Designing molecules with cost function networks - Bridging symbolic and numerical AI.

Thomas Schiex

To cite this version:

Thomas Schiex. Designing molecules with cost function networks - Bridging symbolic and numerical AI.. Journées plénières du GDR IA du CNRS, Oct 2018, Paris, France. hal-02785414

HAL Id: hal-02785414
https://hal.inrae.fr/hal-02785414

Submitted on 4 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

Designing molecules with cost function networks

Bridging symbolic and numerical AI

T. Schiex
D. Allouche, S. Barbe, J. Cortes, M. Ruffini, D. Simoncini, A. Voet, J. Vucinic
S. de Givry, G. Katsirelos, M. Zytnicki

October 2018

Propositional logic, Constraint networks

Constraint network (X, C) **Joint feasibility distribution**

- a sequence X of discrete variables x_i, domain D_i

Propositional logic, Constraint networks

Constraint network (X, C) **Joint feasibility distribution**

- a sequence X of discrete variables x_i, domain D_i
- a set C of constraints

Propositional logic, Constraint networks

Constraint network (X, C) Joint feasibility distribution

- a sequence X of discrete variables x_i, domain D_i
- a set C of constraints
- $c_S \in C$ involves variables in $S \subseteq X$ and is a boolean function $\prod_{i \in S} D_i \rightarrow \{t, f\}$

Propositional logic, Constraint networks

Constraint network (X, C) — **Joint feasibility distribution**

- a sequence X of discrete variables x_i, domain D_i
- a set C of constraints
- $c_S \in C$ involves variables in $S \subseteq X$ and is a boolean function $\prod_{i \in S} D_i \rightarrow \{t, f\}$
- Joint boolean function $F(X) = \bigwedge c_S$

Central problems: SAT/CSP and their solvers

- A solution is an assignment of X that satisfies the joint function (NP-complete)
- Algorithms to find a model/solution or a proof (Backtrack, unit/constraint propagation)

Huge impact of symbolic AI

SAT and CSP technologies

- Solving and generating Sudokus (Le Monde)
- Planning and Scheduling[1][2] (Rosetta-Philae probe plan, CP, LAAS/Toulouse)
- Configuration/verification (also neural nets[5])
- Recent theorem proof (Splitting all pythagorean triples in $\mathbb{N}$: 200 TB proof[4])

Huge impact of symbolic AI

SAT and CSP technologies

- Solving and generating Sudokus (Le Monde)
- Planning and Scheduling[1,2] (Rosetta-Philae probe plan, CP, LAAS/Toulouse) cnes
- Configuration/verification (also neural nets[5]) SIEMENS THALES
- Recent theorem proof (Splitting all pythagorean triples in $\mathbb{N}$: 200 TB proof[4])

Excellent to describe, analyze, design perfectly known complex systems.

Huge impact of symbolic AI

SAT and CSP technologies

- Solving and generating Sudokus (Le Monde)
- Planning and Scheduling[1,2] (Rosetta-Philae probe plan, CP, LAAS/Toulouse)
- Configuration/verification (also neural nets[5]) SIEMENS THALES
- Recent theorem proof (Splitting all pythagorean triples in $\mathbb{N}$: 200 TB proof[4])

Excellent to describe, analyze, design perfectly known complex systems.

Biology is full of imperfectly known complex systems.

From boolean function to numerical functions

Cost function network (X, W) Joint cost/feasibility distribution[2,9]

- a sequence X of discrete variables x_i, domain D_i

From boolean function to numerical functions

Cost function network (X, W) Joint cost/feasibility distribution[2,9]

- a sequence X of discrete variables x_i, domain D_i
- a set W of cost functions

From boolean function to numerical functions

Cost function network (X, W) **Joint cost/feasibility distribution**[2,9]

- a sequence X of discrete variables x_i, domain D_i
- a set W of cost functions
- $w_S \in W$ is a numerical function $\prod_{i \in S} D_i$ (possibly infinite costs)

