Bilevel optimization and its bicriteria approximation in computational protein design

Samuel Buchet Marianne Defresne Simon de Givry Manon Ruffini Thomas Schiex

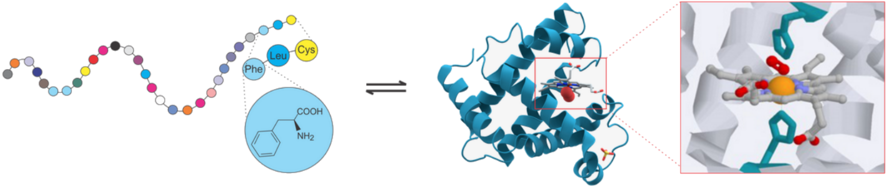
Mathématiques et Informatique Appliquées de Toulouse (MIAT)
INRAE, Castanet-Tolosan, France

Roadef, February 2023

Computational Protein Design (CPD)

Eco-friendly chemical/structural nano-agents

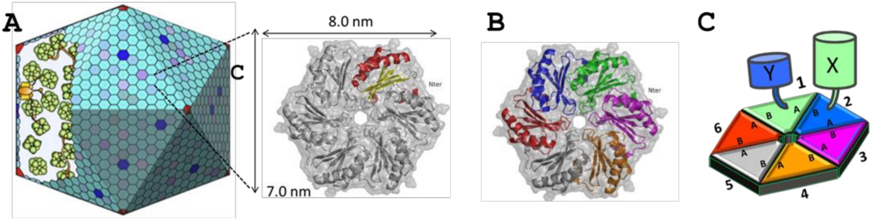
- New drugs for health (human, animals, plants)
- New catalysts (environment, recycling, biofuels, food and feed,...),
- New components for nanotechnologies
- Relying on inexpensive atomic level 3D-printers (bacterias, yeast, ...)



[0]Thanks to the Zhang's lab. for this image.

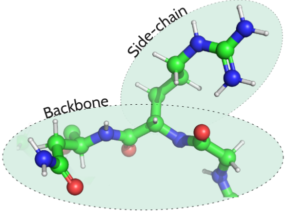
Computational Protein Design

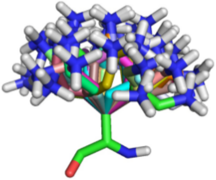
French ANR Project SPACEHex (2019-2023)



Spatial control of enzymes by redesign of A/B interfaces (sequence of $n = 58$ amino acids).

Computational Protein Design





Single-state design aims to minimize this energy function

$$E(\boldsymbol{s}, \chi) = E_{\varnothing} + \sum_{i=1}^{n} E_i(\boldsymbol{s}[i], \chi[i]) + \sum_{(i,j)\in[1,n]^2} E_{ij}(\boldsymbol{s}[i], \chi[i], \boldsymbol{s}[j], \chi[j])$$

NP-hard problem (20^n sequences, $\sim$20 3D-configurations per amino acid)

Computational Protein Design

Multi-state negative design

$$\min_{\boldsymbol{s},\chi} E^{+}(\boldsymbol{s},\chi) - E^{-}(\boldsymbol{s},\psi) \quad \text{where } \psi = \text{argmin}_{\psi} E^{-}(\boldsymbol{s},\psi)$$

Bilevel quadratic integer minimization problem ($\Sigma_2^P - complete$).

CPD as a Cost Function Network

Cost Function Network (CFN)

- $\mathbf{X} = (X_1, \ldots, X_n)$, list of variables with finite domains (d values max.)
- $\mathbf{F} = (F_1, \ldots, F_e)$, list of cost functions, each $F_{\mathbf{S}} \in \mathbf{F}$ involves a subset of variables $\mathbf{S} \subseteq \mathbf{X}$ and returns a cost in $\mathbb{R} \cup \{\infty\}$ to every assignment of $\mathbf{S}$.

The Weighted Constraint Satisfaction Problem[1](WCSP) is to find a complete assignment minimizing the sum of cost functions. It is NP-hard.

