Pushing data into CP models using Graphical Model Learning & Solving

CP 2020

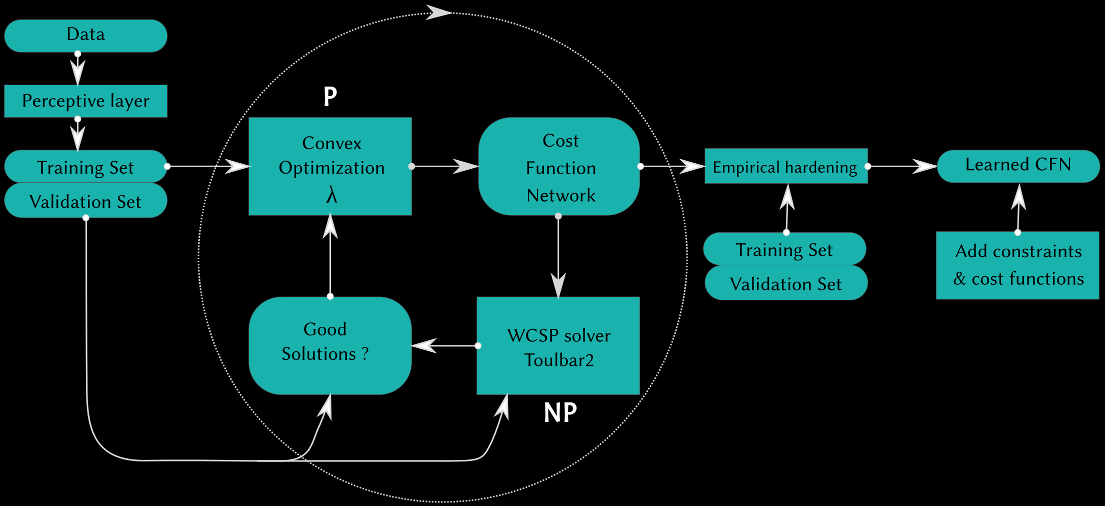
Céline Brouard[1], S. de Givry[2] & T. Schiex[2]

[1] Université Fédérale de Toulouse, INRAE MIAT, UR 875, Toulouse, France
[2] Université Fédérale de Toulouse, ANITI, INRAE MIAT, UR 875, Toulouse, France

CP and ML track

September 2020

Learning a Cost Function Network from high-quality solutions



Please, stay with us and...

You'll learn

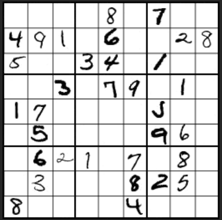
- how we use graphical models to connect CP with probabilistic Machine Learning
- how the NP-hard regularization loop can be made practical
- how we learn playing the Sudoku from images (without rules)
- how it compares with DL architectures that "learn to reason"
- how we can combine learned user preferences with (car) configuration constraints

You'll learn

- how we use graphical models to connect CP with probabilistic Machine Learning
- how the NP-hard regularization loop can be made practical
- how we learn playing the Sudoku from images (without rules)
- how it compares with DL architectures that "learn to reason"
- how we can combine learned user preferences with (car) configuration constraints

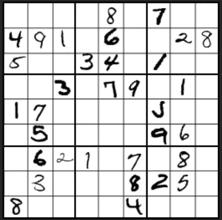
You'll learn

- how we use graphical models to connect CP with probabilistic Machine Learning
- how the NP-hard regularization loop can be made practical
- how we learn playing the Sudoku from images (without rules)
- how it compares with DL architectures that "learn to reason"
- how we can combine learned user preferences with (car) configuration constraints



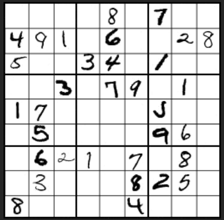
You'll learn

- how we use graphical models to connect CP with probabilistic Machine Learning
- how the NP-hard regularization loop can be made practical
- how we learn playing the Sudoku from images (without rules)
- how it compares with DL architectures that "learn to reason"
- how we can combine learned user preferences with (car) configuration constraints



You'll learn

- how we use graphical models to connect CP with probabilistic Machine Learning
- how the NP-hard regularization loop can be made practical
- how we learn playing the Sudoku from images (without rules)
- how it compares with DL architectures that "learn to reason"
- how we can combine learned user preferences with (car) configuration constraints



What is it ?