From boolean function to numerical functions

Cost function network (X, W) **Joint cost/feasibility distribution**[2,9]

- a sequence X of discrete variables x_i, domain D_i
- a set W of cost functions
- $w_S \in W$ is a numerical function $\prod_{i\in S} D_i$ (possibly infinite costs)
- Joint cost function $W(X) = \sum w_S$

From boolean function to numerical functions

Cost function network (X, W) Joint cost/feasibility distribution[2,9]

- a sequence X of discrete variables x_i, domain D_i
- a set W of cost functions
- $w_S \in W$ is a numerical function $\prod_{i \in S} D_i$ (possibly infinite costs)
- Joint cost function $W(X) = \sum w_S$

- **Generalizes CSP/SAT**: a constraint is a cost function that maps to $\{0, \infty\}$

From boolean function to numerical functions

Cost function network (X, W) Joint cost/feasibility distribution[2,9]

- a sequence X of discrete variables x_i, domain D_i
- a set W of cost functions
- $w_S \in W$ is a numerical function $\prod_{i \in S} D_i$ (possibly infinite costs)
- Joint cost function $W(X) = \sum w_S$

- **Generalizes CSP/SAT**: a constraint is a cost function that maps to $\{0, \infty\}$
- **Complex interactions of graduality with comparability** (likelihood, preferences)

From boolean function to numerical functions

Cost function network (X, W) Joint cost/feasibility distribution[2,9]

- a sequence X of discrete variables x_i, domain D_i
- a set W of cost functions
- $w_S \in W$ is a numerical function $\prod_{i \in S} D_i$ (possibly infinite costs)
- Joint cost function $W(X) = \sum w_S$

Central problems: PWMaxSAT, WCSP, MAP/MRF

- a solution optimizes the joint cost $W(X)$ (WCSP, NP-complete)
- algorithms to find a solution and a proof of optimality (Branch and bound + cost function propagation, core-based)

Example: MAXCUT with hard edges

Graph $G = (V, E)$ with edge weight function w — Graphical model[3]

- A boolean variable x_i per vertex $i \in V$
- A cost function per edge $e = (i, j) \in E : w_{ij} = w(i, j) \times \mathbb{1}[x_i \neq x_j]$
- Hard edges: constraints with costs 0 or $-\infty$ (when $x_i \neq x_j$)

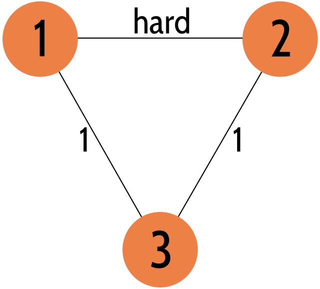
Example: MAXCUT with hard edges

Graph $G = (V, E)$ with edge weight function w — Graphical model[3]

- A boolean variable x_i per vertex $i \in V$
- A cost function per edge $e = (i, j) \in E$: $w_{ij} = w(i, j) \times \mathbb{1}[x_i \neq x_j]$
- Hard edges: constraints with costs 0 or $-\infty$ (when $x_i \neq x_j$)

3-clique

- vertices $\{1, 2, 3\}$
- cut weight 1
- edge $(1, 2)$ hard.



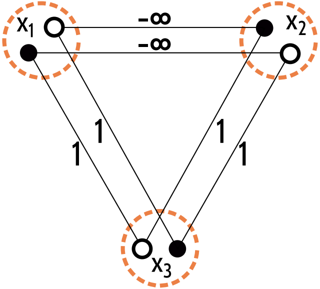
Example: MAXCUT with hard edges

Graph $G = (V, E)$ with edge weight function w — **Graphical model[3]**

- A boolean variable x_i per vertex $i \in V$
- A cost function per edge $e = (i, j) \in E$: $w_{ij} = w(i, j) \times \mathbb{1}[x_i \neq x_j]$
- Hard edges: constraints with costs 0 or $-\infty$ (when $x_i \neq x_j$)

3-clique

- vertices $\{1, 2, 3\}$
- cut weight 1
- edge $(1, 2)$ hard.



