Learning from user and environment in combinatorial optimisation

Prof. Tias Guns
<tias.guns@kuleuven.be>
@TiasGuns

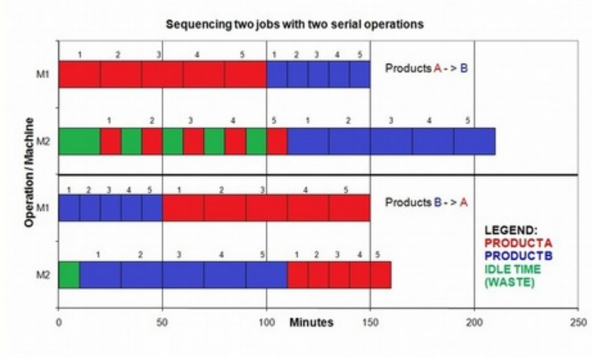
Joint work with team members:
- Rocs Canoy
- Jayanta Mandi
- Maxime Mulamba
- Victor Bucarey Lopez
- Ahmed KA Abdullah
- Emilio Gamba

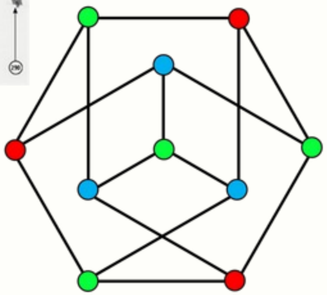
And external colaborators:
- Peter Stuckey (Monash Uni, Au)
- Emir Demirovic (TU Delft, NL)
- Michelangelo Diligenti (Sienna Uni, It)
- Michele Lombardi (Bologna, It)
- Bart Bogaerts (VUB, Be)

Combinatorial optimisation

“Solving *constrained* optimisation problems”

