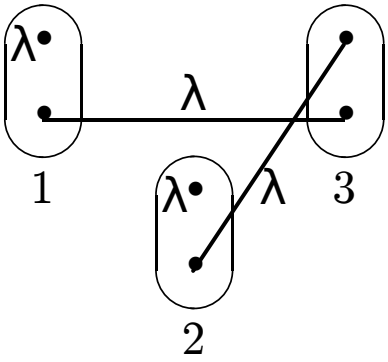




This requires λ virtual cost that needs to be paid by concrete costs...

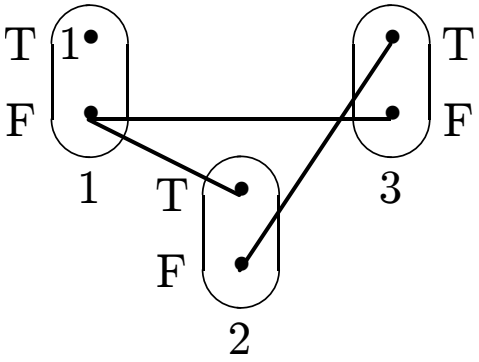
A “simple” example

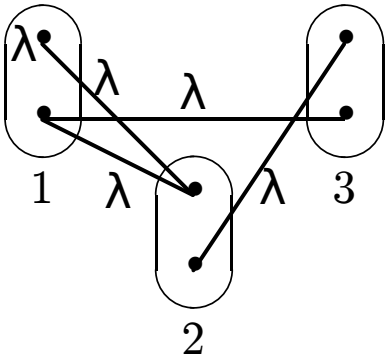




This requires λ virtual cost that needs to be paid by concrete costs... or propagated through EPTs

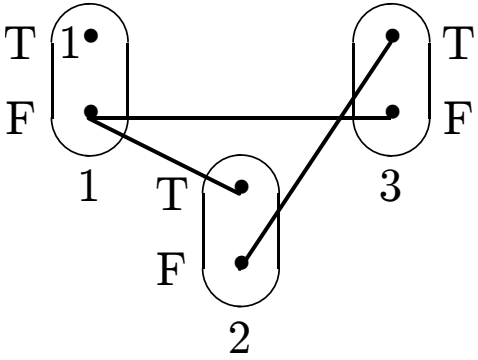
A "simple" example

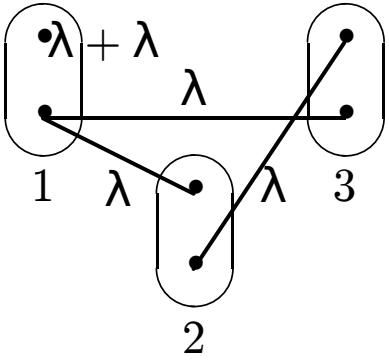




This requires λ virtual cost that needs to be paid by concrete costs... or propagated through EPTs back to concrete costs