Each $E_{\varnothing}, E_i, E_{ij}$ term is a cost function on zero, one or two variables.

[1] [Larrosa and Schiex, AIJ 2004]

Linear relaxation of a WCSP

$$\min \sum_{F_{\boldsymbol{S}} \in \boldsymbol{F}, \tau \in \tau(\boldsymbol{S})} F_{\boldsymbol{S}}(\tau) \times y_\tau$$

$s.t.$

$$y_{\tau_1} = \sum_{\tau_2 \in \tau(\boldsymbol{S}_2), \tau_2[\boldsymbol{S}_1] = \tau_1} y_{\tau_2} \qquad \forall F_{\boldsymbol{S}_1}, F_{\boldsymbol{S}_2} \in \boldsymbol{F}, \boldsymbol{S}_1 \subset \boldsymbol{S}_2,$$

$$\tau_1 \in \tau(\boldsymbol{S}_1), |\boldsymbol{S}_1| \geq 1$$

$$\sum_{\tau \in \tau(\boldsymbol{S})} y_\tau = 1 \qquad \forall F_{\boldsymbol{S}} \in \boldsymbol{F}, |\boldsymbol{S}| \geq 1$$

CFN solving methods in TOULBAR2[2]

- Complete search methods
  - Variable Elimination (VE)
  - Depth-First Branch and Bound (DFBB)
  - Hybrid Best-First Branch and Bound (HBFS, //-HBFS)
  - **DFBB** or HBFS **with Tree Decomposition** (HBFS-BTD)
  - Unified Decomposition-Guided Variable Neighborhood Search (UDGVNS, //-UDGVNS)
- Approximate methods
  - **Soft Local Consistency** (EDAC,VAC,...)
  - Intensification/Diversification Walk Local Search (INCOP)
  - Partition Crossover Iterative Local Search (PILS)

[1] https://toulbar2.github.io/toulbar2/publications.html

Soft local consistency

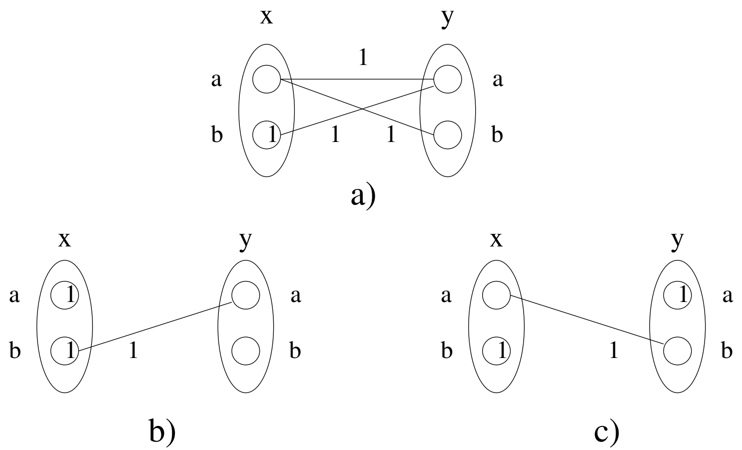


Figure: Three equivalent CFNs.

Complexity of soft local consistencies

Soft Local Consistency	Strenght	Complexity time	Complexity space	Polynomial Classes
NC*	weakest	$O(nd)$	$O(nd)$	-
AC*	$\vert$	$O(n^2d^2 + ed^3)$	$O(ed)$	-
DAC	$\not\vert$	$O(ed^2)$	$O(ed)$	*Trees*
FDAC*	$\vert$	$O(end^3)$	$O(ed)$	*Trees*
EDAC*	$\vert$	$O(ed^2 \max(nd, k))$	$O(ed)$	*Trees*
VAC_ϵ	$\downarrow$	$O(ed^2 k/\epsilon)$	$O(ed)$	Trees, Submodular func.
OSAC	strongest	$poly(ed + n)$	$poly(ed^2 + nd)$	Trees, Submodular func.

OSAC is the dual of the previous WCSP linear relaxation.[3]
In practice, EDAC* is used during search.