A description of a multivariate function as the combination of small functions

Cost Function Network $\mathcal{M}$ (unbounded)

- a set $\boldsymbol{V}$ of variables — n variables
- variable $X \in \boldsymbol{V}$ has domain D^X — max. size d
- a set $\boldsymbol{C}$ of cost functions
- $c_{\boldsymbol{S}} \in \boldsymbol{C} : \prod_{X \in \boldsymbol{S}} D^X \to \bar{\mathbb{Z}}$ — (∞)

Joint cost function — Weighted Constraint Satisfaction Problem

$$\boldsymbol{C}_{\mathcal{M}}(\boldsymbol{v}) = \sum_{c_{\boldsymbol{S}} \in \boldsymbol{C}} c_{\boldsymbol{S}}(\boldsymbol{v}[\boldsymbol{S}])$$

What is it ?

A description of a multivariate function as the combination of small functions

Cost Function Network $\mathcal{M}$ (unbounded)

- a set $\boldsymbol{V}$ of variables — n variables
- variable $X \in \boldsymbol{V}$ has domain D^X — max. size d
- a set $\boldsymbol{C}$ of cost functions
- $c_{\boldsymbol{S}} \in \boldsymbol{C} : \prod_{X \in \boldsymbol{S}} D^X \to \bar{\mathbb{Z}}$ — (∞)

Joint cost function — Weighted Constraint Satisfaction Problem

$$C_{\mathcal{M}}(\boldsymbol{v}) = \sum_{c_{\boldsymbol{S}} \in \boldsymbol{C}} c_{\boldsymbol{S}}(\boldsymbol{v}[\boldsymbol{S}])$$

What is it ?

A description of a multivariate function as the combination of small functions

Cost Function Network $\mathcal{M}$ (unbounded)

- a set $\boldsymbol{V}$ of variables — n variables
- variable $X \in \boldsymbol{V}$ has domain D^X — max. size d
- a set $\boldsymbol{C}$ of cost functions
- $c_{\boldsymbol{S}} \in \boldsymbol{C} : \prod_{X \in \boldsymbol{S}} D^X \to \bar{\mathbb{Z}}$ — (∞)

Joint cost function — Weighted Constraint Satisfaction Problem

$$C_{\mathcal{M}}(\boldsymbol{v}) = \sum_{c_{\boldsymbol{S}} \in \boldsymbol{C}} c_{\boldsymbol{S}}(\boldsymbol{v}[\boldsymbol{S}])$$

What do we want to learn ?

Definition (Learning a pairwise CFN from high quality solutions)

Given:

- a set of variables $\boldsymbol{V}$,
- a set of assignments $\boldsymbol{E}$ i.i.d. from an unknown distribution of high-quality solutions

Find a pairwise CFN $\mathcal{M}$ that can be solved to produce high-quality solutions

Pairwise CFN with cost-tables

- $\frac{n(n-1)}{2}$ tables of d^2 costs + n tables of d costs
- A constant table can be ignored.

Definition (Learning a pairwise CFN from high quality solutions)

Given:

- a set of variables $\boldsymbol{V}$,
- a set of assignments $\boldsymbol{E}$ i.i.d. from an unknown distribution of high-quality solutions

Find a pairwise CFN $\mathcal{M}$ that can be solved to produce high-quality solutions

Pairwise CFN with cost-tables

- $\frac{n(n-1)}{2}$ tables of d^2 costs + n tables of d costs
- A constant table can be ignored.