toulbar2 input file

MAXCUT on a 3-clique with hard edge

```
{
 "problem" :{"name": "MaxCut", "mustbe": ">0.0"},
 "variables": {"x1": ["l","r"], "x2": ["l","r"], "x3": ["l","r"]},
 "functions": {
     "cut12": {"scope": ["x1","x2"], "costs": [0,-100,-100,0]},
     "cut13": {"scope": ["x1","x3"], "costs": [0,1,1,0]},
     "cut23": {"scope": ["x2","x3"], "costs": [0,1,1,0]}
  }
}
```

MIT licence, https://github.com/toulbar2/toulbar2

The cost distribution

Can be concisely expressed as

- A set of weighted clauses
- An integer linear program
- A Markov Random Field (stochastic graphical model with additive potentials)
- A quadratic boolean polynomial

The cost distribution

Can be concisely expressed as

- A set of weighted clauses
- An integer linear program
- A Markov Random Field (stochastic graphical model with additive potentials)
- A quadratic boolean polynomial

And the WCSP problem tackled with

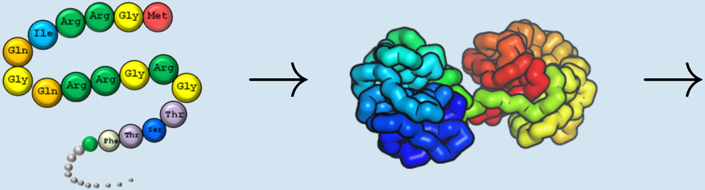
- MaxHS (PWMaxSat solver)
- CPLEX/GUROBI (ILP solver)
- MAP/MRF solvers (very few provide guarantees: toulbar2, daoopt)
- A quadratic boolean polynomial (SDP based BiqMac)

Proteins

Most active molecules of life

Flexible sequence of “amino-acids”, each chosen among a set of 20 natural ones (or more)

Folding



Function

Transporter, binder/regulator, motor, catalyst...
Hemoglobine, TAL effector, ATPase, dehydrogenases...

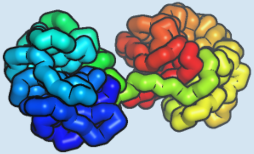
Protein Design

Most active molecules of life

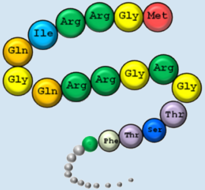
Flexible sequence of “amino-acids”, each chosen among a set of 20 natural ones (or more)

Inverse folding

Function →



→



Transporter, binder/regulator, motor, catalyst...
Hemoglobine, TAL effector, ATPase, dehydrogenases...

Why is it worth designing new proteins?

Eco-friendly chemical/structural nano-agents

- Biodegradable (have been mass produced for billions of year)

Why is it worth designing new proteins?

Eco-friendly chemical/structural nano-agents

- Biodegradable (have been mass produced for billions of year)
- “Easy” to produce (transformed E. coli)

Why is it worth designing new proteins?

Eco-friendly chemical/structural nano-agents

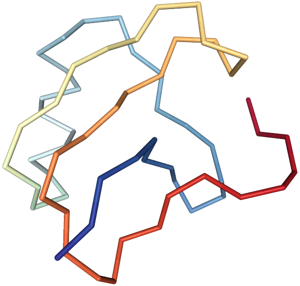
- Biodegradable (have been mass produced for billions of year)
- "Easy" to produce (transformed E. coli)
- Useful for green chemistry[8] (biofuels, plastic recycling, food and feed, cosmetics...), nanotechnologies,[13] drugs...

