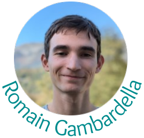
Efficient Neuro-symbolic Learning of Constraints and Criteria from Sudoku to new functional molecules

(Defresne, Gambardella, et al. 2026)

Thomas Schiex



INRAE

ANITI

Université de Toulouse

Human reasoning and scientific discovery

Inductive and deductive reasoning

- From observations (solutions) we construct a theory ($F = m\gamma$)
- We then use the theory to make predictions and design objects
- Until the theory is proven to be incorrect

Sudoku grid with solution

Protein structure with its sequence

The theory is written as a pairwise Graphical Model (a Cost Function Network)

Human reasoning and scientific discovery

Inductive and deductive reasoning

- From observations (solutions) we construct a theory ($F = m\gamma$)
- We then use the theory to make predictions and design objects
- Until the theory is proven to be incorrect

Sudoku grid with solution

Protein structure with its sequence

The theory is written as a pairwise Graphical Model (a Cost Function Network)

Human reasoning and scientific discovery

Inductive and deductive reasoning

- From observations (solutions) we construct a theory ($F = m\gamma$)
- We then use the theory to make predictions and design objects
- Until the theory is proven to be incorrect

Sudoku grid with solution

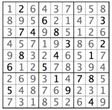
Protein structure with its sequence

The theory is written as a pairwise Graphical Model (a Cost Function Network)

Human reasoning and scientific discovery

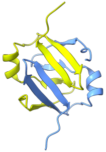
Inductive and deductive reasoning

- From observations (solutions) we construct a theory ($F = m\gamma$)
- We then use the theory to make predictions and design objects
- Until the theory is proven to be incorrect



1	**2**	6	4	3	7	9	5	8
8	9	5	**6**	2	1	4	7	**3**
3	**7**	**4**	9	**8**	5	1	2	6
4	5	7	1	9	**3**	8	6	**2**
9	**8**	3	2	**4**	6	5	**1**	7
6	1	2	**5**	7	8	3	9	4
2	6	9	3	**1**	4	**7**	**8**	5
5	4	8	7	6	**9**	2	3	1
7	3	1	8	5	2	6	**4**	9

Sudoku grid with solution



Protein structure with its sequence

The theory is written as a pairwise Graphical Model (a Cost Function Network)

Additive discrete graphical model

CFN/WCSP

Reminder

- A set X of variables n variables
- Variable x_i has domain D_i max. size d
- a set of cost functions $c_{ij} : D_i \times D_j \to \mathbb{R}^+ \cup \{\infty\}$

Variables and parameters/costs

- The cost $C(t)$ of an assignment t is the sum of all cost functions on t
- The cost is linear in the parameters (costs in CF tables)
- It defines a probability distribution: $P(t) \propto \exp(-C(t))$ Markov Random Fields

Additive discrete graphical model

Reminder

- A set X of variables — n variables
- Variable x_i has domain D_i — max. size d
- a set of cost functions — $c_{ij} : D_i \times D_j \rightarrow \mathbb{R}^+ \cup \{\infty\}$

Variables and parameters/costs

- The cost $C(t)$ of an assignment t is the sum of all cost functions on t
- The cost is linear in the parameters (costs in CF tables)
- It defines a probability distribution: $P(t) \propto \exp(-C(t))$ — Markov Random Fields

Structured output prediction (SOP) with a CFN model

	2							
			6					3
	7	4		8				
					3			2
	8			4			1	
6			5					
				1		7	8	
5					9			
							4	

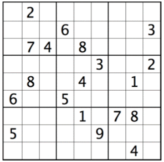
ω

Structured output prediction (SOP) with a CFN model

	2							
			6					3
	7	4		8				
					3			2
	8			4			1	
6			5					
				1		7	8	
5					9			
							4	

$\omega \longrightarrow$ Neural net

Structured output prediction (SOP) with a CFN model

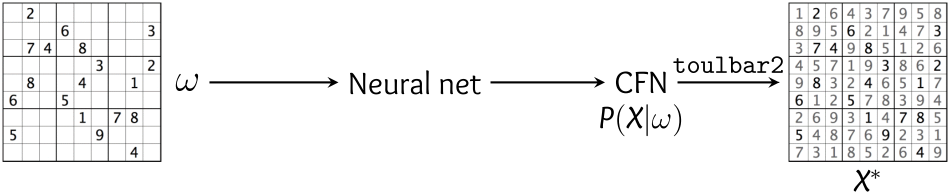


	2							
			6					3
	7	4		8				
					3			2
	8			4			1	
6			5					
				1		7	8	
5					9			
							4	

ω ⟶ Neural net ⟶ CFN

$P(X|\omega)$

Structured output prediction (SOP) with a CFN model



	2							
			6					3
	7	4		8				
					3			2
	8			4			1	
6			5					
				1		7	8	
5					9			
							4	