Markov Random Field $\mathcal{M}$

- a set $\boldsymbol{V}$ of domain variables
- a set Φ of potential functions
- $\varphi_{\boldsymbol{S}} \in \Phi : \prod_{X \in \boldsymbol{S}} D^X \rightarrow \mathbb{R}^+$

Joint function and probability distribution

$$\Phi_{\mathcal{M}}(\boldsymbol{v}) = \prod_{\varphi_{\boldsymbol{S}} \in \Phi} \varphi_{\boldsymbol{S}}(\boldsymbol{v}[\boldsymbol{S}]) \qquad P_{\mathcal{M}}(\boldsymbol{v}) \propto \Phi_{\mathcal{M}}(\boldsymbol{v})$$

From products to sum and back (up to some precision)

$$\text{MRF } \mathcal{M} \xrightarrow[-\log(x)]{} \text{CFN } \mathcal{M}^{\ell} \xrightarrow[\exp(-x)]{} \text{MRF } \mathcal{M}$$

Markov Random Field $\mathcal{M}$

- a set $\boldsymbol{V}$ of domain variables
- a set Φ of potential functions
- $\varphi_{\boldsymbol{S}} \in \Phi : \prod_{X \in \boldsymbol{S}} D^X \to \mathbb{R}^+$

Joint function and probability distribution

$$\Phi_{\mathcal{M}}(\boldsymbol{v}) = \prod_{\varphi_{\boldsymbol{S}} \in \Phi} \varphi_{\boldsymbol{S}}(\boldsymbol{v}[\boldsymbol{S}]) \qquad P_{\mathcal{M}}(\boldsymbol{v}) \propto \Phi_{\mathcal{M}}(\boldsymbol{v})$$

From products to sum and back (up to some precision)

$$\text{MRF } \mathcal{M} \xrightarrow[-\log(x)]{} \text{CFN } \mathcal{M}^{\ell} \xrightarrow[\exp(-x)]{} \text{MRF } \mathcal{M}$$

Markov Random Field $\mathcal{M}$

- a set $\boldsymbol{V}$ of domain variables
- a set Φ of potential functions
- $\varphi_{\boldsymbol{S}} \in \Phi : \prod_{X \in \boldsymbol{S}} D^X \to \mathbb{R}^+$

Joint function and probability distribution

$$\Phi_{\mathcal{M}}(\boldsymbol{v}) = \prod_{\varphi_{\boldsymbol{S}} \in \Phi} \varphi_{\boldsymbol{S}}(\boldsymbol{v}[\boldsymbol{S}]) \qquad P_{\mathcal{M}}(\boldsymbol{v}) \propto \Phi_{\mathcal{M}}(\boldsymbol{v})$$

From products to sum and back (up to some precision)

$$\text{MRF } \mathcal{M} \xrightarrow[-\log(x)]{} \text{CFN } \mathcal{M}^{\ell} \xrightarrow[\exp(-x)]{} \text{MRF } \mathcal{M}$$

Maximum likelihood estimation from i.i.d. sample $\boldsymbol{E}$

- Likelihood of $\mathcal{M}$: probability of $\boldsymbol{E}$ under $\mathcal{M}$
- Maximum likelihood $\mathcal{M}$: a MRF $\mathcal{M}$ that gives maximum probability to $\boldsymbol{E}$.

Maximum loglikelihood $\mathcal{M}$ on $\mathcal{M}_\ell$

$$\begin{aligned}
\mathcal{L}(\mathcal{M}, \boldsymbol{E}) &= \log(\textstyle\prod_{\boldsymbol{v}\in \boldsymbol{E}} P_{\mathcal{M}}(\boldsymbol{v})) = \textstyle\sum_{\boldsymbol{v}\in \boldsymbol{E}} \log(P_{\mathcal{M}}(\boldsymbol{v}))\\
&= \textstyle\sum_{\boldsymbol{v}\in \boldsymbol{E}} \log(\Phi_{\mathcal{M}}(\boldsymbol{v})) - \log(Z_{\mathcal{M}})\\
&= \underbrace{\sum_{\boldsymbol{v}\in \boldsymbol{E}} (-C_{\mathcal{M}^\ell}(\boldsymbol{v}))}_{\text{-costs of } \boldsymbol{E} \text{ samples}} \underbrace{- \log(\sum_{\boldsymbol{t}\in \prod_{x\in \boldsymbol{V}} D^X} \exp(-C_{\mathcal{M}^\ell}(\boldsymbol{t})))}_{\text{Soft-Min of all assignment costs}}
\end{aligned}$$