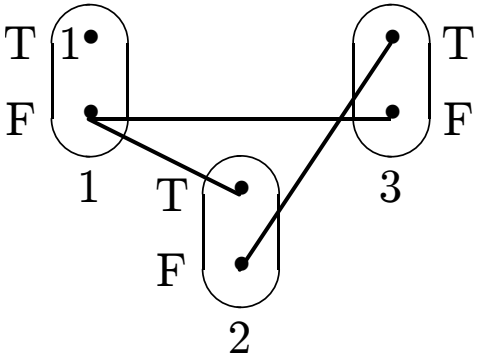
A "simple" example

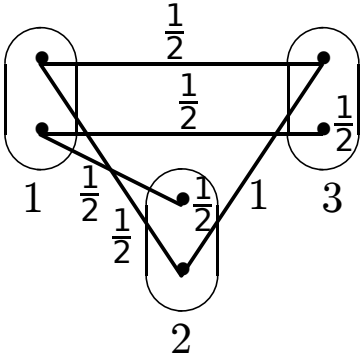




We need 2λ on $(1, T)$ and have only 1 unit of cost: $\lambda = \frac{1}{2}$

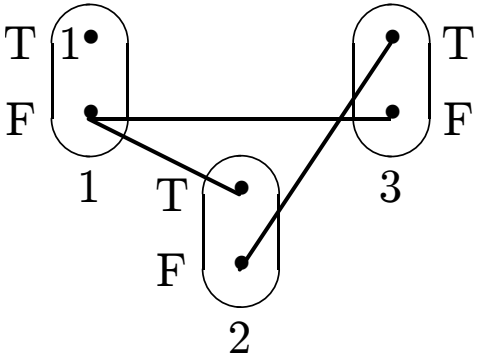
A “simple” example

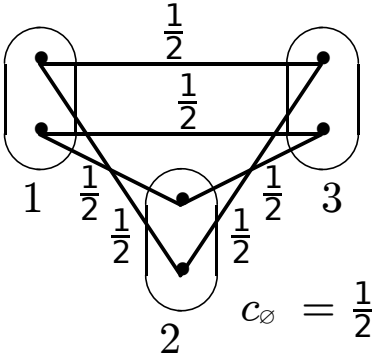




We replay the EPTs using the values of λ

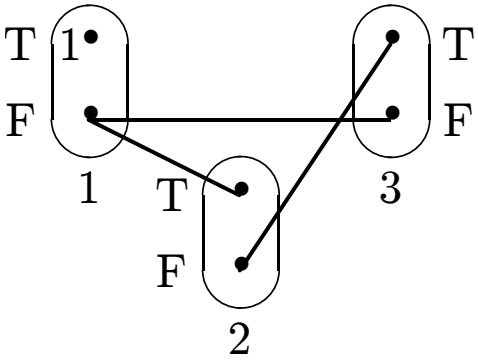
A “simple” example





At the end we are able to project λ to $c_\varnothing$

A “simple” example



Costs in $\mathbb{Q}$. In practice: fixed point representation.

Complexity

Table cost functions

1. Each iteration is in $O(ed^r)$, ε-convergence in $O(ed^r.k/\varepsilon)$
2. accelerates CPLEX on the ILP formulation of a CFN/CPD (local polytope[64]).

Complexity

Table cost functions

1. Each iteration is in $O(ed^r)$, ε-convergence in $O(ed^r.k/\varepsilon)$
2. accelerates CPLEX on the ILP formulation of a CFN/CPD (local polytope[64]).

Prusa and Werner showed that any "normal" LP can be reduced to a "local polytope" problem in linear time (constructive proof).

Complexity

Table cost functions

1. Each iteration is in $O(ed^r)$, ε-convergence in $O(ed^r.k/\varepsilon)$
2. accelerates CPLEX on the ILP formulation of a CFN/CPD (local polytope[64]).

Prusa and Werner showed that any "normal" LP can be reduced to a "local polytope" problem in linear time (constructive proof).

`toulbar2` uses EDAC, possibly VAC at root node or during search (pairwise cost functions). $h(n) = c_\emptyset$.

Variable/value ordering

Heuristics are crucial for efficiency

- node: which node should I choose in *open* (A^*/HBFS)
- variable: which variable should I split (DFS/HBFS)
- value: which son should I explore first (DFS/HBFS)

Variable/value ordering

Heuristics are crucial for efficiency

- node: which node should I choose in *open* (A^*/HBFS)
- variable: which variable should I split (DFS/HBFS)
- value: which son should I explore first (DFS/HBFS)

General principles

- node, value: most promising one (low $h(n)$, deep)
- variable: max. increase in $h(n)$ (first fail principle)

Variable/value ordering

Heuristics are crucial for efficiency

- node: which node should I choose in *open* (A^*/HBFS)
- variable: which variable should I split (DFS/HBFS)
- value: which son should I explore first (DFS/HBFS)

General principles

- node, value: most promising one (low $h(n)$, deep)
- variable: max. increase in $h(n)$ (first fail principle)

Clever variable ordering heuristics that *learn* which variables are "hard" during search. Eg.: weighted degree.[6]

Variable elimination

Relies on two simple operations on cost functions

Variable elimination

Relies on two simple operations on cost functions

Sum of 2 cost functions. $O(d^{|S \cup T|})$

$f_{S \cup T} = g_S + h_T$ defined as $f_{S \cup T}(x) = g_S(x[S]) + h_T(x[T])$

Relies on two simple operations on cost functions

Sum of 2 cost functions. $O(d^{|S \cup T|})$

$f_{S \cup T} = g_S + h_T$ defined as $f_{S \cup T}(x) = g_S(x[S]) + h_T(x[T])$

min-Elimination of a variable. $O(d^{|S|})$

$$g_S^{-i}(x) = \min_{r \in D^i} g_S(x \cdot i_r)$$

Each x has an associated arg min, denoted as x^{+i}.

Variable elimination

Eliminating variable i

1. Let $C_i = \{c_S \in C \mid i \in S\}$
2. compute $m_T = (\sum_{c_S \in C_i} c_S)^{-i}$
3. replace i and C_i by m_T

Variable elimination

Eliminating variable i

1. Let $C_i = \{c_S \in C \mid i \in S\}$
2. compute $m_T = (\sum_{c_S \in C_i} c_S)^{-i}$
3. replace i and C_i by m_T

- one less variable and same optimal energy
- x, optimal solution can be extended to an optimal solution of the original problem using $x[T]^{+i}$.

Variable elimination

Eliminating variable i

1. Let $C_i = \{c_S \in C \mid i \in S\}$
2. compute $m_T = (\sum_{c_S \in C_i} c_S)^{-i}$
3. replace i and C_i by m_T

- one less variable and same optimal energy
- x, optimal solution can be extended to an optimal solution of the original problem using $x[T]^{+i}$.

Eliminate any variable i with degree less than δ or such that $|D_i| = 1$ on the fly, at each node.[30]

Tree decomposition

Cutoffs generate sparsity

Distant residue have no interacting terms generating conditional independencies that can be captured in a “tree decomposition”.