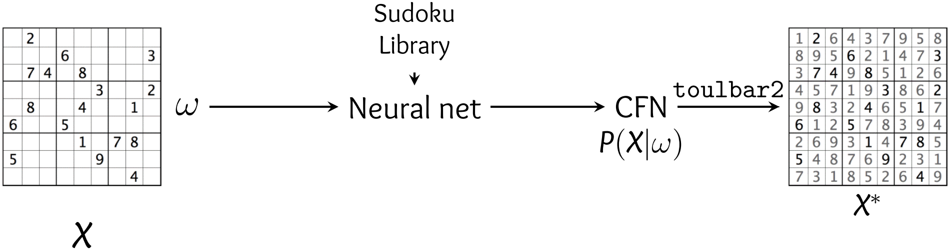
$$\omega \longrightarrow \text{Neural net} \longrightarrow \text{CFN} \xrightarrow{\texttt{toulbar2}}$$

$P(X|\omega)$

1	**2**	6	4	3	7	9	5	8
8	9	5	**6**	2	1	4	7	**3**
3	**7**	**4**	9	**8**	5	1	2	6
4	5	7	1	9	**3**	8	6	**2**
9	**8**	3	2	**4**	6	5	**1**	7
6	1	2	**5**	7	8	3	9	4
2	6	9	3	**1**	4	**7**	**8**	5
5	4	8	7	6	**9**	2	3	1
7	3	1	8	5	2	6	**4**	9

X^*

Structured output prediction (SOP) with a CFN model



	2							
			6					3
	7	4		8				
					3			2
	8			4			1	
6			5					
				1		7	8	
5					9			
							4	

X

$\omega \longrightarrow$ Neural net $\longrightarrow$ CFN $\xrightarrow{\texttt{toulbar2}}$

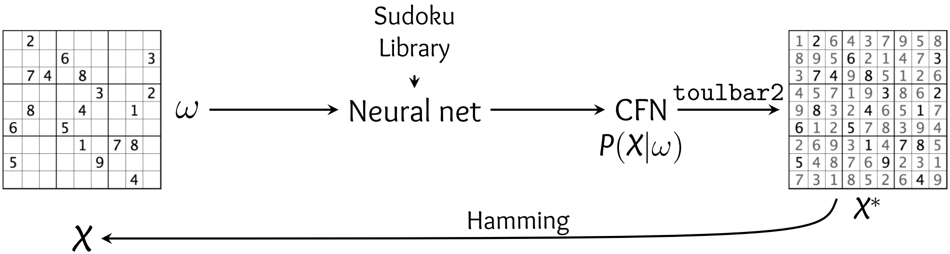
Sudoku Library → Neural net

CFN $P(X|\omega)$

1	2	6	4	3	7	9	5	8
8	9	5	6	2	1	4	7	3
3	7	4	9	8	5	1	2	6
4	5	7	1	9	3	8	6	2
9	8	3	2	4	6	5	1	7
6	1	2	5	7	8	3	9	4
2	6	9	3	1	4	7	8	5
5	4	8	7	6	9	2	3	1
7	3	1	8	5	2	6	4	9

X^*

Structured output prediction (SOP) with a CFN model



	2							
			6					3
	7	4		8				
					3			2
	8			4			1	
6			5					
				1		7	8	
5					9			
							4	

ω

Sudoku
Library

Neural net

CFN
$P(X|\omega)$

`toulbar2`

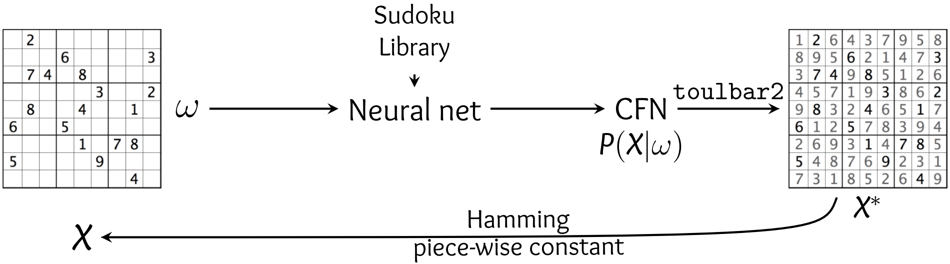
1	2	6	4	3	7	9	5	8
8	9	5	6	2	1	4	7	3
3	7	4	9	8	5	1	2	6
4	5	7	1	9	3	8	6	2
9	8	3	2	4	6	5	1	7
6	1	2	5	7	8	3	9	4
2	6	9	3	1	4	7	8	5
5	4	8	7	6	9	2	3	1
7	3	1	8	5	2	6	4	9

X^*

Hamming

X

Structured output prediction (SOP) with a CFN model



	2							
			6					3
	7	4		8				
					3			2
	8			4			1	
6			5					
				1		7	8	
5					9			
							4	

ω → Neural net → CFN $P(X|\omega)$ $\xrightarrow{\texttt{toulbar2}}$

Sudoku Library

1	2	6	4	3	7	9	5	8
8	9	5	6	2	1	4	7	3
3	7	4	9	8	5	1	2	6
4	5	7	1	9	3	8	6	2
9	8	3	2	4	6	5	1	7
6	1	2	5	7	8	3	9	4
2	6	9	3	1	4	7	8	5
5	4	8	7	6	9	2	3	1
7	3	1	8	5	2	6	4	9