Maximum likelihood estimation from i.i.d. sample $\boldsymbol{E}$

- Likelihood of $\mathcal{M}$: probability of $\boldsymbol{E}$ under $\mathcal{M}$
- Maximum likelihood $\mathcal{M}$: a MRF $\mathcal{M}$ that gives maximum probability to $\boldsymbol{E}$.

Maximum loglikelihood $\mathcal{M}$ on $\mathcal{M}_\ell$

$$\begin{aligned}
\mathcal{L}(\mathcal{M}, \boldsymbol{E}) &= \log(\textstyle\prod_{\boldsymbol{v}\in\boldsymbol{E}} P_{\mathcal{M}}(\boldsymbol{v})) = \sum_{\boldsymbol{v}\in\boldsymbol{E}} \log(P_{\mathcal{M}}(\boldsymbol{v}))\\
&= \textstyle\sum_{\boldsymbol{v}\in\boldsymbol{E}} \log(\Phi_{\mathcal{M}}(\boldsymbol{v})) - \log(Z_{\mathcal{M}})\\
&= \underbrace{\sum_{\boldsymbol{v}\in\boldsymbol{E}} (-C_{\mathcal{M}^\ell}(\boldsymbol{v}))}_{\text{-costs of } \boldsymbol{E} \text{ samples}} \underbrace{- \log(\sum_{\boldsymbol{t}\in\prod_{x\in \boldsymbol{V}} D^X} \exp(-C_{\mathcal{M}^\ell}(\boldsymbol{t})))}_{\text{Soft-Min of all assignment costs}}
\end{aligned}$$

Regularized approximate max-log-likelihood estimation

Regularized Log-Likelihood estimation

- penalizes log-likelihood proportionally to the L_1 norm of the costs learned (λ)
- avoids over-fitting by pushing non essential costs to zero: learns scopes.

PE MRF: ADMM optimized convex approximation of regularized loglikelihood[1]

- avoids #P-completeness using a concave approximation of $Z_{\mathcal{M}}$
- statistically sparsistent
- provides a CFN as output

[1] Youngsuk Park et al. "Learning the network structure of heterogeneous data via pairwise exponential Markov random fields". In: *Proceedings of machine learning research* 54 (2017), p. 1302.

Regularized approximate max-log-likelihood estimation

Regularized Log-Likelihood estimation

- penalizes log-likelihood proportionally to the L_1 norm of the costs learned (λ)
- avoids over-fitting by pushing non essential costs to zero: learns scopes.

PE MRF: ADMM optimized convex approximation of regularized loglikelihood[1]

- avoids #P-completeness using a concave approximation of $Z_{\mathcal{M}}$
- statistically sparsistent
- provides a CFN as output

[1] Youngsuk Park et al. "Learning the network structure of heterogeneous data via pairwise exponential Markov random fields". In: *Proceedings of machine learning research* 54 (2017), p. 1302.

Selecting a suitable value of λ

Using empirical risk minimization

- for each sample v in the validation set
- assign a fraction of v and solve with a WCSP solver
- prefer λ that gives solutions close to v

Controlling PyToulbar2 NP-hard optimization effort

- bounded optimization effort (backtrack, time, gap. Here: 50,000 backtracks)
- controllable fraction of v assigned

Empirical hardening

Set positive costs that are never violated in the training/validation sets to ∞.