- Vehicle Routing
- Scheduling
- Configuration
- Graph problems

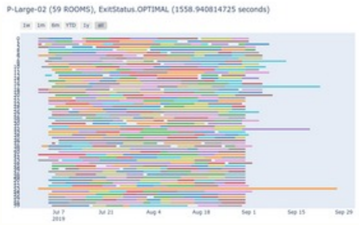




Constraint solving paradigm

Model + Solve

Decision variables
Constraints
Objective function



Research trend

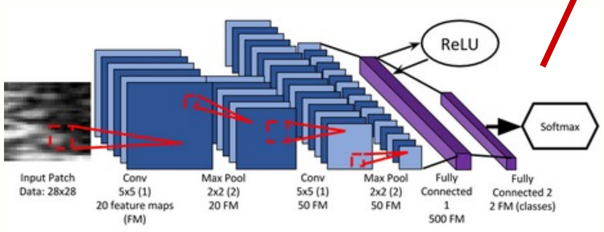
Model + **Solve**

Decision variables
Constraints
Objective function

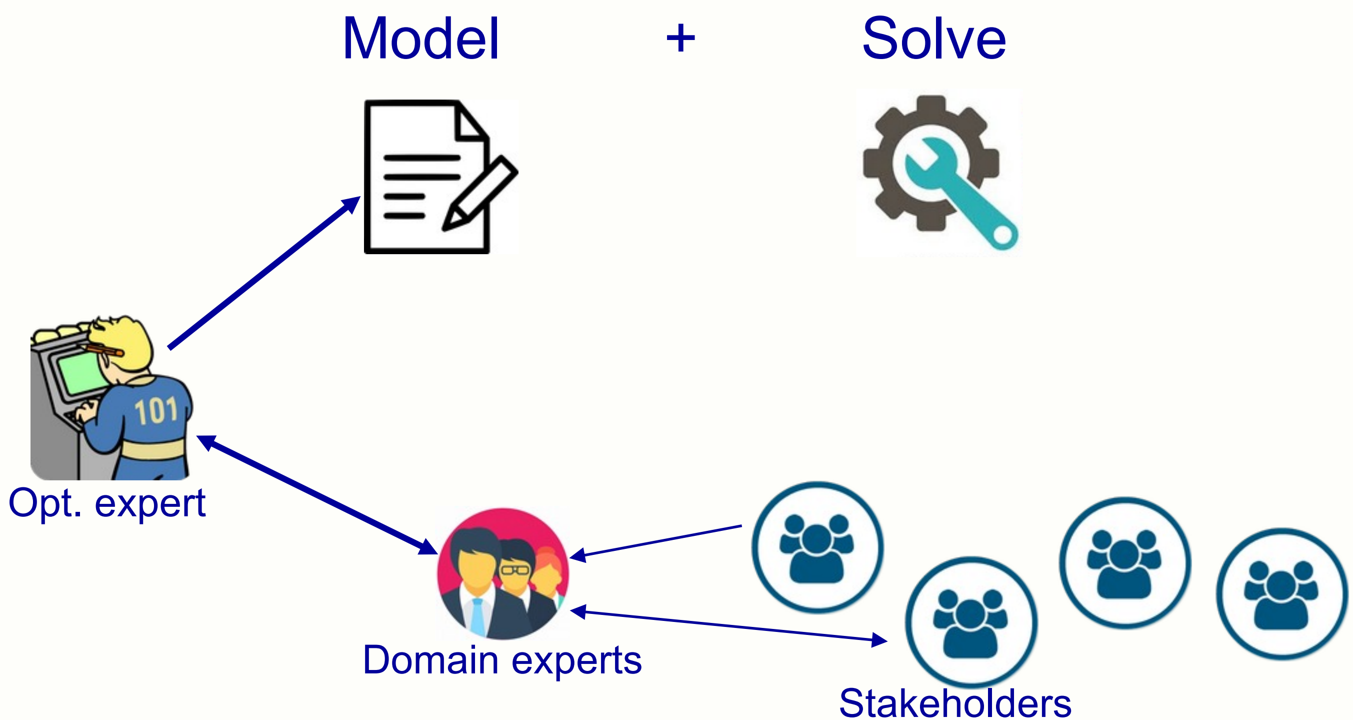
1) learn to model

2) learn to solve (faster)

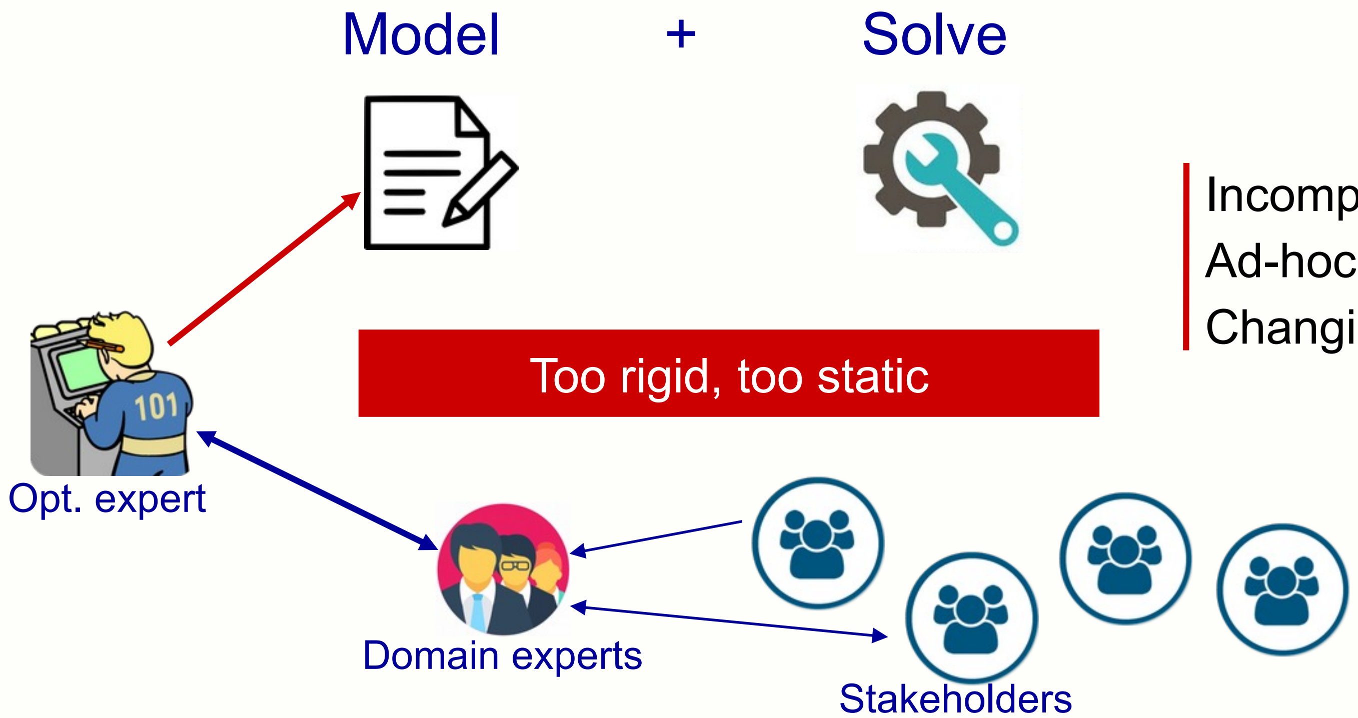
Can we *learn* it instead?