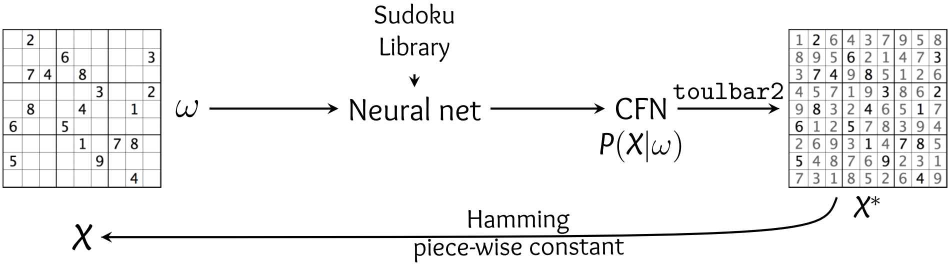
X^*

X ← Hamming piece-wise constant

Issues

- Gradients either zero or undefined
- Requires to repeatedly solve random NP-hard instances

Structured output prediction (SOP) with a CFN model



	2							
			6					3
	7	4		8				
					3			2
	8			4			1	
6			5					
				1		7	8	
5					9			
							4	

ω → Sudoku Library → Neural net → CFN $P(X|\omega)$ $\xrightarrow{\texttt{toulbar2}}$

1	**2**	6	4	3	7	9	5	8
8	9	5	**6**	2	1	4	7	**3**
3	**7**	**4**	9	**8**	5	1	2	6
4	5	7	1	9	**3**	8	6	**2**
9	**8**	3	2	**4**	6	5	**1**	7
6	1	2	**5**	7	8	3	9	4
2	6	9	3	**1**	4	**7**	**8**	5
5	4	8	7	6	**9**	2	3	1
7	3	1	8	5	2	6	**4**	9

X^*

X ← Hamming piece-wise constant

Natural choice: the negative loglikelihood

#P-hard

- Use Besag's pseudo-loglikelihood (1975, efficient)
- Kicks the solver out of the training loop (scalable training)

(Besag 1975)

Besag's Pseudo-Loglikelihood

- The Pseudo-LL masks each variable successively in the solution X (Besag 1975)

$$NLL = -\sum_{x} \log(P(x)) \qquad NPLL = -\sum_{x} \sum_{x_i} \log(P(x_i|x_{-i}))$$

- Nice asymptotic properties statistically consistent
- The NPLL is a “Fenchel-Young” loss (Defresne, Gambardella, et al. 2026; Blondel et al. 2020)
- Does not work in practice (high costs) (Montanari et al. 2009)

Vanishing gradient issue

- If enough constraints have been learned to force the observed value of X_i in the context of x_{-i}, it becomes impossible to learn other constraints.
- Related to the idempotence of logical information

Besag's Pseudo-Loglikelihood

- The Pseudo-LL masks each variable successively in the solution X (Besag 1975)

$$NLL = -\sum_{x} \log(P(x)) \qquad NPLL = -\sum_{x} \sum_{x_i} \log(P(x_i|x_{-i}))$$

- Nice asymptotic properties statistically consistent
- The NPLL is a “Fenchel-Young” loss (Defresne, Gambardella, et al. 2026; Blondel et al. 2020)
- Does not work in practice (high costs) (Montanari et al. 2009)

Vanishing gradient issue

- If enough constraints have been learned to force the observed value of X_i in the context of x_{-i}, it becomes impossible to learn other constraints.
- Related to the idempotence of logical information

Besag's Pseudo-Loglikelihood

- The Pseudo-LL masks each variable successively in the solution X (Besag 1975)

$$NLL = -\sum_{x} \log(P(x)) \qquad NPLL = -\sum_{x} \sum_{x_i} \log(P(x_i|x_{-i}))$$

- Nice asymptotic properties statistically consistent
- The NPLL is a “Fenchel-Young” loss (Defresne, Gambardella, et al. 2026; Blondel et al. 2020)
- Does not work in practice (high costs) (Montanari et al. 2009)

Vanishing gradient issue

- If enough constraints have been learned to force the observed value of X_i in the context of x_{-i}, it becomes impossible to learn other constraints.
- Related to the idempotence of logical information

Besag's Pseudo-Loglikelihood

- The Pseudo-LL masks each variable successively in the solution X (Besag 1975)

$$NLL = -\sum_{x} \log(P(x)) \qquad NPLL = -\sum_{x} \sum_{x_i} \log(P(x_i|x_{-i}))$$

- Nice asymptotic properties statistically consistent
- The NPLL is a “Fenchel-Young” loss (Defresne, Gambardella, et al. 2026; Blondel et al. 2020)
- Does not work in practice (high costs) (Montanari et al. 2009)

Vanishing gradient issue

- If enough constraints have been learned to force the observed value of X_i in the context of x_{-i}, it becomes impossible to learn other constraints.
- Related to the idempotence of logical information

The Emmental-PLL

Removing vanishing gradients

We ignore a random fraction of the neighbors/functions when computing $P(x_i|x_{-i})$