Selecting a suitable value of λ

Using empirical risk minimization

- for each sample v in the validation set
- assign a fraction of v and solve with a WCSP solver
- prefer λ that gives solutions close to v

Controlling PyToulbar2 NP-hard optimization effort

- bounded optimization effort (backtrack, time, gap. Here: 50,000 backtracks)
- controllable fraction of v assigned

Empirical hardening

Set positive costs that are never violated in the training/validation sets to ∞.

Using empirical risk minimization

- for each sample v in the validation set
- assign a fraction of v and solve with a WCSP solver
- prefer λ that gives solutions close to v

Controlling PyToulbar2 NP-hard optimization effort

- bounded optimization effort (backtrack, time, gap. Here: 50,000 backtracks)
- controllable fraction of v assigned

Empirical hardening

Set positive costs that are never violated in the training/validation sets to ∞.

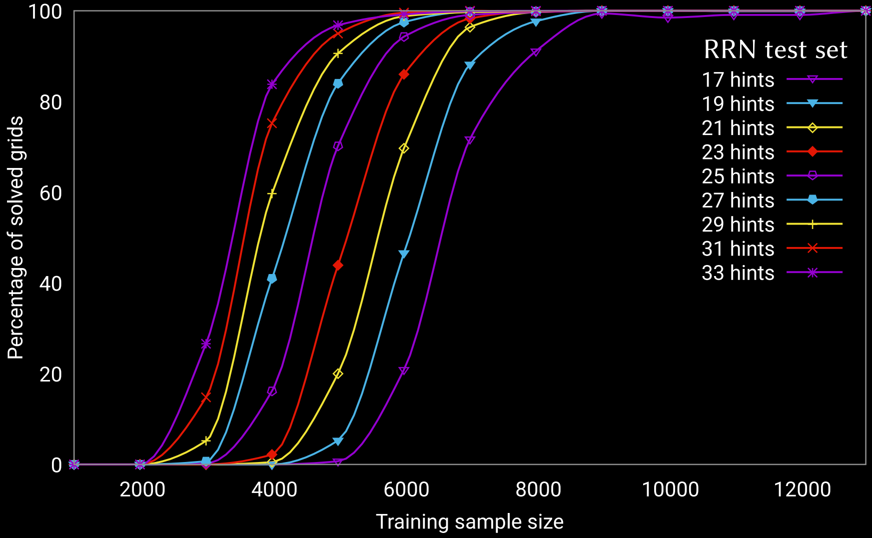
An exemplar of reasoning for benchmarking

- Recurrent Relational Neural Net[2]: $18 \times (10,000 + 1,000 + 1,000)$ training, validation and test samples of variable difficulty (17 to 34 hints).
- SAT-Net[3] (DL friendly convex Max-SAT relaxation): $(9,000 + 1,000)$ easy training and test samples (36.2 hints average).

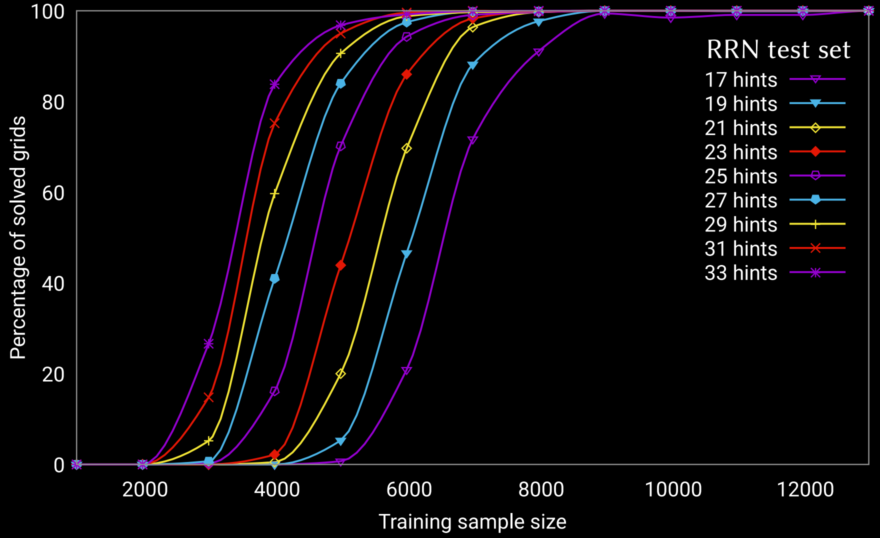
[2] Rasmus Berg Palm, Ulrich Paquet, and Ole Winther. "Recurrent Relational Networks". In: *Advances in Neural Information Processing Systems, Montréal, Canada.* 2018, pp. 3372–3382.