Current combinatorial optimisation practice



Current combinatorial opt. practice, problem

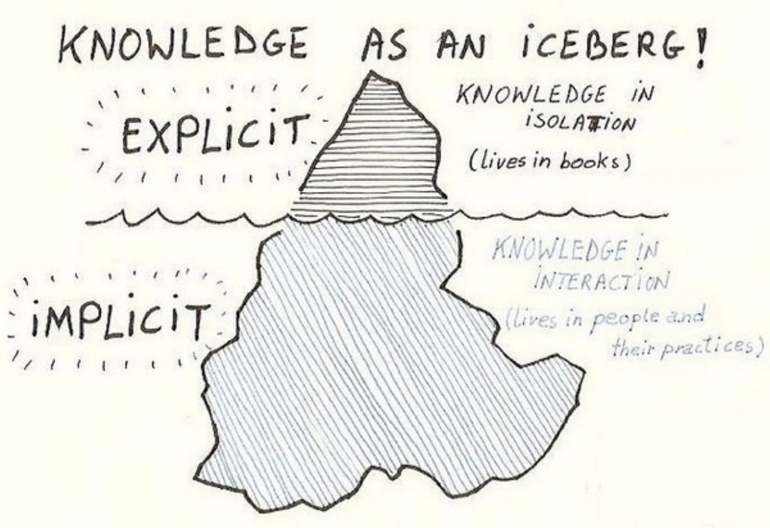


Incomplete information
Ad-hoc modifications
Changing preferences

Prediction + constraint solving

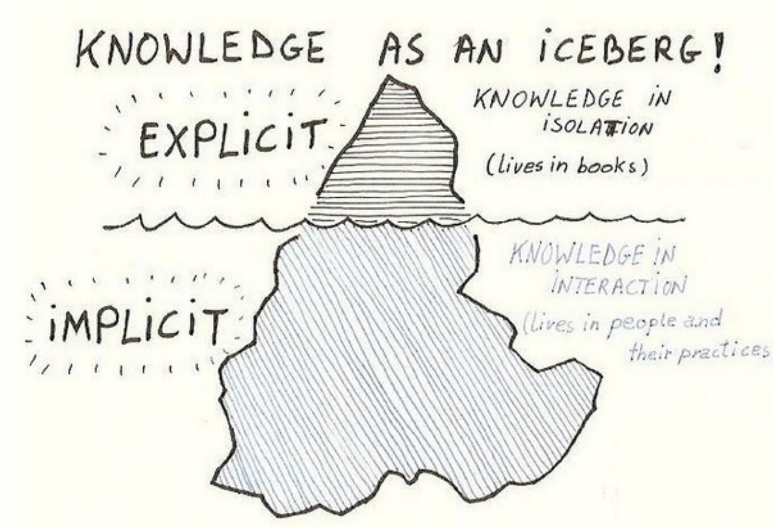
- Part explicit knowledge:
  in a formal language