Easy, hard and extremely hard Sudoku grids

Type	Approach	Acc.	#hints	Train set	Param.	Train time (h)
DL	RRN (Palm et al. 2018)	96.6%	17	180,000	200k	> 50
	Rec. Trans. (Yang et al. 2023)	96.7%	17	180,000	211k	> 50
	Rec. Trans.	76.2–78.2%	17	9,000	-	1.8
	DDPM (Ye et al. 2025)	99.2–100%	33.8	100,000	6M	13.6
	DDPM	0.2%	17	-	-	-
	HRM (G. Wang et al. 2025)	55%	24.8	1,000 × 1,001	27M	>10
Relax+DL	SATNet (P.-W. Wang et al. 2019)	95.1–99.8%	36.2	9,000	600k	2.9
	SATNet	86.1–86.2%	17	-	-	-
CO	(Bessiere et al. 2023)	**100%**	-	200	-	0.01
CO + ML	(Brouard et al. 2020)	**100%**	17	9,000	-	1.5
CO+DL	Hinge (Defresne, Barbe, et al. 2023)	**100%**	17	1,000	180k	>50
	E-PLL (ours)	**100%**	17	**100**	22k	0.05
	E-PLL (HRM dataset)	**100%**	24.8	**10** × 100	22k	0.04

Learning to play Many-Solutions Sudokus

Sudokus have only one solution (single target for DL)

- Existing DL architectures fail on many-solutions Sudokus (Nandwani et al. 2021)
- Corrected using a Reinforcement learning approach
- Training set with 5 solutions per instance
- Ability to generate additional solutions

Our architecture directly learns how to solve many-solutions Sudokus

Learning to play Many-Solutions Sudokus

Sudokus have only one solution (single target for DL)

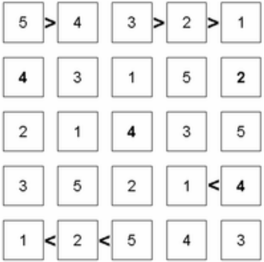
- Existing DL architectures fail on many-solutions Sudokus (Nandwani et al. 2021)
- Corrected using a Reinforcement learning approach
- Training set with 5 solutions per instance
- Ability to generate additional solutions

Our architecture directly learns how to solve many-solutions Sudokus

Learning to play Futoshiki

Sudoku is easy, only one type of constraint

- Our architecture directly learns how to play Futoshiki
- Includes both difference and inequality constraints
- Perfect solving, expected constraints learned



5 > 4 3 > 2 > 1

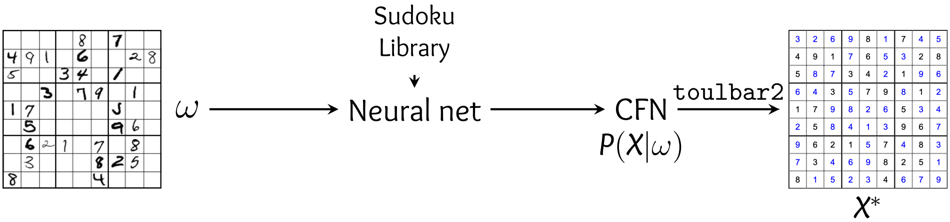
4 3 1 5 2

2 1 4 3 5

3 5 2 1 < 4

1 < 2 < 5 4 3

Learning to play Visual Sudoku

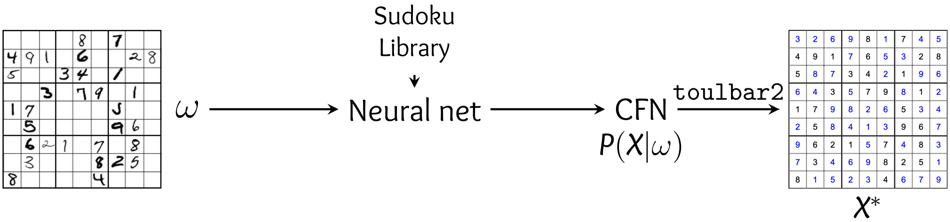


3	2	6	9	8	1	7	4	5
4	9	1	7	6	5	3	2	8
5	8	7	3	4	2	1	9	6
6	4	3	5	7	9	8	1	2
1	7	9	8	2	6	5	3	4
2	5	8	4	1	3	9	6	7
9	6	2	1	5	7	4	8	3
7	3	4	6	9	8	2	5	1
8	1	5	2	3	4	6	7	9

Simultaneously learns to recognize digits and to play the Sudoku

SATNet	Theoretical (no corrections)	Hybrid
63.2 %	74.2%	$94.1 \pm 0.8\%$

Learning to play Visual Sudoku

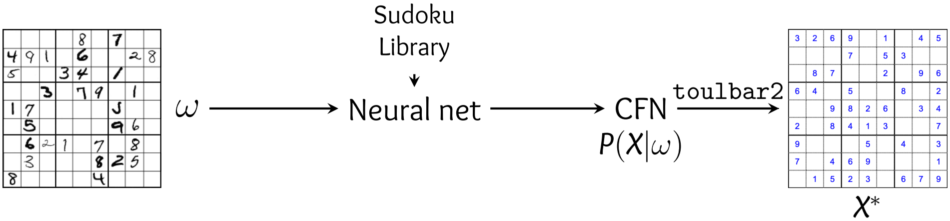


3	2	6	9	8	1	7	4	5
4	9	1	7	6	5	3	2	8
5	8	7	3	4	2	1	9	6
6	4	3	5	7	9	8	1	2
1	7	9	8	2	6	5	3	4
2	5	8	4	1	3	9	6	7
9	6	2	1	5	7	4	8	3
7	3	4	6	9	8	2	5	1
8	1	5	2	3	4	6	7	9