[3] Po-Wei Wang et al. "SATNet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver". In: *Proc. of ICML-19, Long Beach, California, USA.* vol. 97. PMLR, 2019, pp. 6545–6554.

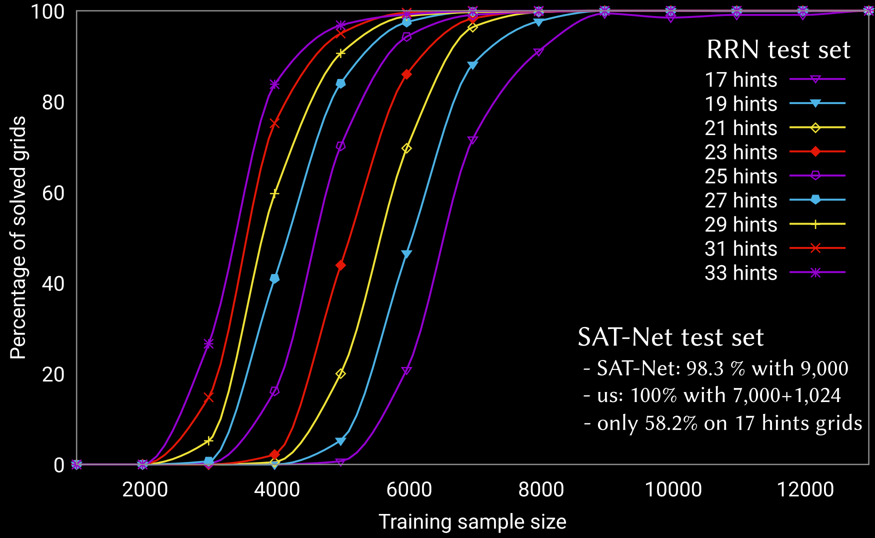
Better with less data and comparable biases



Better with less data and comparable biases



RRN: 96.6%
180,000 samples
(scopes given)



SAT-Net test set

- SAT-Net: 98.3 % with 9,000
- us: 100% with 7,000+1,024
- only 58.2% on 17 hints grids

RRN: 96.6%
180,000 samples
(scopes given)

Sudoku digits can be LeNet decoded and fed to PE MRF/Toulbar2

- LeNet has 99.2% accuracy on handwritten digits
- SAT-Net test set, hints as images (36.2 avg): · 74.7% max. accuracy
- Hints + solutions as images: · 52% max. accuracy

Performances on SAT-Net test set

- SAT-Net (hints as images), 9,000 samples · 63.2%
- PE MRF+Toulbar2, 8,000+1,024 samples · 78.1%
- On hard 17 hints test RRN problems · 81.2%
- Empirical hardening · $> 99\%$
- Hints and solutions as images · 76.3%

Sudoku digits can be LeNet decoded and fed to PE MRF/Toulbar2

- LeNet has 99.2% accuracy on handwritten digits
- SAT-Net test set, hints as images (36.2 avg): · 74.7% max. accuracy
- Hints + solutions as images: · 52% max. accuracy

Performances on SAT-Net test set

- SAT-Net (hints as images), 9,000 samples · 63.2%
- PE MRF+Toulbar2, 8,000+1,024 samples · 78.1%
- On hard 17 hints test RRN problems · 81.2%
- Empirical hardening · > 99%
- Hints and solutions as images · 76.3%