Tree decomposition

Cutoffs generate sparsity

Distant residue have no interacting terms generating conditional independencies that can be captured in a "tree decomposition".

Definition (Tree decomposition of $G = (V, E)$[4,48])

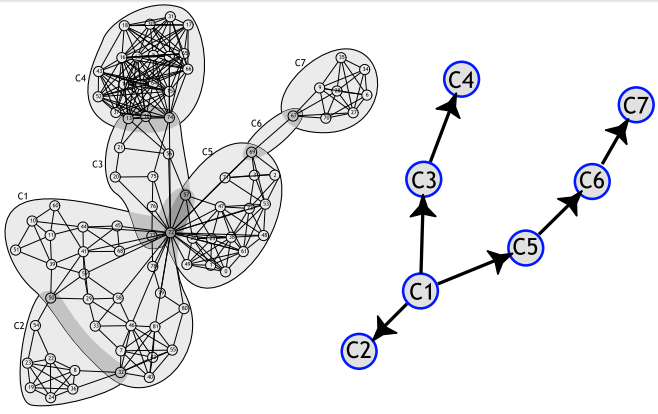
Pair (K, T), K is a set of subsets of V (clusters), (K, T) is a tree.

1. Clusters cover all variables: $\cup_{I \in \mathbf{K}} I = V$
2. Clusters cover edges: $\forall e \in E, \exists I \in K \mid e \subset I$
3. RIP: if $i \in V$ appears in two clusters I and m, then i appears in all clusters between I and m in T (unique path).

Tree decomposition

Cutoffs generate sparsity

Distant residue have no interacting terms generating conditional independencies that can be captured in a “tree decomposition”.



Tree decomposition

Combined with DFS or HBFS

- assigning all variables of a *separator* creates disjoint problems.
- they can be solved independently (additive complexity, not multiplicative).

Tree decomposition

Combined with DFS or HBFS

- assigning all variables of a *separator* creates disjoint problems.
- they can be solved independently (additive complexity, not multiplicative).

- 2006: dynamic programming[4] based algorithm introduced for side chain packing by Xu and Berger (JACM). Space complexity issues. Heuristically simplified in SCWRL3 and 4.[63]
- 2003: combined with tree search and local consistencies for CFN.[11,16,22] Much better space/time behavior.

Constraints

You may add constraints (cost $+\infty$) that could break ergodicity.

Beyond pairwise finite energies

Constraints

You may add constraints (cost $+\infty$) that could break ergodicity.

3-bodies terms

It is possible to use terms involving more than 2 variables.

Beyond pairwise finite energies

Constraints

You may add constraints (cost $+\infty$) that could break ergodicity.

3-bodies terms

It is possible to use terms involving more than 2 variables.

Or even an arbitrary number of variables

If it has a suitable semantics. This is a global cost function.

Examples of "global cost functions"

Global Cardinality Constraint on $S \subset X$

Is the set of variables S, the number of times each possible value is used is constrained by a lower and upper bound.

Examples of "global cost functions"

Global Cardinality Constraint on $S \subset X$

Is the set of variables S, the number of times each possible value is used is constrained by a lower and upper bound.

Regular (Grammar) on $S \subset X$

A finite state automata (grammar) defines the language of the authorized assignment of S. Can be weighted.

Examples of "global cost functions"

Global Cardinality Constraint on $S \subset X$

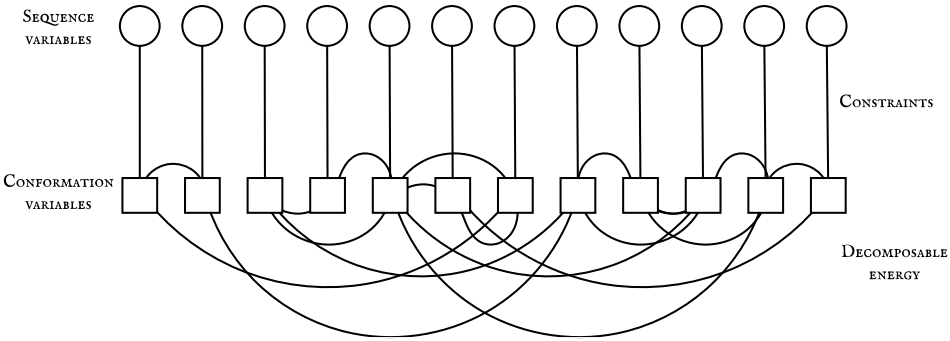
Is the set of variables S, the number of times each possible value is used is constrained by a lower and upper bound.