20^n sequences! intractable for experimental techniques

Protein Design as a discrete optimisation problem

Molecular modeling

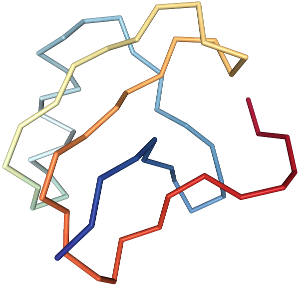
- Full atom model of a protein backbone (assumed to be rigid)

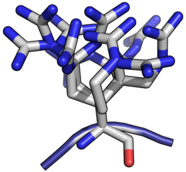


Protein Design as a discrete optimisation problem

Molecular modeling

- Full atom model of a protein backbone (assumed to be rigid)
- Catalog of all 20 side-chains in different conformations ($\approx$ 400 overall)

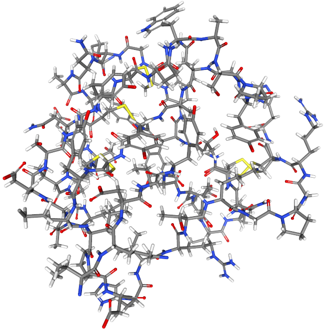




Protein Design as a discrete optimisation problem

Molecular modeling

- Full atom model of a protein backbone (assumed to be rigid)
- Catalog of all 20 side-chains in different conformations (≈ 400 overall)
- Huge sequence-conformation space: 400^n (or more)



Protein stability

Thermodynamics: forces, energy and stability

- Full atom empirical force field (bonds, electrostatics, solvant...)

Protein stability

Thermodynamics: forces, energy and stability

- Full atom empirical force field (bonds, electrostatics, solvant...)
- Usually decomposed as a sum of pairwise terms that depends on atom positions

Protein stability

Thermodynamics: forces, energy and stability

- Full atom empirical force field (bonds, electrostatics, solvant...)
- Usually decomposed as a sum of pairwise terms that depends on atom positions

Imperfect

- Approximations: rigidity, solvent effect
- Very empirical representation of crucial quantum mechanic effects

Finding sequences with low energy conformations

Central problem (plenty of tricky/harder variants)

Maximum stability $\equiv$ Minimum energy NP-hard[7]

Finding sequences with low energy conformations

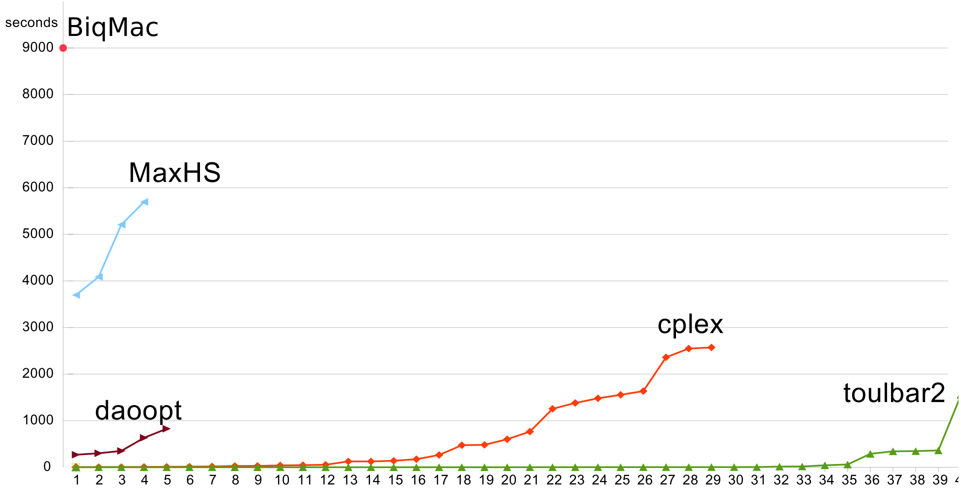
Central problem (plenty of tricky/harder variants)

Maximum stability $\equiv$ Minimum energy NP-hard[7]