- Part implicit knowledge:
  learned from data



Prediction + constraint solving

- Part <u>explicit</u> knowledge:
  in a formal language



- Part <u>implicit</u> knowledge:
  learned from data

» **perception** *(vision, natural language, ...)*

» tacit knowledge *(user preferences, social, ...)*

» complex environment *(demand, prices, ...)*

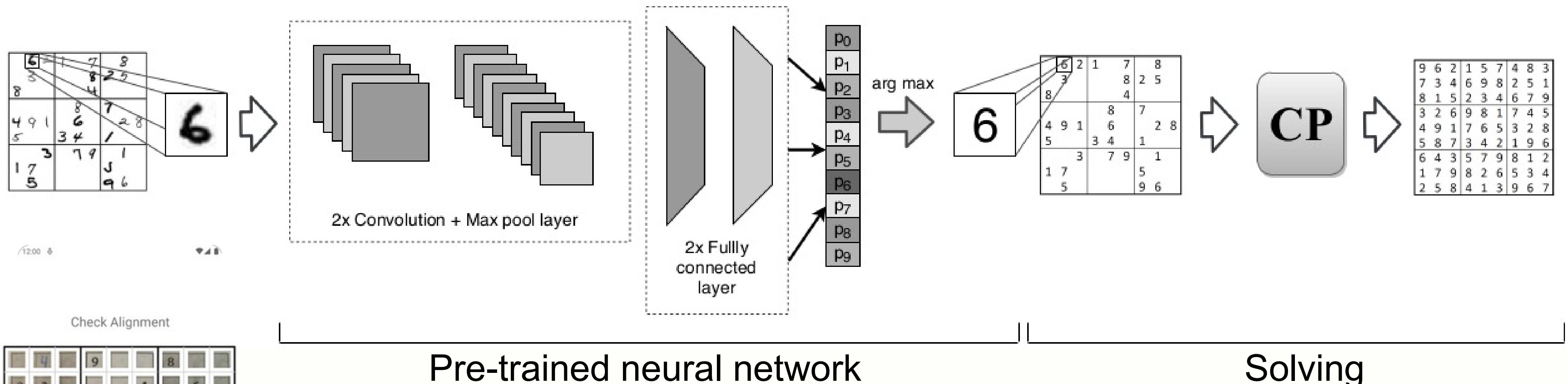
Perception-based Constraint Solving: a demo application