Regular (Grammar) on $S \subset X$

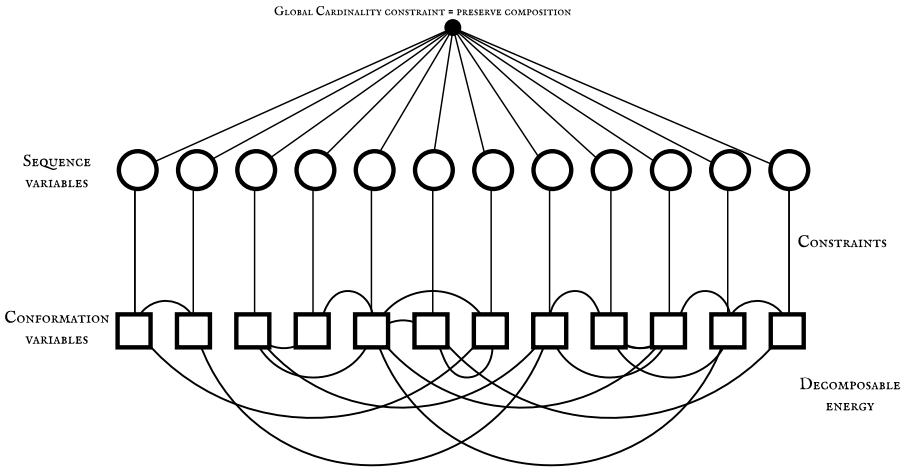
A finite state automata (grammar) defines the language of the authorized assignment of S. Can be weighted.

Local consistency enforced by graph algorithms (mincost flow for *GCC*) or decomposition in ternary cost functions (*WeightedRegular*). More than just these (see the doc).

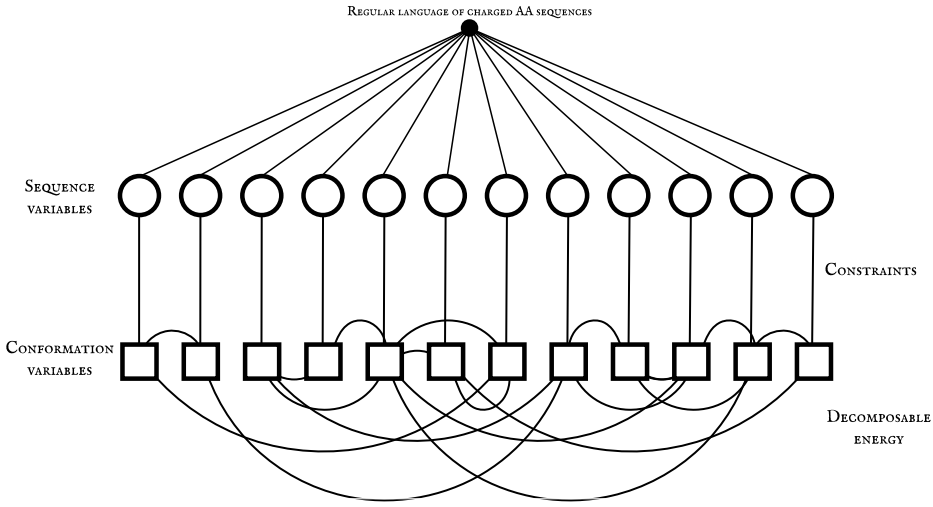
More complex design models



More complex design models



More complex design models



The time needed to solve these models has to be tested.

Internal local search method

INCOP - IdWalk[39]

A stochastic local search algorithm that provides a non trivial upper bound. Does not deal with global cost functions.

Internal local search method

INCOP - IdWalk[39]

A stochastic local search algorithm that provides a non trivial upper bound. Does not deal with global cost functions.

You just need to add `-i` to your command line.

Neighborhoods

$x \in D^X$ an assignment. $x' \in D^X$ is a neighbor of x iff it can be obtained by perturbating x. Eg. change the value of one or more variables.

Stochastic local search

Neighborhoods

$x \in D^X$ an assignment. $x' \in D^X$ is a neighbor of x iff it can be obtained by perturbating x. Eg. change the value of one or more variables.

Stochastic Local Search starting from x

- Build x' (best/random) neighbor of x $(p, 1-p)$
- If $E(x') \leq E(x)$, accept it (unless taboo move)
- Else (possibly) accept with probability $e^{\frac{E(x)-E(x')}{T}}$
- Adjust T, p, taboo moves list and repeat.

Stochastic local search

Neighborhoods

$x \in D^X$ an assignment. $x' \in D^X$ is a neighbor of x iff it can be obtained by perturbating x. Eg. change the value of one or more variables.

Stochastic Local Search starting from x

- Build x' (best/random) neighbor of x $(p, 1-p)$
- If $E(x') \leq E(x)$, accept it (unless taboo move)
- Else (possibly) accept with probability $e^{\frac{E(x)-E(x')}{T}}$
- Adjust T, p, taboo moves list and repeat.

Monte Carlo, Taboo, Greedy Stochastic...

Variable Neighborhood search

Variable Neighborhood Search

Set of neighborhoods: $\{N_1, \ldots, N_n\}$. $i = 1$.