[3] [Cooper *et al*, AIJ 2010] https://miat.inrae.fr/degivry/web/Cooper10a.pdf

Bilevel CPD toy example


```
{"problem":{"name":"P1","mustbe":"<1000"},
"variables":{
"X1":["V0","V1","K2","K3"],
"X2":["V0","K1"],
"X3":["V0","K1","K2"]},
"functions":{
"F_X1":{"scope":["X1"],"costs":[2,3,2,0]},
"F_X2":{"scope":["X2"],"costs":[3,0]},
"F_X3":{"scope":["X3"],"costs":[5,2,0]},
"F_X1_X2":{"scope":["X1","X2"],"costs":[1,0,2,3,0,1,0,0]},
"F_X2_X3":{"scope":["X2","X3"],"costs":[0,0,0,0,1,2,0]}}}
```

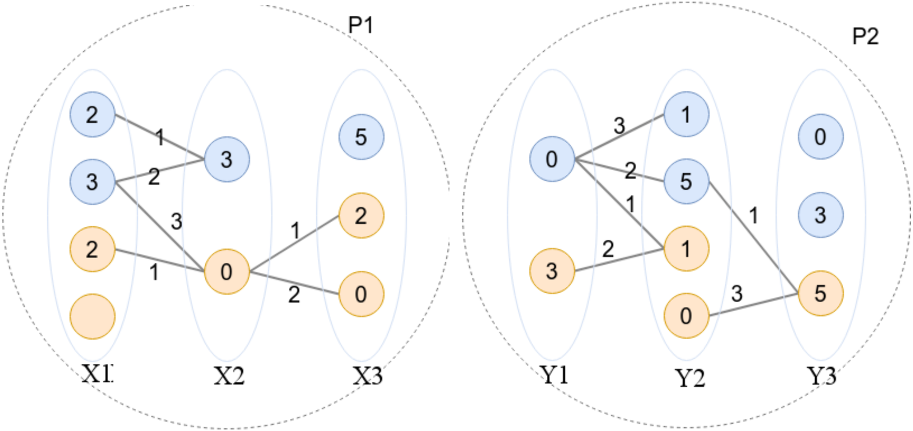
Bilevel CPD toy example


```
{"problem":{"name":"P2","mustbe":"<1000"},
"variables":{
"X1":["V0","V1","K2","K3"],
"X2":["V0","K1"],
"X3":["V0","K1","K2"],
"Y1":["V0","K1"],
"Y2":["V0","V1","K2","K3"],
"Y3":["V0","V1","K2"]},
"functions":{
"E_Y1":{"scope":["Y1"],"costs":[0,3]},
"E_Y2":{"scope":["Y2"],"costs":[1,5,1,0]},
"E_Y3":{"scope":["Y3"],"costs":[0,3,5]},
"E_Y1_Y2":{"scope":["Y1","Y2"],"costs":[3,2,1,0,0,0,2,0]},
"E_Y2_Y3":{"scope":["Y2","Y3"],"costs":[0,0,0,0,0,1,0,0,0,0,0,3]},
"same_seq_X1_Y1":{"scope":["X1","Y1"],"costs":[0,"inf",0,"inf","inf",0,"inf",0]},
"same_seq_X2_Y2":{"scope":["X2","Y2"],"costs":[0,0,"inf","inf","inf","inf",0,0]},
"same_seq_X3_Y3":{"scope":["X3","Y3"],"costs":[0,0,"inf","inf","inf",0,"inf","inf",0]}}}
```