As a Cost Function Network

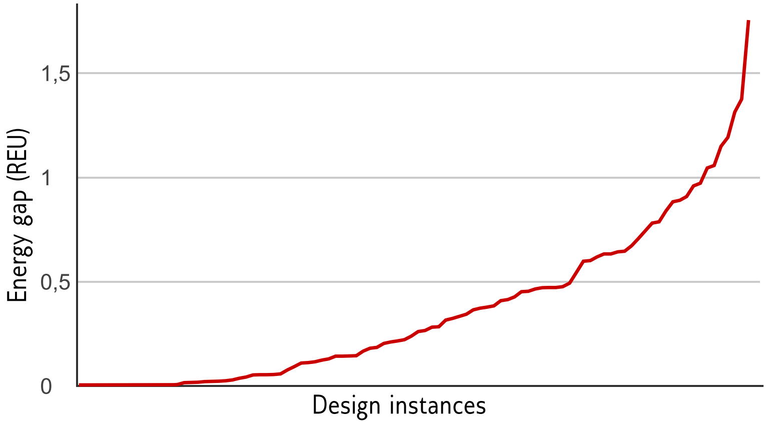
- One variable per position in the protein sequence
- Domain: catalog of few hundreds amino acids conformations
- Functions: decomposed energy (pairwise terms)

Toulbar2 vs. CPLEX, MaxHS...(real instances)



# of instances solved (X) within a per instance cpu-time limit (Y)

Comparison with Rosetta's Simulated annealing[11]

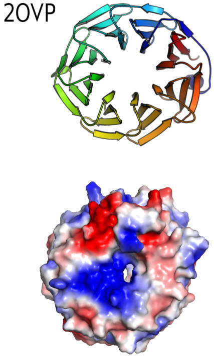


Optimality gap of the Simulated annealing solution as problems get harder

Asymptotic convergence can be arbitrarily slow (infinity can be arbitrarily far)

From bits to atoms (col. A. Voet, KU Leuven, D. Simoncini, INRA/INSA)

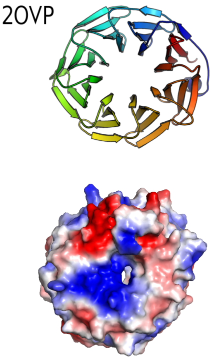
C8 pseudo-symetric 2OVP symmetrized into a nano-component

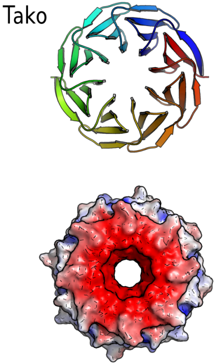


From bits to atoms (col. A. Voet, KU Leuven, D. Simoncini, INRA/INSA)

C8 pseudo-symetric 2OVP symmetrized into a nano-component

-  Tako: (R)evolution + Rosetta/talaris14 8 fold

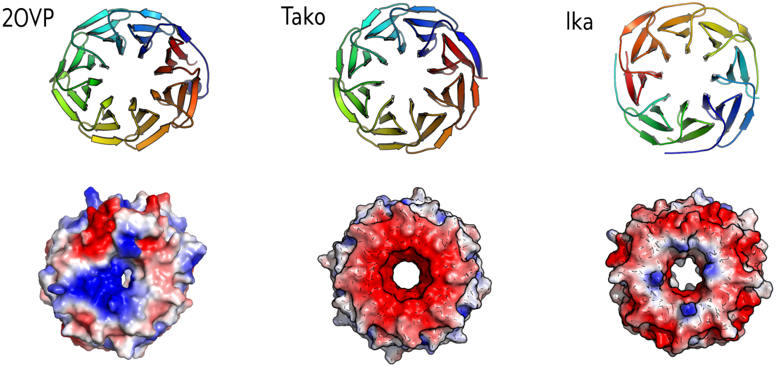




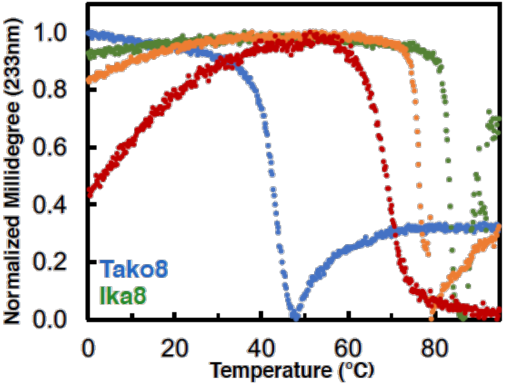
From bits to atoms (col. A. Voet, KU Leuven, D. Simoncini, INRA/INSA)