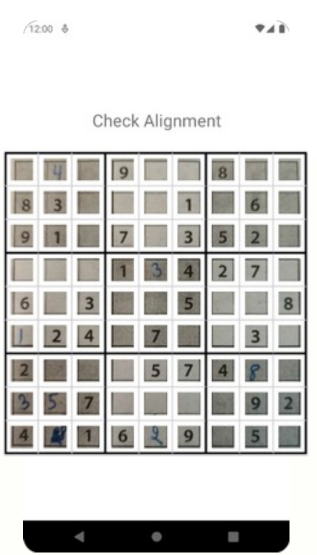
GET IT ON Google Play

https://visualsudoku.cs.kuleuven.be

Perception-based constraint solving

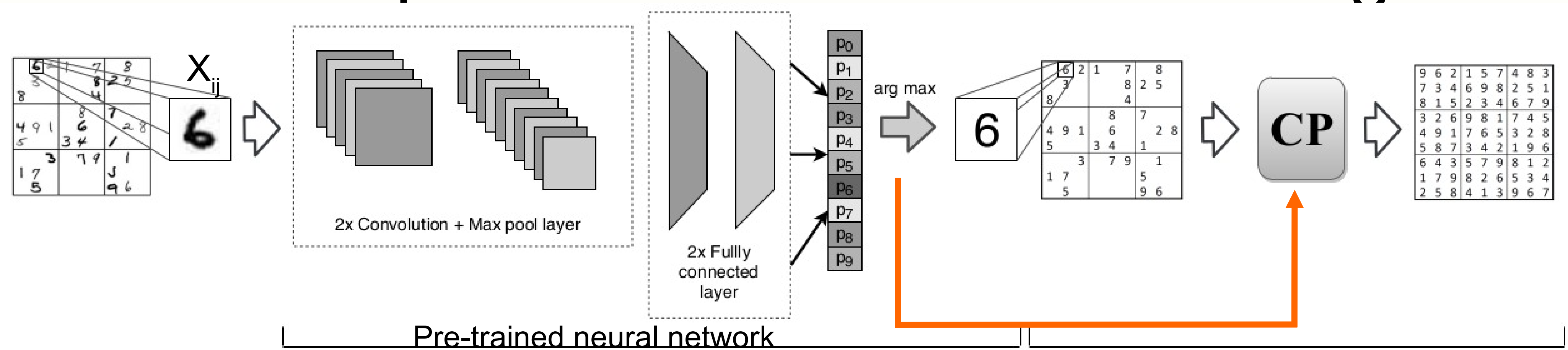
Visual sudoku (naïve)





	accuracy img	accuracy cell	accuracy grid	failure rate grid	time average (s)
baseline	94.75%	15.51%	14.67%	84.43%	0.01

Perception-based constraint solving



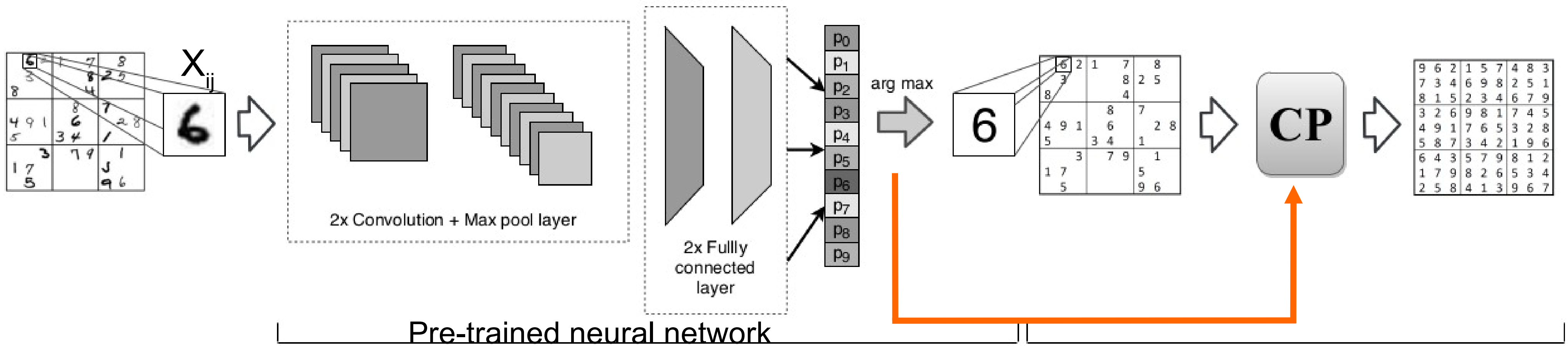
<u>What about the *next* most likely interpretation?</u>

- Treat prediction as *joint inference* problem:

$$\hat{y} = \arg\max \prod_{ij} P(y_{ij} = k | X_{ij}) \quad \text{s.t.} \quad \text{sudoku}(\hat{y})$$