Simultaneously learns to recognize digits and to play the Sudoku

SATNet	Theoretical (no corrections)	Hybrid
63.2 %	74.2%	**94.1** $\pm$ 0.8%

Reading numbers without cheating (grounding)



ω → Neural net → CFN $P(X|\omega)$ $\xrightarrow{\texttt{toulbar2}}$ X^*

Sudoku Library ↓ Neural net

3	2	6	9		1		4	5
			7		5	3		
	8	7			2		9	6
6	4		5			8		2
		9	8	2	6		3	4
2		8	4	1	3			7
9				5		4		3
7		4	6	9				1
	1	5	2	3		6	7	9

X^*

Grounding issue: a nasty form of data leakage

(Chang et al. 2020)

- The training set contains images and associated decoded digits (hints).
- Solved using a complex architecture (InfoGAN+clustering+Distillation) (Topan et al. 2021)
- E-PLL: missing data, imputation by optimisation
- Much longer training (a few hours)

Existing approaches

Approach	MNIST accuracy	Percep.	Solved	Training (h)
SATNet	0.0 %	0.0 %	0.0 %	-
Rec. Trans (Yang et al. 2023)	99.4%	74.8%	75.6%	5.1
NeSy. Prog. (Li et al. 2023)	99.6–99.7%	90.7–93.1%	92.2–94.4%	4.7
E-PLL (Ours)	98.8%	72.9%	93.4%	3.2

Relation with Decision-focused learning

From DFL to Structured Output Prediction

- Data: pairs $\langle \omega, c \rangle$ where c define the criterion parameters
- Assumes constraints are known
- we can compute a SOP data-set (ω, X^*)
- Aim: minimize regret (difference in real cost of the predicted and optimal solution)

Relation with Decision-focused learning

From DFL to Structured Output Prediction

- Data: pairs $\langle \omega, c \rangle$ where c define the criterion parameters
- Assumes constraints are known
- we can compute a SOP data-set (ω, X^*)
- Aim: minimize regret (difference in real cost of the predicted and optimal solution)

Relation with Decision-focused learning

From DFL to Structured Output Prediction

- Data: pairs $\langle \omega, c \rangle$ where c define the criterion parameters
- Assumes constraints are known
- we can compute a SOP data-set (ω, X^*)
- Aim: minimize regret (difference in real cost of the predicted and optimal solution)

Relation with Decision-focused learning

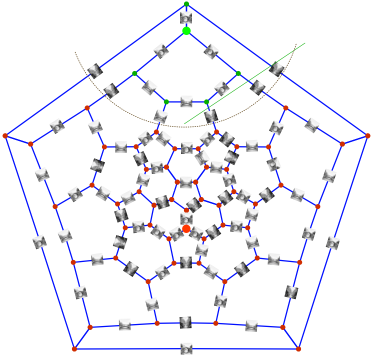
From DFL to Structured Output Prediction

- Data: pairs $\langle \omega, c \rangle$ where c define the criterion parameters
- Assumes constraints are known
- we can compute a SOP data-set (ω, X^*)
- Aim: minimize regret (difference in real cost of the predicted and optimal solution)

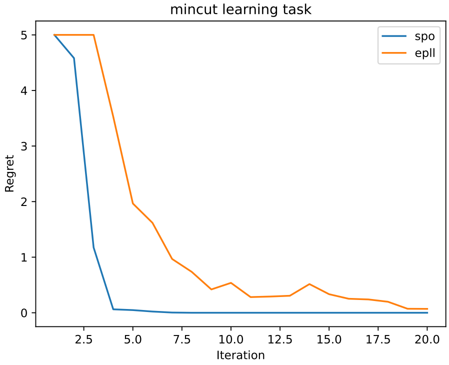
MinCut and MaxCut solving

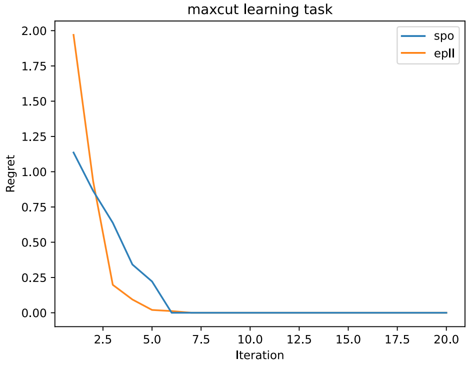
The Min/Max-Cut problems

- one Boolean variable per vertex (cut side)
- per-edge difference (Min-Cut) or equality (Max-Cut) function scaled by a predicted scalar $c(\omega)$
- bridge images with Gaussian noise (std-dev 10)

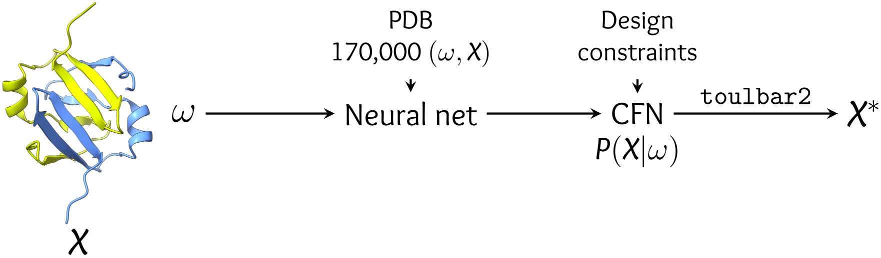


MinCut, MaxCut, Regret and SPO+ (Elmachtoub et al. 2022)