C8 pseudo-symetric 2OVP symmetrized into a nano-component

-  Tako: (R)evolution + Rosetta/talaris14 8 fold
-  Ika: toulbar2 + talaris14 4 fold



Ika more stable than Tako and can self assemble



Compares Tako and Ika structural stability as temperature increases
(circular dichroism)

Energy function

Imperfect

Simplest way around this: inject more information than just energy.

Energy function

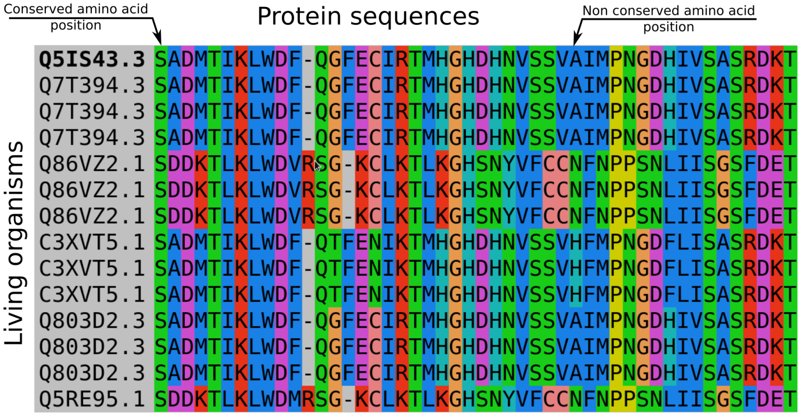
Imperfect

Simplest way around this: inject more information than just energy.

Evolutionary information

- Use similar proteins (homologs) from databases
- All have been through millions of year of selection by “reality”
- Multiple alignment: align similar regions of the sequences

A multiple alignment with conserved positions



```
Q5IS43.3 SADMTIKLWDF-QGFECIRTMHGHDHNVSSVAIMPNGDHIVSASRDKT
Q7T394.3 SADMTIKLWDF-QGFECIRTMHGHDHNVSSVAIMPNGDHIVSASRDKT
Q7T394.3 SADMTIKLWDF-QGFECIRTMHGHDHNVSSVAIMPNGDHIVSASRDKT
Q7T394.3 SADMTIKLWDF-QGFECIRTMHGHDHNVSSVAIMPNGDHIVSASRDKT
Q86VZ2.1 SDDKTLKLWDVRSG-KCLKTLKGHSNYVFCCNFNPPSNLIISGSFDET
Q86VZ2.1 SDDKTLKLWDVRSG-KCLKTLKGHSNYVFCCNFNPPSNLIISGSFDET
Q86VZ2.1 SDDKTLKLWDVRSG-KCLKTLKGHSNYVFCCNFNPPSNLIISGSFDET
C3XVT5.1 SADMTIKLWDF-QTFENIKTMHGHDHNVSSVHFMPNGDFLISASRDKT
C3XVT5.1 SADMTIKLWDF-QTFENIKTMHGHDHNVSSVHFMPNGDFLISASRDKT
C3XVT5.1 SADMTIKLWDF-QTFENIKTMHGHDHNVSSVHFMPNGDFLISASRDKT
Q803D2.3 SADMTIKLWDF-QGFECIRTMHGHDHNVSSVAIMPNGDHIVSASRDKT
Q803D2.3 SADMTIKLWDF-QGFECIRTMHGHDHNVSSVAIMPNGDHIVSASRDKT
Q803D2.3 SADMTIKLWDF-QGFECIRTMHGHDHNVSSVAIMPNGDHIVSASRDKT
Q5RE95.1 SDDKTLKLWDMRSG-KCLKTLKGHSNYVFCCNFNPPSNLIISGSFDET
```

Simple integration of information

- Force amino acid choice (constraint) at conserved positions.