- This is the **constrained** 'maximum likelihood' interpretation

Perception-based constraint solving



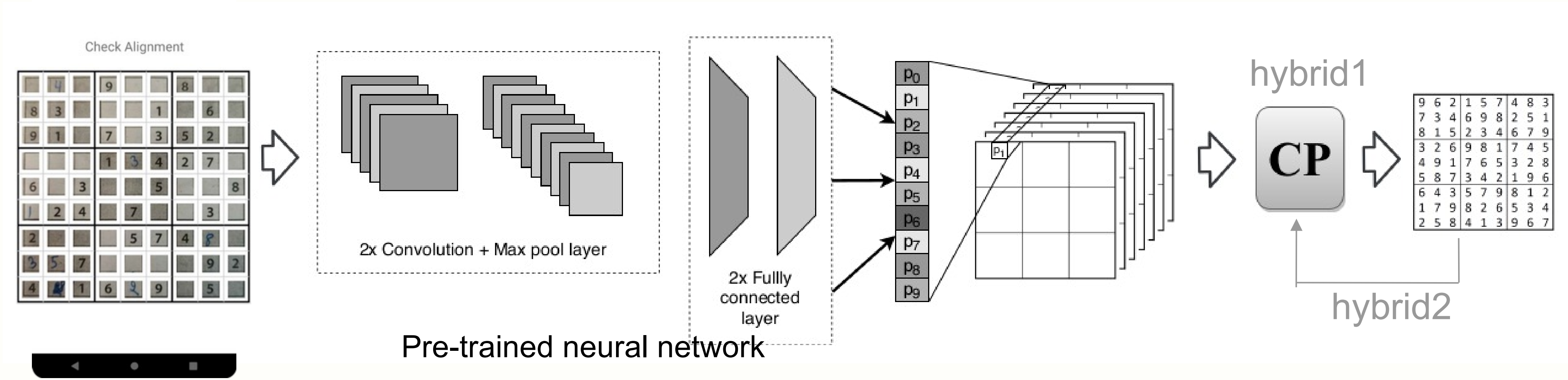
$$\hat{y} = \arg\max \prod_{ij} P(y_{ij} = k | X_{ij}) \quad \text{s.t.} \quad \text{sudoku}(\hat{y})$$

- Log-likelihood trick:

$$min \sum_{\substack{(i,j)\in \\ given}} \sum_{\substack{k\in \\ \{1,..,9\}}} \underbrace{-log(P_\theta(y_{ij} = k | X_{ij}))}_{constant} * \mathbb{1}[s_{ij} = k] \quad \text{s.t.} \quad \text{sudoku}(\hat{y})$$

Perception-based constraint solving

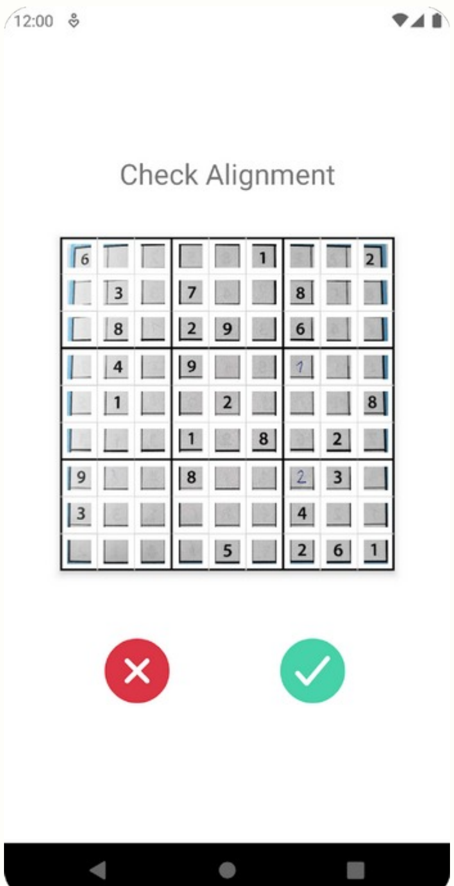
Hybrid: CP solver does *joint inference* over raw probabilities