Learning to design proteins: Effie



Neural architecture

- More complex SE(3)-equivariant neural network
- Relies on Gated MLPs (post-transformer architecture)

(Liu et al. 2021)

Optimizing a complete protein sequence

Full redesign of large proteins in the test set

- Guaranteed `toulbar2` solution expensive
- Using LR-BCD SDP solver instead (Durante et al. 2022)

Outperforms all-atoms XIXth-century physics

- Metric: **Native Sequence Recovery** rate (NSR)

Approach	Rosetta	Effie
NSR	17.9%	32.8%

Optimizing a complete protein sequence

Full redesign of large proteins in the test set

- Guaranteed `toulbar2` solution expensive
- Using LR-BCD SDP solver instead (Durante et al. 2022)

Outperforms all-atoms XIXth-century physics

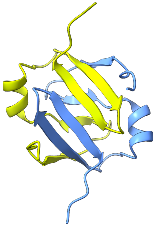
- Metric: **Native Sequence Recovery** rate (NSR)

Approach	Rosetta	Effie
NSR	17.9%	32.8%

Effie vs. the model-free ProteinMPNN

(Dauparas et al. 2022)

GPT-style



PDB

$\downarrow$

$\omega \longrightarrow$ Autoregressive NN $\longrightarrow X_1, X_2, \ldots$

$P(X_i|X_{i-1}, \ldots, X_1, \omega)$

Pros and cons

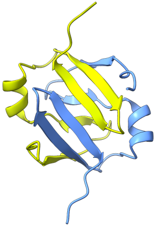
- $P(X_1, \ldots, X_n) = P(X_1)P(X_2|X_1) \cdots P(X_n|X_1, \ldots X_{n-1})$ is a mathematical identity
- But an easily broken one (*e.g.*, low temperature sampling)
- Heuristic score instead of NP-hard solving
- Limited control for design constraints, hard to “reason forward”

	ProteinMPNN	Effie
NSR	45.9%	48.4%

Effie vs. the model-free ProteinMPNN

(Dauparas et al. 2022)

GPT-style



PDB

↓

$\omega \longrightarrow$ Autoregressive NN $\longrightarrow X_1, X_2, \ldots$

$P(X_i|X_{i-1}, \ldots, X_1, \omega)$

Pros and cons

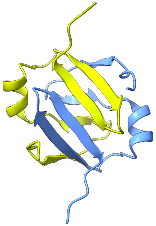
- $P(X_1, \ldots, X_n) = P(X_1)P(X_2|X_1) \cdots P(X_n|X_1, \ldots X_{n-1})$ is a mathematical identity
- But an easily broken one (*e.g.*, low temperature sampling)
- Heuristic score instead of NP-hard solving
- Limited control for design constraints, hard to “reason forward”

	ProteinMPNN	Effie
NSR	45.9%	48.4%

Effie vs. the model-free ProteinMPNN

(Dauparas et al. 2022)

GPT-style



PDB

$$\omega \longrightarrow \overset{\overset{\text{PDB}}{\downarrow}}{\underset{P(X_i|X_{i-1},\ldots,X_1,\omega)}{\text{Autoregressive NN}}} \longrightarrow X_1, X_2, \ldots$$

Pros and cons

- $P(X_1,\ldots,X_n) = P(X_1)P(X_2|X_1)\cdots P(X_n|X_1,\ldots X_{n-1})$ is a mathematical identity
- But an easily broken one (*e.g.*, low temperature sampling)
- Heuristic score instead of NP-hard solving
- Limited control for design constraints, hard to “reason forward”

	ProteinMPNN	Effie
NSR	45.9%	**48.4%**

Predicting SARS-CoV2 variants

Enumerate CoViD variants with a bounded number of mutations

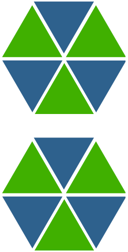
- Uses only the initial March 2020 RBD-ACE2 structure + Effie/toulbar2
- Relies on a global constraint to bound mutations (Ruffini et al. 2021)
- Predicts all the first SARS-CoV2 VoCs ($\alpha, \beta, \gamma, \delta, \kappa, \iota, \lambda$ and μ)
- In a few seconds, on one CPU-thread.

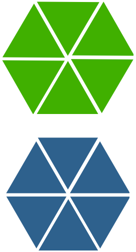
Not achievable by pure autoregressive models (ProteinMPNN).
Previously shown to predict contagious antibody-resistant variants (Colom et al. 2024).