Learning CFNs from data

Boltzman distribution connects probability and cost/energy

$$P(X) \propto e^{-W(X)}$$

Learning CFNs from data

Boltzman distribution connects probability and cost/energy

$$P(X) \propto e^{-W(X)}$$

From CFN to probabilities and back

- After e^{-x} transform, a CFN defines a probability distribution (MRF)

Learning CFNs from data

Boltzman distribution connects probability and cost/energy

$$P(X) \propto e^{-W(X)}$$

From CFN to probabilities and back

- After e^{-x} transform, a CFN defines a probability distribution (MRF)
- Which can be learned from data using maximum penalized likelihood[1,6,10]

Learning CFNs from data

Boltzman distribution connects probability and cost/energy

$$P(X) \propto e^{-W(X)}$$

From CFN to probabilities and back

- After e^{-x} transform, a CFN defines a probability distribution (MRF)
- Which can be learned from data using maximum penalized likelihood[1,6,10]
- And transformed back into a CFN with a $-\log(x)$ transform

Learning in practice

- We start from a complete pairwise CFN with unknown cost functions

Learning in practice

- We start from a complete pairwise CFN with unknown cost functions
- We have a total of $d^2 \cdot \frac{n(n-1)}{2}$ parameters to learn

$w_{ij}(\cdot,\cdot)$

Learning in practice

- We start from a complete pairwise CFN with unknown cost functions $w_{ij}(\cdot, \cdot)$
- We have a total of $d^2 \cdot \frac{n(n-1)}{2}$ parameters to learn
- Let $\ell(D|w_{ij})$ be the log-probability of data D given the w_{ij}

Learning in practice

- We start from a complete pairwise CFN with unknown cost functions
- We have a total of $d^2 \cdot \frac{n(n-1)}{2}$ parameters to learn $w_{ij}(\cdot, \cdot)$
- Let $\ell(D|w_{ij})$ be the log-probability of data D given the w_{ij}

Maximize $\ell(D|w_{ij}) - \lambda \cdot ||w_{ij}||$ concave

Learning in practice

- We start from a complete pairwise CFN with unknown cost functions
- We have a total of $d^2 \cdot \frac{n(n-1)}{2}$ parameters to learn $w_{ij}(\cdot,\cdot)$
- Let $\ell(D|w_{ij})$ be the log-probability of data D given the w_{ij}

Maximize $\ell(D|w_{ij}) - \lambda \cdot ||w_{ij}||$ concave

Efficient L2 norm based implementation available[10]

- Uses conjugate gradient optimization
- fast C or very fast CUDA implementation
- n variables, d values, s samples: $O(d^2n^2 + dns)$ space.

Learning in practice

- We start from a complete pairwise CFN with unknown cost functions
- We have a total of $d^2 \cdot \frac{n(n-1)}{2}$ parameters to learn $w_{ij}(\cdot, \cdot)$
- Let $\ell(D|w_{ij})$ be the log-probability of data D given the w_{ij}

Maximize $\ell(D|w_{ij}) - \lambda \cdot ||w_{ij}||$ concave

Efficient L2 norm based implementation available[10]

- Uses conjugate gradient optimization
- fast C or very fast CUDA implementation
- n variables, d values, s samples: $O(d^2n^2 + dns)$ space.

600 variables, domain size 21 80, 000, 000 parameters, estimated in minutes

Symbolic vs. numerical AI

A counter-productive insulation of fields

- Symbolic (gradient-free) AI already reached super-human performances

Symbolic vs. numerical AI

A counter-productive insulation of fields

- Symbolic (gradient-free) AI already reached super-human performances
- Numerical (differentiable) AI: you certainly know! (Alpha Go/Zero)

Symbolic vs. numerical AI

A counter-productive insulation of fields

- Symbolic (gradient-free) AI already reached super-human performances
- Numerical (differentiable) AI: you certainly know! (Alpha Go/Zero)
- But reasoning/planning with Deep Nets? Not at this point.