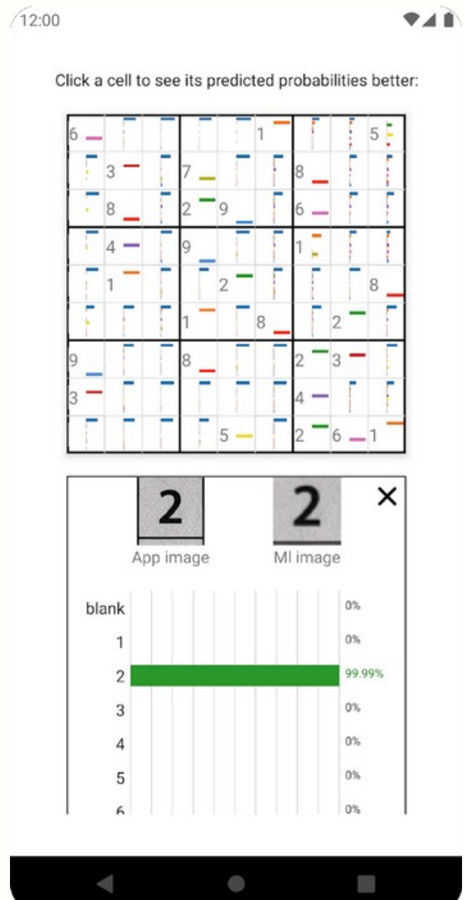


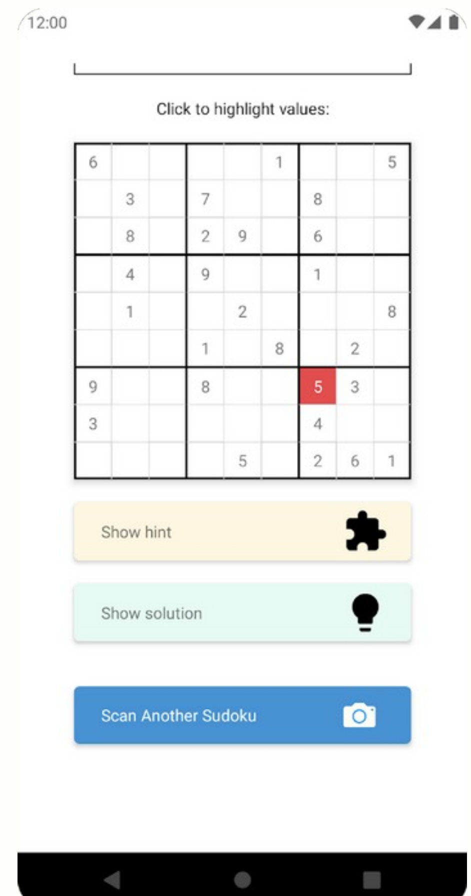
	accuracy img	accuracy cell	accuracy grid	failure rate grid	time average (s)
baseline	94.75%	15.51%	14.67%	84.43%	0.01
hybrid1	99.69%	99.38%	92.33%	0%	0.79
hybrid2	99.72%	99.44%	92.93%	0%	0.83

[Maxime Mulamba, Jayanta Mandi, Rocsildes Canoy, Tias Guns, CPAIOR20]