Bilevel CPD toy example

```
import pytoulbar2 as tb2
# create restricted leader problem
cfn1 = tb2.CFN(1000)
cfn1.AddVariable('X1',range(4))
cfn1.AddVariable('X2',range(2))
cfn1.AddVariable('X3',range(3))
cfn1.AddFunction(['X1'], [ 2 , 3 , 2 , 0 ])
cfn1.AddFunction(['X2'], [ 3 , 0 ])
cfn1.AddFunction(['X3'], [ 5 , 2 , 0 ])
cfn1.AddFunction(['X1','X2'], [ 1 , 0 , 2 , 3 , 0 , 1 , 0 , 0 ])
cfn1.AddFunction(['X2','X3'], [ 0 , 0 , 0 , 0 , 1 , 2 , 0 ])
cfn1.Dump('problem1.cfn')
# create follower problem
cfn2 = tb2.CFN(1000)
cfn1.AddVariable('X1',range(4))
cfn1.AddVariable('X2',range(2))
cfn1.AddVariable('X3',range(3))
cfn2.AddVariable('Y1',range(2))
cfn2.AddVariable('Y2',range(4))
cfn2.AddVariable('Y3',range(3))
cfn2.AddFunction(['Y1'], [ 0 , 3 ])
cfn2.AddFunction(['Y2'], [ 1 , 5 , 1 , 0 ])
cfn2.AddFunction(['Y3'], [ 0 , 3 , 5 ])
cfn2.AddFunction(['Y1','Y2'], [ 3 , 2 , 1 , 0 , 0 , 0 , 2 , 0 ])
cfn2.AddFunction(['Y2','Y3'], [ 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 3 ])
cfn2.AddFunction(['X1','Y1'], [ 0 , 'inf' , 0 , 'inf' , 'inf' , 0 , 'inf' , 0 ])
cfn2.AddFunction(['X2','Y2'], [ 0 , 0 , 'inf' , 'inf' , 'inf' , 'inf' , 0 , 0 ])
cfn2.AddFunction(['X3','Y3'], [ 0 , 0 , 'inf' , 'inf' , 'inf' , 0 , 'inf' , 'inf' ,0])
cfn2.Dump('problem2.cfn')
# solve by external command: toulbar2 -bilevel problem1.cfn problem2.cfn
```

Bilevel CPD toy example



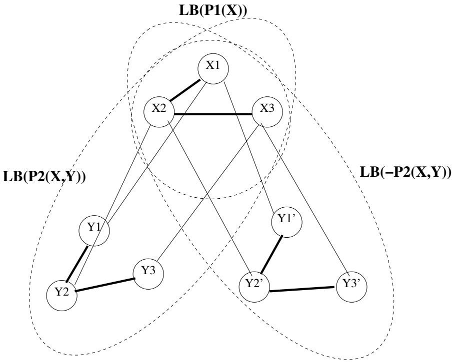
$$\min_{\boldsymbol{X},\boldsymbol{Y}} P1(\boldsymbol{X}) - P2(\boldsymbol{X},\boldsymbol{Y}) \quad \text{where } \boldsymbol{Y} = \text{argmin}_{\boldsymbol{Y}} P2(\boldsymbol{X},\boldsymbol{Y})$$

Bilevel CPD toy example

```
//P1(X)-P2(X,Y)   X        P1(X)        Y              P2(X,Y)
11  X1=V1  X2=K1  X3=V0  11  Y1=V0  Y2=K3  Y3=V0  0
9  X1=V1  X2=V0  X3=V0  13  Y1=V0  Y2=V0  Y3=V0  4
7  X1=V0  X2=V0  X3=V0  11  Y1=V0  Y2=V0  Y3=V0  4
7  X1=V0  X2=K1  X3=V0  7  Y1=V0  Y2=K3  Y3=V0  0
6  X1=K2  X2=V0  X3=V0  10  Y1=K1  Y2=V0  Y3=V0  4
5  X1=K2  X2=K1  X3=V0  8  Y1=K1  Y2=K3  Y3=V0  3
4  X1=K3  X2=V0  X3=V0  8  Y1=K1  Y2=V0  Y3=V0  4
...
-3  X1=V0  X2=V0  X3=K2  6  Y1=V0  Y2=V0  Y3=K2  9
-3  X1=V0  X2=K1  X3=K2  4  Y1=V0  Y2=K2  Y3=K2  7
-4  X1=K3  X2=V0  X3=K1  5  Y1=K1  Y2=V0  Y3=K2  9
-4  X1=K2  X2=V0  X3=K2  5  Y1=K1  Y2=V0  Y3=K2  9
-5  X1=K2  X2=K1  X3=K1  6  Y1=K1  Y2=K3  Y3=K2  11
-6  X1=K3  X2=V0  X3=K2  3  Y1=K1  Y2=V0  Y3=K2  9
-6  X1=K2  X2=K1  X3=K2  5  Y1=K1  Y2=K3  Y3=K2  11
-8  X1=K3  X2=K1  X3=K1  3  Y1=K1  Y2=K3  Y3=K2  11
-9  X1=K3  X2=K1  X3=K2  2  Y1=K1  Y2=K3  Y3=K2  11
```