Symbolic vs. numerical AI

A counter-productive insulation of fields

- Symbolic (gradient-free) AI already reached super-human performances
- Numerical (differentiable) AI: you certainly know! (Alpha Go/Zero)
- But reasoning/planning with Deep Nets? Not at this point.
- It's now possible to connect them and build hybrid AIs that reason and learn

Symbolic vs. numerical AI

A counter-productive insulation of fields

- Symbolic (gradient-free) AI already reached super-human performances
- Numerical (differentiable) AI: you certainly know! (Alpha Go/Zero)
- But reasoning/planning with Deep Nets? Not at this point.
- It's now possible to connect them and build hybrid AIs that reason and learn
- Graphical models look like a good place to start

Thanks

AI/toulbar2

S. de Givry (INRA)
G. Katsirelos (INRA)
M. Zytnicki (PhD, INRA)
D. Allouche (INRA)
H. Nguyen (PhD, INRA)
M. Cooper (IRIT, Toulouse)
J. Larrosa (UPC, Spain)
F. Heras (UPC, Spain)
M. Sanchez (Spain)
E. Rollon (UPC, Spain)
P. Meseguer (CSIC, Spain)
G. Verfaillie (ONERA, ret.)
JH. Lee (CU. Hong Kong)
C. Bessiere (LIMM, Montpellier)
JP. Métivier (GREYC, Caen)
S. Loudni (GREYC, Caen)
M. Fontaine (GREYC, Caen)

Protein Design

A. Voet (KU Leuven)
D. Simoncini (INSA, Toulouse)
S. Barbe (INSA, Toulouse)
S. Traoré (PhD, CEA)
C. Viricel (PhD)
PyRosetta (U. John Hopkins)
OSPREY (Duke U.)

References I

[1] Sivaraman Balakrishnan et al. “Learning generative models for protein fold families”. In: Proteins: Structure, Function, and Bioinformatics 79.4 (2011), pp. 1061–1078.

[2] Martin C Cooper et al. “Soft arc consistency revisited”. In: Artificial Intelligence 174.7 (2010), pp. 449–478.

[3] Daphne Koller and Nir Friedman. Probabilistic graphical models: principles and techniques. MIT press, 2009.

[4] Oliver Kullmann. “The Science of Brute Force”. In: Communications of the ACM (2017).

[5] Nina Narodytska et al. “Verifying properties of binarized deep neural networks”. In: Proc. of AAAI’18. 2018.

[6] Youngsuk Park et al. “Learning the Network Structure of Heterogeneous Data via Pairwise Exponential Markov Random Fields”. In: Artificial Intelligence and Statistics. 2017, pp. 1302–1310.

[7] Niles A Pierce and Erik Winfree. “Protein design is NP-hard.”. In: Protein Eng. 15.10 (Oct. 2002), pp. 779–82. ISSN: 0269-2139. URL: `http://www.ncbi.nlm.nih.gov/pubmed/12468711`.

[8] Daniela Röthlisberger et al. “Kemp elimination catalysts by computational enzyme design”. In: Nature 453.7192 (2008), p. 190.

[9] 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.

[10] Stefan Seemayer, Markus Gruber, and Johannes Söding. “CCMpred—fast and precise prediction of protein residue–residue contacts from correlated mutations”. In: Bioinformatics 30.21 (2014), pp. 3128–3130.

[11] David Simoncini et al. “Guaranteed Discrete Energy Optimization on Large Protein Design Problems”. In: Journal of Chemical Theory and Computation 11.12 (2015), pp. 5980–5989. DOI: `10.1021/acs.jctc.5b00594`.

References II

[12] Gilles Simonin et al. “Scheduling scientific experiments for comet exploration”. In: Constraints 20.1 (2015), pp. 77–99.

[13] Arnout RD Voet et al. “Computational design of a self-assembling symmetrical β-propeller protein”. In: Proceedings of the National Academy of Sciences 111.42 (2014), pp. 15102–15107.