1. Let $x' \in N_i(x)$
2. Greedy local search from x' using N_i
3. If solution improved $x \leftarrow x'$, $i \leftarrow 1$
4. Else $i \leftarrow i + 1$

Combining VNS and exact search

VNS+toulbar2[41]

1. N_i: change $f(i)$ randomly chosen variables (f increasing).
2. Greedy local search $\rightarrow$ `toulbar2`, partial search mode (LDS[21,36]).
3. Outer loop increasing partial search scope (until complete).
4. Tree decomposition aware + parallelized (MPI).

Provides better solutions sooner but HBFS better at proving optimality.

Computing the partition function Z

A #-P complete problem. One call to a #-P oracle can solve any problem in the PH.[59] Intimidating.

$$Z = \sum_{x \in D^X} e^{-\frac{E(x)}{k_B T}}$$

Computing the partition function Z

A #-P complete problem. One call to a #-P oracle can solve any problem in the PH.[59] Intimidating.

$$Z = \sum_{x \in D^X} e^{-\frac{E(x)}{k_B T}}$$

Z and affinity in a solvated complex $A + B$

$$K_a = e^{-\frac{G_{AB} - (G_A + G_B)}{k_B T}} = \frac{Z_{AB}}{Z_A Z_B}$$

$$K_a \approx e^{-\frac{E_{AB} - (E_A + E_B)}{k_B T}}$$

Beyond optimization

Computing the partition function Z

A #-P complete problem. One call to a #-P oracle can solve any problem in the PH.[59] Intimidating.

$$Z = \sum_{x \in D^X} e^{-\frac{E(x)}{k_B T}}$$

Z and affinity in a solvated complex $A + B$

$$K_a = e^{-\frac{G_{AB} - (G_A + G_B)}{k_B T}} = \frac{Z_{AB}}{Z_A Z_B}$$

$$K_a \approx e^{-\frac{E_{AB} - (E_A + E_B)}{k_B T}}$$

Possible guaranteed access to fixed BB/sequence "free energy"

Computing Z with ε-guarantees

Deterministic Finite Distance Guarantees

1. (Weighted) #SAT solvers (Cachet,[49,50] SharpSAT[58])
2. Knowledge compilers: compile graphical models in a target "language" where counting is easy (ACE,[7] minic2d[42]). Rely on SAT solvers.
3. K^*: extension of DEE/A* search.[15,40] Exploits A^* capacity to sort solutions in increasing order of energy.

Computing Z with ε-guarantees

Deterministic Finite Distance Guarantees

1. (Weighted) #SAT solvers (Cachet,[49,50] SharpSAT[58])
2. Knowledge compilers: compile graphical models in a target "language" where counting is easy (ACE,[7] minic2d[42]). Rely on SAT solvers.
3. K^*: extension of DEE/A* search.[15,40] Exploits A^* capacity to sort solutions in increasing order of energy.

Limited to very small problems.

A CFN approach

Z_ε^{*62}

1. (DFS)+(upper bound $\bar{z}(n)$ on Z)+(search invariant)
2. Enumeration-based: accumulate probability masses in $\hat{Z} < Z$
3. Boosted by $(+, \times)$ on the fly variable elimination
4. Maintains an upper bound U on thrown away probability mass
5. Prunes if $U + \bar{z}(n) \leq \varepsilon \hat{Z}$ (invariant)

$$Z \geq \hat{Z} \geq \frac{Z}{1+\varepsilon}$$

Z_ε^{*}[62]

1. (DFS)+(upper bound $\bar{z}(n)$ on Z)+(search invariant)
2. Enumeration-based: accumulate probability masses in $\hat{Z} < Z$
3. Boosted by $(+, \times)$ on the fly variable elimination
4. Maintains an upper bound U on thrown away probability mass
5. Prunes if $U + \bar{z}(n) \leq \varepsilon \hat{Z}$ (invariant)

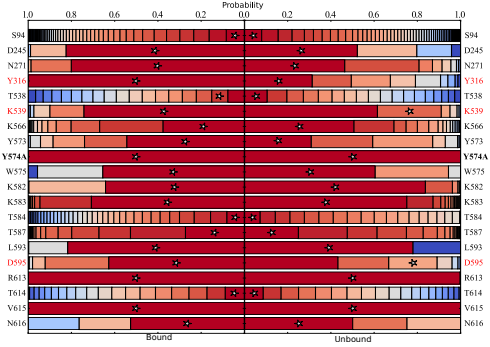
$$Z \geq \hat{Z} \geq \frac{Z}{1+\varepsilon}$$

Simple upper bounds ($c_\varnothing$ and unary cost functions) or (dynamic programming on a tree cover). Reinforced by local consistencies.

1AHW[0] - Skempi (most cases: little E vs. G difference)

Using beta_nov15, 11 mutations+WT

- $\Delta\Delta E$: $R = -0.18$ $\Delta\Delta G$: $R = 0.65$
- A pure enumeration of $\approx 85\%$ of Z took 5 days CPU time.
- Guaranteed $Z^*_{10^{-3}}$ took 5 minutes.