Design of an enzyme organizing platform

Design of an heteromeric hexamer

- Design  and  that self-assemble as  but not as  or 
- Physics+logic: requires bi-level optimization (NP^{NP}-complete) (Vucinic et al. 2020)
- Compare Effie+tb2 (NP-complete) with ProteinMPNN, bi-criteria (Buchet et al. 2024)

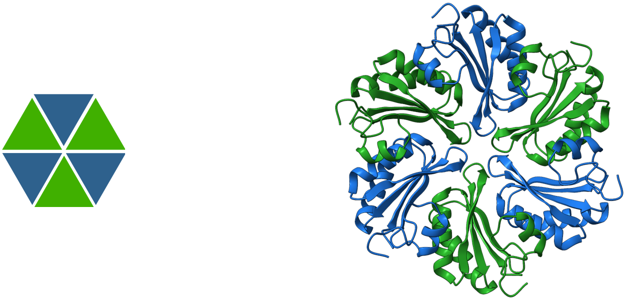




Design of an enzyme organizing platform

Design of an heteromeric hexamer

- Design ▲ and ▲ that self-assemble as ⬢ but not as ⬢ or ⬢
- Physics+logic: requires bi-level optimization (NP^{NP}-complete) (Vucinic et al. 2020)
- Compare Effie+tb2 (NP-complete) with ProteinMPNN, bi-criteria (Buchet et al. 2024)

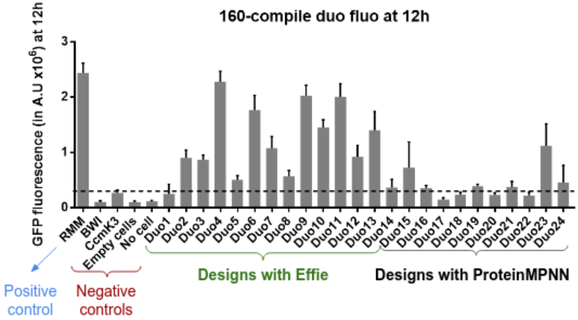


How often is ⬢ better than ⬢?

Scoring →	Effie	PMPNN
Effie	100 %	99.5 %
PMPNN	3.0 %	82.6 %

How often is ⬢ better than ⬢?

Scoring →	Effie	PMPNN
Effie	100 %	99.5 %
PMPNN	3.0 %	82.6 %



Conclusions

A Neural Net, a GM, and a discrete optimizer in a NeSy autoencoder

- A NeSy Generative AI that benefits from each component
- Neural Network: ideal to extract a representation of $P(X|\omega)$ from raw inputs
- Represented as a GM in a fully explorable and controllable latent layer
- Using decoding by discrete reasoning (toulbar2) that accepts side constraints
- All this with scalable training

Conclusions

A Neural Net, a GM, and a discrete optimizer in a NeSy autoencoder

- A NeSy Generative AI that benefits from each component
- Neural Network: ideal to extract a representation of $P(X|\omega)$ from raw inputs
- Represented as a GM in a fully explorable and controllable latent layer
- Using decoding by discrete reasoning (toulbar2) that accepts side constraints
- All this with scalable training

Conclusions

A Neural Net, a GM, and a discrete optimizer in a NeSy autoencoder

- A NeSy Generative AI that benefits from each component
- Neural Network: ideal to extract a representation of $P(X|\omega)$ from raw inputs
- Represented as a GM in a fully explorable and controllable latent layer
- Using decoding by discrete reasoning (toulbar2) that accepts side constraints
- All this with scalable training

Conclusions

A Neural Net, a GM, and a discrete optimizer in a NeSy autoencoder

- A NeSy Generative AI that benefits from each component
- Neural Network: ideal to extract a representation of $P(X|\omega)$ from raw inputs
- Represented as a GM in a fully explorable and controllable latent layer
- Using decoding by discrete reasoning (toulbar2) that accepts side constraints
- All this with scalable training

Conclusions

A Neural Net, a GM, and a discrete optimizer in a NeSy autoencoder

- A NeSy Generative AI that benefits from each component
- Neural Network: ideal to extract a representation of $P(X|\omega)$ from raw inputs
- Represented as a GM in a fully explorable and controllable latent layer
- Using decoding by discrete reasoning (toulbar2) that accepts side constraints
- All this with scalable training

Acknowledgments

AI/toulbar2

S. de Givry (INRA)
G. Katsirelos (INRA)
M. Zytnicki (PhD, INRA)
D. Allouche (INRA)
M. Ruffini (INRA)
V. Durante (ANITI, PhD)
H. Nguyen (PhD, INRA)
C. Brouard (ML, INRA)
S. Buchet (INRAE/ANITI)
P. Montalbano (ANITI, PhD)
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),...

DL/Protein Design

A. Voet (KU Leuven)
A. Olichon (INSERM)
D. Simoncini (UFT, Toulouse)
S. Barbe (INSA, Toulouse)
M. Defresne (INRAE, PhD)
Y. Bouchiba (INSA, PhD)
C. Dumont (INSA, Toulouse)
J. Vucinic (INRA/INSA)
S. Traoré (PhD, CEA)
C. Viricel (PhD)
K. Zhang (Riken, CBDR)
S. Yagi (Riken, CBDR)
S. Tagami (Riken, CBDR)
RosettaCommons (U. Washington)
W. Sheffler (U. Washington)
V. Mulligan (Flatiron Institute, NY)
C. Bahl (IPI, Boston)
PyRosetta (U. John Hopkins)
B. Donald (U. North Carolina)
K. Roberts (U. North Carolina)
T. Simonson (Polytechnique)
J. Cortes (LAAS/CNRS),...

My apologies to those missing in these lists. Even imperfect lists seem better than no list

References

Besag, Julian (1975). “Statistical analysis of non-lattice data”. In: Journal of the Royal Statistical Society: Series D (The Statistician) 24.3, pp. 179–195.