solved in 17 nodes (search space of 64 feasible assignments).

Branch and Bound with Tree Decomposition



Exploiting tree decomposition and soft local consistency[4]

[4] [Schiex *et al*, AAAI 2006] `https://miat.inrae.fr/degivry/web/Schiex06a.pdf`

Bilevel random problem solving

Experiments made on a Linux Intel i7-4600U CPU running at 3.3GHz max and 1TB, using only one core.

Comparison on a randomly-generated problem:

- CFN leader has 8 variables with domain size of 5 and 28 binary cost functions, follower has 4 extra variables with domain size of 5 and 66 binary cost functions
- 01LP HPR has 2411 cols and 953/1616 rows

	Mix [5]/cplex v12.10	toulbar2
optimum	**-1947**	**-1947**
time (sec.)	4608	**5**
nodes	**131817**	642350

[5] [Fischetti *et al*, OR 2017] `https://msinnl.github.io/pages/bilevel.html`

Bicriteria problem formulation

Approximation:
Coarse-grain energy model learnt on the sequence only (without χ,ψ).

f being the energy on the desired backbone of a protein sequence, and g on the undesired backbone:

$$\min_{\boldsymbol{s}} f(\boldsymbol{s}) = E^{+}(\boldsymbol{s})$$
$$\max_{\boldsymbol{s}} g(\boldsymbol{s}) = E^{-}(\boldsymbol{s})$$

Dominance: x^1 dominates (is better than) x^2 iif $f(x^1) \leq f(x^2)$ and $g(x^1) \geq g(x^2)$ with a strict inequality for f or g

Pareto front: set of all non dominated solutions (optimal compromises between f and g)

Scalarization technique

$$\text{solve} \max_{x \in \mathcal{X}} \lambda_1 * f(x) + \lambda_2 * g(x)$$
$$\lambda_1, \lambda_2 \in \mathbb{R}$$

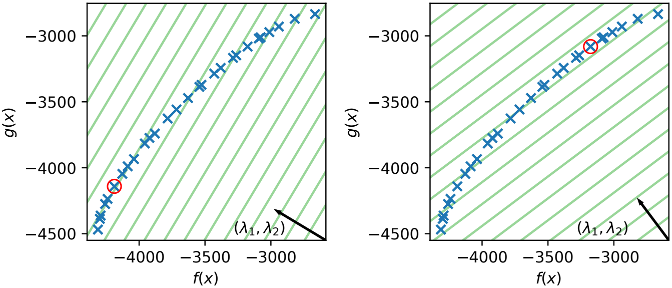
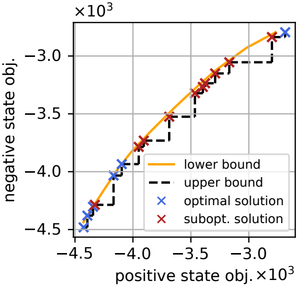


Figure: Optimization with two different pairs of weights.

Properties: set of supported pareto solutions (convex hull) by enumeration of weights [Aneja & Nair]
`https://doi.org/10.1287/mnsc.25.1.73`

Implementation in ToulBar2:

- scalarization and weight enumeration [6]
- computation of a lower bound curve



Ongoing work: embedding g as a constraint

$$\min_{x \in \mathcal{X}} f(x)$$
$$s.t. \quad g(x) \geq q$$

[6] https://toulbar2.github.io/toulbar2/examples/tuto_biwlsp.html