[0] A Complex of Extracellular Domain of Tissue Factor with an Inhibitory Fab

Practicals this afternoon

Virtual machines with PyRosetta, `toulbar2`, OSPREY

- Computing energy matrices with PyRosetta (OSPREY)
- Solving the SCP problem with Pyrosetta and `toulbar2`
- Designing with PyRosetta and `toulbar2`
- Enumerating sequence·conformations
- Enumerating sequences only
- Biasing the energy function with fitness.
- Affinity: $\Delta\Delta G$ and $\Delta\Delta E$

Many thanks...

- Juan Cortes, Frederic Cazals, Charles Robert, Yann Ponti (organizers)
- **D. Simoncini, C. Viricel** (Postdoc, PhD student)
- S. de Givry, G. Katsirelos, D. Allouche, M. Zytnicki (`toulbar2`)
- J. Larrosa, E. Rollon, J. H Lee... (`toulbar2`)
- S. Barbe, S. Traoré, I. André (protein design, LISBP)
- B. Donald, K. Roberts (U. North Carolina, OSPREY)
- D. Baker lab., W. Sheffler (U. Washington, Rosetta), S. Lyskov (J. Hopkins, Pyrosetta)
- many people I met and which gave useful advices (Juan, Frederic, Thomas...).

Thank you!

Please do come to chat a bit. I'm always happy to discuss and discover new stuff

References I

[1] David Allouche et al. “Computational protein design as an optimization problem”. In: *Artificial Intelligence* 212 (2014), pp. 59–79.

[2] Oscar Alvizo and Stephen L Mayo. “Evaluating and optimizing computational protein design force fields using fixed composition-based negative design”. In: *Proc. Natl. Acad. Sci. U.S.A.* 105.34 (2008), pp. 12242–12247.

[3] C. Anfinsen. “Principles that govern the folding of protein chains”. In: *Science* 181.4096 (1973), pp. 223–253.

[4] Umberto Bertelé and Francesco Brioshi. *Nonserial Dynamic Programming*. Academic Press, 1972.

[5] E. Boros and P. Hammer. “Pseudo-Boolean Optimization”. In: *Discrete Appl. Math.* 123 (2002), pp. 155–225.

[6] Frédéric Boussemart et al. “Boosting systematic search by weighting constraints”. In: *ECAI*. Vol. 16. 2004, p. 146.

[7] Mark Chavira and Adnan Darwiche. “On probabilistic inference by weighted model counting”. In: *Artificial Intelligence* 172.6 (2008), pp. 772–799.

[8] M C. Cooper and T. Schiex. “Arc consistency for soft constraints”. In: *Artificial Intelligence* 154.1-2 (2004), pp. 199–227.

[9] Martin C Cooper et al. “Virtual Arc Consistency for Weighted CSP.”. In: *AAAI*. Vol. 8. 2008, pp. 253–258.

References II

[10] M. Cooper et al. "Soft arc consistency revisited". In: *Artificial Intelligence* 174 (2010), pp. 449–478.

[11] Rina Dechter and Robert Mateescu. "AND/OR search spaces for graphical models". In: *Artificial intelligence* 171.2-3 (2007), pp. 73–106.

[12] Roland L Dunbrack. "Rotamer Libraries in the 21 st Century". In: *Current opinion in structural biology* 12.4 (2002), pp. 431–440.

[13] Eugene C. Freuder. "Eliminating Interchangeable Values in Constraint Satisfaction Problems". In: *Proc. of AAAI'91*. Anaheim, CA, 1991, pp. 227–233.

[14] Pablo Gainza et al. "OSPREY: Protein design with ensembles, flexibility, and provable algorithms". In: *Methods Enzymol.* 523 (2012), pp. 87–107.

[15] Ivelin Georgiev, Ryan H Lilien, and Bruce R Donald. "The minimized dead-end elimination criterion and its application to protein redesign in a hybrid scoring and search algorithm for computing partition functions over molecular ensembles.". In: *Journal of computational chemistry* 29.10 (July 2008), pp. 1527–42. ISSN: 1096-987X. DOI: 10.1002/jcc.20909. URL: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3263346%5C&tool=pmcentrez%5C&rendertype=abstract.

[16] S. de Givry, T. Schiex, and G. Verfaillie. "Exploiting Tree Decomposition and Soft Local Consistency in Weighted CSP". In: *Proc. of the National Conference on Artificial Intelligence, AAAI-2006*. 2006, pp. 22–27.

References III

[17] R F Goldstein. "Efficient rotamer elimination applied to protein side-chains and related spin glasses.". In: *Biophysical journal* 66.5 (May 1994), pp. 1335–40. ISSN: 0006-3495. DOI: 10.1016/S0006-3495(94)80923-3. URL: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=1275854%5C&tool=pmcentrez%5C&rendertype=abstract.

[18] Mark A Hallen, Jonathan D Jou, and Bruce R Donald. "LUTE (Local Unpruned Tuple Expansion): Accurate Continuously Flexible Protein Design with General Energy Functions and Rigid-rotamer-like Efficiency". In: *Research in Computational Molecular Biology*. Springer. 2016, pp. 122–136.