Bessiere, Christian et al. (Aug. 2023). “Learning Constraint Networks over Unknown Constraint Languages”. In: Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23. Ed. by Edith Elkind. Main Track. International Joint Conferences on Artificial Intelligence Organization, pp. 1876–1883. DOI: 10.24963/ijcai.2023/208. URL: https://doi.org/10.24963/ijcai.2023/208.

Blondel, Mathieu et al. (2020). “Learning with Fenchel-Young losses”. In: J. Mach. Learn. Res. 21, 35:1–35:69. URL: https://jmlr.org/papers/v21/19-021.html.

Brouard, Céline et al. (2020). “Pushing Data into CP Models Using Graphical Model Learning and Solving”. In: International Conference on Principles and Practice of Constraint Programming. Springer, pp. 811–827.

Buchet, Samuel et al. (2024). “Bi-objective Discrete Graphical Model Optimization”. In: International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research. Springer, pp. 136–152.

Chang, Oscar et al. (2020). “Assessing SATNet’s Ability to Solve the Symbol Grounding Problem”. In: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, virtual. Ed. by Hugo Larochelle et al. URL: https://proceedings.neurips.cc/paper/2020/hash/0ff8033cf9437c213ee13937b1c4c455-Abstract.html.

References II

Colom, Mireia Solà et al. (Aug. 2024). “Complete combinatorial mutational enumeration of a protein functional site enables sequence-landscape mapping and identifies highly-mutated variants that retain activity”. In: Protein Science 33.8, e5109. DOI: `10.1002/pro.5109`. URL: `https://hal.science/hal-04646616`.

Dauparas, J. et al. (2022). “Robust deep learning-based protein sequence design using ProteinMPNN”. In: Science 378.6615, pp. 49–56. DOI: `10.1126/science.add2187`.

Defresne, Marianne, Sophie Barbe, et al. (2023). “Scalable Coupling of Deep Learning with Logical Reasoning”. In: Thirty-second International Joint Conference on Artificial Intelligence, IJCAI’2023.

Defresne, Marianne, Romain Gambardella, et al. (2026). “Scaling Neuro-symbolic Problem Solving: Solver-Free Learning of Constraints and Objectives”. In: Journal of Artificial Intelligence Research.

Durante, Valentin et al. (July 2022). “Efficient low rank convex bounds for pairwise discrete Graphical Models”. In: Thirty-ninth International Conference on Machine Learning.

Elmachtoub, Adam N. et al. (2022). “Smart “Predict, then Optimize””. In: Management Science 68.1, pp. 9–26. DOI: `10.1287/mnsc.2020.3922`.

Li, Zenan et al. (2023). “Neuro-symbolic learning yielding logical constraints”. In: Advances in Neural Information Processing Systems 36, pp. 21635–21657.

Liu, Hanxiao et al. (2021). “Pay Attention to MLPs”. In: Annual Conference on Neural Information Processing Systems, NeurIPS, pp. 9204–9215.

References III

Montanari, Andrea et al. (2009). “Which graphical models are difficult to learn?” In: Advances in Neural Information Processing Systems. Ed. by Y. Bengio et al. Vol. 22. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2009/file/22fb0cee7e1f3bde58293de743871417-Paper.pdf.

Nandwani, Yatin et al. (2021). “Neural Learning of One-of-Many Solutions for Combinatorial Problems in Structured Output Spaces”. In: 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. OpenReview.net. URL: https://openreview.net/forum?id=ATp1nW2FuZL.

Palm, Rasmus et al. (2018). “Recurrent Relational Networks”. In: Advances in Neural Information Processing Systems. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2018/file/b9f94c77652c9a76fc8a442748cd54bd-Paper.pdf.

Ruffini, Manon et al. (2021). “Guaranteed diversity and optimality in cost function network based computational protein design methods”. In: Algorithms 14.6, p. 168.

Topan, Sever et al. (2021). “Techniques for Symbol Grounding with SATNet”. In: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, virtual. Ed. by Marc’Aurelio Ranzato et al., pp. 20733–20744. URL: https://proceedings.neurips.cc/paper/2021/hash/ad7ed5d47b9baceb12045a929e7e2f66-Abstract.html.

Vucinic, Jelena et al. (2020). “Positive multistate protein design”. In: Bioinformatics 36.1, pp. 122–130.

Wang, Guan et al. (2025). “Hierarchical Reasoning Model”. In: arXiv preprint arXiv:2506.21734.

References IV

- Wang, Po-Wei et al. (2019). “SATNet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver”. In: Proceedings of the 36th International Conference on Machine Learning. Vol. 97. Proceedings of Machine Learning Research. PMLR, pp. 6545–6554.
- Yang, Zhun et al. (2023). “Learning to Solve Constraint Satisfaction Problems with Recurrent Transformer”. In: The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023. OpenReview.net. URL: https://openreview.net/forum?id=udNhDCr2KQe.
- Ye, Jiacheng et al. (2025). “Beyond Autoregression: Discrete Diffusion for Complex Reasoning and Planning”. In: The Thirteenth International Conference on Learning Representations. URL: https://openreview.net/forum?id=NRYgUzSPZz.