Sudoku digits can be LeNet decoded and fed to PE MRF/Toulbar2

- LeNet has 99.2% accuracy on handwritten digits
- SAT-Net test set, hints as images (36.2 avg): ..... 74.7% max. accuracy
- Hints + solutions as images: ..... 52% max. accuracy

Performances on SAT-Net test set

- SAT-Net (hints as images), 9,000 samples 63.2%
- PE MRF+Toulbar2, 8,000+1,024 samples 78.1%
- On hard 17 hints test RRN problems 81.2%
- Empirical hardening ..... > 99%
- Hints and solutions as images 76.3%

Sudoku digits can be LeNet decoded and fed to PE MRF/Toulbar2

- LeNet has 99.2% accuracy on handwritten digits
- SAT-Net test set, hints as images (36.2 avg): .... 74.7% max. accuracy
- Hints + solutions as images: .... 52% max. accuracy

Performances on SAT-Net test set

- SAT-Net (hints as images), 9,000 samples 63.2%
- PE MRF+Toulbar2, 8,000+1,024 samples 78.1%
- On hard 17 hints test RRN problems 81.2%
- Empirical hardening .... $> 99\%$
- Hints and solutions as images 76.3%

Sudoku digits can be LeNet decoded and fed to PE MRF/Toulbar2

- LeNet has 99.2% accuracy on handwritten digits
- SAT-Net test set, hints as images (36.2 avg): · 74.7% max. accuracy
- Hints + solutions as images: · 52% max. accuracy

Performances on SAT-Net test set

- SAT-Net (hints as images), 9,000 samples · 63.2%
- PE MRF+Toulbar2, 8,000+1,024 samples · 78.1%
- On hard 17 hints test RRN problems · 81.2%
- Empirical hardening · $> 99\%$
- Hints and solutions as images · 76.3%

Sudoku digits can be LeNet decoded and fed to PE MRF/Toulbar2

- LeNet has 99.2% accuracy on handwritten digits
- SAT-Net test set, hints as images (36.2 avg): …… 74.7% max. accuracy
- Hints + solutions as images: …… 52% max. accuracy

Performances on SAT-Net test set

- SAT-Net (hints as images), 9,000 samples …… 63.2%
- PE MRF+Toulbar2, 8,000+1,024 samples …… 78.1%
- On hard 17 hints test RRN problems …… 81.2%
- Empirical hardening …… > 99%
- Hints and solutions as images …… 76.3%

Sudoku digits can be LeNet decoded and fed to PE MRF/Toulbar2

- LeNet has 99.2% accuracy on handwritten digits
- SAT-Net test set, hints as images (36.2 avg): . . . . 74.7% max. accuracy
- Hints + solutions as images: . . . . 52% max. accuracy

Performances on SAT-Net test set

- SAT-Net (hints as images), 9,000 samples 63.2%
- PE MRF+Toulbar2, 8,000+1,024 samples 78.1%
- On hard 17 hints test RRN problems 81.2%
- Empirical hardening . . . . $> 99\%$
- Hints and solutions as images 76.3%

Benefits from numerical learning and solving

Sudoku digits can be LeNet decoded and fed to PE MRF/Toulbar2

- LeNet has 99.2% accuracy on handwritten digits
- SAT-Net test set, hints as images (36.2 avg): .... 74.7% max. accuracy
- Hints + solutions as images: .... 52% max. accuracy

Performances on SAT-Net test set

- SAT-Net (hints as images), 9,000 samples 63.2%
- PE MRF+Toulbar2, 8,000+1,024 samples 78.1%
- On hard 17 hints test RRN problems 81.2%
- Empirical hardening .... $> 99\%$
- Hints and solutions as images 76.3%