[19] Mark A Hallen, Daniel A Keedy, and Bruce R Donald. "Dead-end elimination with perturbations (DEEPer): A provable protein design algorithm with continuous sidechain and backbone flexibility". In: *Proteins* 81.1 (2013), pp. 18–39.

[20] Peter L Hammer, Pierre Hansen, and Bruno Simeone. "Roof duality, complementation and persistency in quadratic 0–1 optimization". In: *Mathematical programming* 28.2 (1984), pp. 121–155.

[21] W. D. Harvey and M. L. Ginsberg. "Limited Discrepency Search". In: *Proc. of the 14th IJCAI*. Montréal, Canada, 1995.

[22] Philippe Jégou and Cyril Terrioux. "Hybrid backtracking bounded by tree-decomposition of constraint networks". In: *Artificial Intelligence* 146.1 (2003), pp. 43–75.

[23] Joerg Kappes et al. "A comparative study of modern inference techniques for discrete energy minimization problems". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 1328–1335.

References IV

[24] Vladimir Kolmogorov. "A new look at reweighted message passing". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 37.5 (2015), pp. 919–930.

[25] Vladimir Kolmogorov. "Convergent tree-reweighted message passing for energy minimization". In: *IEEE transactions on pattern analysis and machine intelligence* 28.10 (2006), pp. 1568–1583.

[26] Vladimir Kolmogorov. "Convergent tree-reweighted message passing for energy minimization". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 28.10 (2006), pp. 1568–1583.

[27] A M C A. Koster. "Frequency assignment: Models and Algorithms". Available at www.zib.de/koster/thesis.html. PhD thesis. The Netherlands: University of Maastricht, Nov. 1999.

[28] VK Koval' and Mykhailo Ivanovich Schlesinger. "Two-dimensional programming in image analysis problems". In: *Avtomatika i Telemekhanika* 8 (1976), pp. 149–168.

[29] VA Kovalevsky and VK Koval. "A diffusion algorithm for decreasing energy of max-sum labeling problem". In: *Glushkov Institute of Cybernetics, Kiev, USSR* (1975).

[30] J. Larrosa. "Boosting search with variable elimination". In: *Principles and Practice of Constraint Programming - CP 2000*. Vol. 1894. LNCS. Singapore, Sept. 2000, pp. 291–305.

[31] J. Larrosa and T. Schiex. "In the quest of the best form of local consistency for Weighted CSP". In: *Proc. of the 18th IJCAI*. Acapulco, Mexico, Aug. 2003, pp. 239–244.

[32] J. Larrosa et al. "Existential arc consistency: getting closer to full arc consistency in weighted CSPs". In: *Proc. of the 19th IJCAI*. Edinburgh, Scotland, Aug. 2005, pp. 84–89.

References V

[33] A Leaver-Fay et al. “Scientific benchmarks for guiding macromolecular energy function improvement”. In: *Methods Enzymol.* 523 (2013), p. 109.

[34] Christophe Lecoutre, Olivier Roussel, and Djamel E Dehani. “WCSP integration of soft neighborhood substitutability”. In: *Principles and Practice of Constraint Programming*. Springer. 2012, pp. 406–421.

[35] L L Looger and H W Hellinga. “Generalized dead-end elimination algorithms make large-scale protein side-chain structure prediction tractable: implications for protein design and structural genomics.”. In: *Journal of molecular biology* 307.1 (Mar. 2001), pp. 429–45. ISSN: 0022-2836. DOI: 10.1006/jmbi.2000.4424. URL: http://www.ncbi.nlm.nih.gov/pubmed/11243829.

[36] Samir Loudni and Patrice Boizumault. “Solving constraint optimization problems in anytime contexts”. In: *Proc. of IJCAI'2003*. 2003, pp. 251–256.

[37] S C Lovell et al. “The penultimate rotamer library.”. In: *Proteins* 40.3 (Aug. 2000), pp. 389–408. ISSN: 0887-3585. URL: http://www.ncbi.nlm.nih.gov/pubmed/10861930.

[38] Christopher Negron and Amy E Keating. “Multistate protein design using CLEVER and CLASSY”. In: *Methods Enzymol* 523 (2013), pp. 171–190.

[39] Bertrand Neveu, Gilles Trombettoni, and Fred Glover. “Id walk: A candidate list strategy with a simple diversification device”. In: *Principles and Practice of Constraint Programming–CP 2004* (2004), pp. 423–437.

References VI

[40] Adegoke A Ojewole et al. "BBK^*(Branch and Bound over K^*): A Provable and Efficient Ensemble-Based Algorithm to Optimize Stability and Binding Affinity over Large Sequence Spaces". In: *International Conference on Research in Computational Molecular Biology*. Springer. 2017, pp. 157–172.

[41] A. Ouali et al. "Iterative Decomposition Guided Variable Neighborhood Search for Graphical Model Energy Minimization". In: *Proc. of the 33rd Conference on Uncertainty in Artificial Intelligence*. Ed. by Association for Uncertainty in Artificial Intelligence. Sydney, Australia, 2017.