Renault "big" dataset `irit.fr/ Helene.Fargier/BR4CP/benches.html`

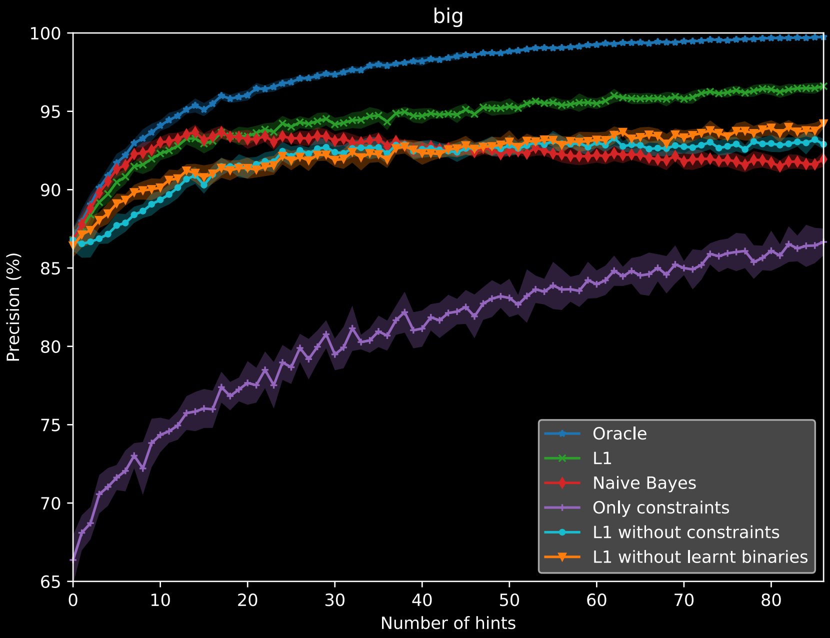
- 268 variables (87 decision variables) with 324 values at most
- 332 constraints (max. arity 12)
- 24,566,537,954,855,758,069,760 possible configurations ($\approx 2^{74}$)[4]
- sample of 8,337 user configurations

Configuration assistant: complete ongoing user configuration on the next variable

- Learned CFN (unary + binary CFs, 10 fold CV, 2' / fold) + configuration constraints
- Naive Bayes (knowing the partial assignment)
- Oracle (optimal stochastic choice for the test set given the partial assignment)

[4] Counted in 1.8" by Toulbar2 treewidth aware solution counter.

Accuracy


big
Precision (%)
100
95
90
85
80
75
70
65
0
10
20
30
40
50
60
70
80
Number of hints
Oracle
L1
Naive Bayes
Only constraints
L1 without constraints
L1 without learnt binaries

Conclusions

- Flexible learning framework for CP with understandable and editable output
- Numerical (even integer) weights are enough to interact with DL output
- Anytime NP-hard prediction: more powerful than differentiable convex relaxation
- Convex relaxations may be the best we can do in P (Unique Game Conjecture)
- One should try anytime NP-hard learning too

Conclusions

- Flexible learning framework for CP with understandable and editable output
- Numerical (even integer) weights are enough to interact with DL output
- Anytime NP-hard prediction: more powerful than differentiable convex relaxation
- Convex relaxations may be the best we can do in P (Unique Game Conjecture)
- One should try anytime NP-hard learning too

Conclusions

- Flexible learning framework for CP with understandable and editable output
- Numerical (even integer) weights are enough to interact with DL output
- Anytime NP-hard prediction: more powerful than differentiable convex relaxation
- Convex relaxations may be the best we can do in P (Unique Game Conjecture)
- One should try anytime NP-hard learning too

Conclusions

- Flexible learning framework for CP with understandable and editable output
- Numerical (even integer) weights are enough to interact with DL output
- Anytime NP-hard prediction: more powerful than differentiable convex relaxation
- Convex relaxations may be the best we can do in P (Unique Game Conjecture)
- One should try anytime NP-hard learning too

Conclusions

- Flexible learning framework for CP with understandable and editable output
- Numerical (even integer) weights are enough to interact with DL output
- Anytime NP-hard prediction: more powerful than differentiable convex relaxation
- Convex relaxations may be the best we can do in P (Unique Game Conjecture)
- One should try anytime NP-hard learning too