[42] Umut Oztok and Adnan Darwiche. "A top-down compiler for sentential decision diagrams". In: *Proceedings of the 24th International Conference on Artificial Intelligence. AAAI Press*. 2015.

[43] N. Pierce et al. "Conformational splitting: A more powerful criterion for dead-end elimination". In: *Journal of computational chemistry* 21.11 (2000), pp. 999–1009.

[44] Navin Pokala and Tracy M Handel. "Energy functions for protein design: adjustment with protein–protein complex affinities, models for the unfolded state, and negative design of solubility and specificity". In: *Journal of molecular biology* 347.1 (2005), pp. 203–227.

[45] C Pralet, G Verfaillie, and T Schiex. "An algebraic graphical model for decision with uncertainties, feasibilities, and utilities". In: *Journal of Artificial Intelligence Research* 29 (2007), pp. 421–489.

[46] Daniel Prusa and Tomas Werner. "Universality of the local marginal polytope". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 37.4 (2015), pp. 898–904.

[47] Florian Richter et al. "De novo enzyme design using Rosetta3". In: *PloS one* 6.5 (2011), e19230.

References VII

[48] N. Robertson and P. D. Seymour. "Graph minors. II. Algorithmic aspects of tree-width". In: *Journal of Algorithms* 7 (1986), pp. 309–322.

[49] Tian Sang, Paul Beame, and Henry Kautz. "Solving Bayesian networks by weighted model counting". In: *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*. Vol. 1. 2005, pp. 475–482.

[50] Tian Sang, Paul Beame, and Henry A Kautz. "Performing Bayesian inference by weighted model counting". In: *AAAI*. Vol. 5. 2005, pp. 475–481.

[51] T. Schiex. "Arc consistency for soft constraints". In: *Principles and Practice of Constraint Programming - CP 2000*. Vol. 1894. LNCS. Singapore, Sept. 2000, pp. 411–424.

[52] T. Schiex, H. Fargier, and G. Verfaillie. "Valued Constraint Satisfaction Problems: hard and easy problems". In: *Proc. of the 14th IJCAI*. Montréal, Canada, Aug. 1995, pp. 631–637.

[53] M.I. Schlesinger. "Sintaksicheskiy analiz dvumernykh zritelnikh signalov v usloviyakh pomekh (Syntactic analysis of two-dimensional visual signals in noisy conditions)". In: *Kibernetika* 4 (1976), pp. 113–130.

[54] Maxim V Shapovalov and Roland L Dunbrack. "A smoothed backbone-dependent rotamer library for proteins derived from adaptive kernel density estimates and regressions". In: *Structure* 19.6 (2011), pp. 844–858.

[55] P. Shenoy. "Valuation-based systems for discrete optimization". In: *Uncertainty in AI*. Ed. by Bonissone et al. North-Holland Publishers, 1991.

References VIII

[56] David Sontag et al. "Tightening LP relaxations for MAP using message passing". In: *arXiv preprint arXiv:1206.3288* (2012).

[57] Paul Swoboda et al. "Partial optimality by pruning for MAP-inference with general graphical models". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 1170–1177.

[58] Marc Thurley. "sharpSAT–counting models with advanced component caching and implicit BCP". In: *International Conference on Theory and Applications of Satisfiability Testing*. Springer. 2006, pp. 424–429.

[59] Seinosuke Toda. "On the computational power of PP and $\oplus$P". In: *Foundations of Computer Science, 1989., 30th Annual Symposium on*. IEEE. 1989, pp. 514–519.

[60] Seydou Traoré et al. "Fast search algorithms for computational protein design". In: *Journal of computational chemistry* (2016).

[61] P Tuffery et al. "A new approach to the rapid determination of protein side chain conformations". In: *Journal of Biomolecular structure and dynamics* 8.6 (1991), pp. 1267–1289.

[62] C. Viricel et al. "Guaranteed Weighted Counting for Affinity Computation: Beyond Determinism and Structure". In: *Proc. of the 22nd International Conference on Principles and Practice of Constraint Programming*. Springer. Toulouse, France, Sept. 2016, pp. 736–750.

References IX

[63] Qiang Wang, Adrian A Canutescu, and Roland L Dunbrack. "SCWRL and MolIDE: computer programs for side-chain conformation prediction and homology modeling". In: *Nature protocols* 3.12 (2008), pp. 1832–1847.

[64] T. Werner. "A Linear Programming Approach to Max-sum Problem: A Review.". In: *IEEE Trans. on Pattern Recognition and Machine Intelligence* 29.7 (July 2007), pp. 1165–1179. URL: http://dx.doi.org/10.1109/TPAMI.2007.1036.

[65] Jinbo Xu and Bonnie Berger. "Fast and accurate algorithms for protein side-chain packing". In: *Journal of the ACM (JACM)* 53.4 (2006), pp. 533–557.