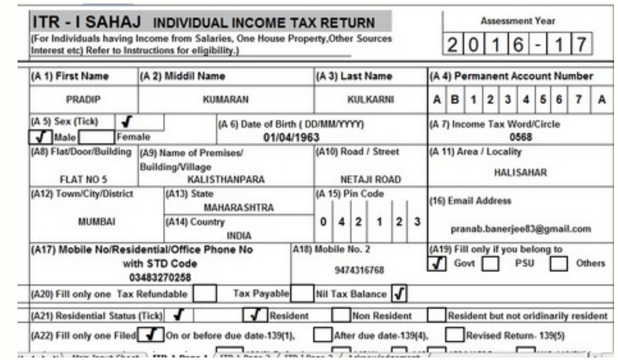
Sudoku Assistant demo, continued







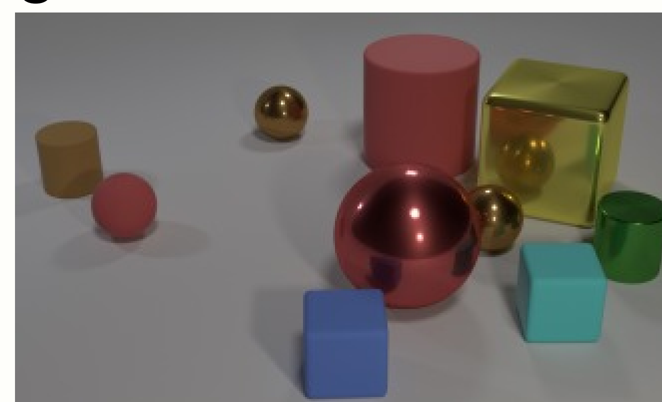
Perception data and constraint solving


ITR - I SAHAJ INDIVIDUAL INCOME TAX RETURN
(For Individuals having Income from Salaries, One House Property,Other Sources Interest etc) Refer to Instructions for eligibility.)
Assessment Year
2016-17

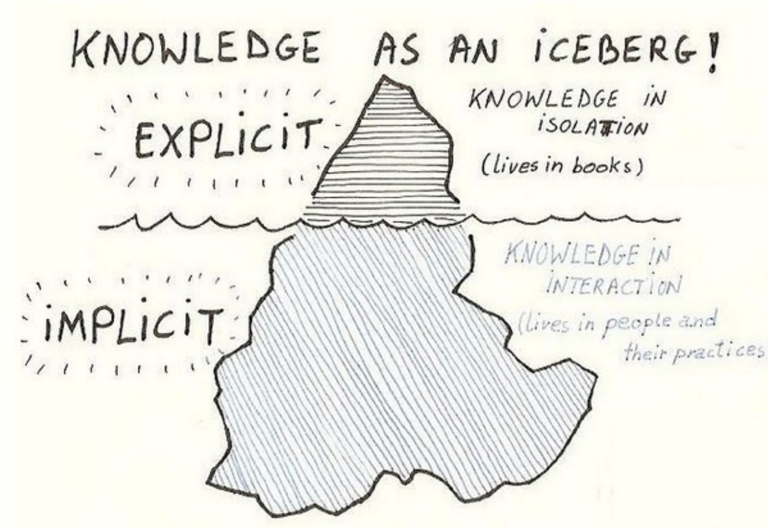
(A 1) First Name: PRADIP
(A 2) Middil Name: KUMARAN
(A 3) Last Name: KULKARNI
(A 4) Permanent Account Number: A B 1 2 3 4 5 6 7 A
(A 5) Sex (Tick) ✓ Male Female
(A 6) Date of Birth (DD/MM/YYYY) 01/04/1963
(A 7) Income Tax Word/Circle 0568
(A8) Flat/Door/Building: FLAT NO 5
(A9) Name of Premises/ Building/Village: KALISTHANPARA
(A10) Road / Street: NETAJI ROAD
(A 11) Area / Locality: HALISAHAR
(A12) Town/City/District: MUMBAI
(A13) State: MAHARASHTRA
(A14) Country: INDIA
(A 15) Pin Code: 0 4 2 1 2 3
(16) Email Address: pranab.banerjee83@gmail.com
(A17) Mobile No/Residential/Office Phone No with STD Code 03483270258
A18) Mobile No. 2: 9474316768
(A19) Fill only if you belong to ✓ Govt PSU Others
(A20) Fill only one Tax Refundable Tax Payable Nil Tax Balance ✓
(A21) Residential Status (Tick) ✓ Resident Non Resident Resident but not oridinarily resident
(A22) Fill only one Filed ✓ On or before due date-139(1). After due date-139(4). Revised Return- 139(5)

Other application settings:

- Document analysis
- Paper-based configuration problems (tax forms)
- Object-detection based reasoning
- Visual relationship detection
- ...



Prediction + constraint solving



- Part <u>explicit</u> knowledge:
  in a formal language

- Part <u>implicit</u> knowledge:
  learned from data

» perception *(vision, natural language, ...)*

» **tacit knowledge** *(user preferences, social, ...)*

» complex environment *(demand, prices, ...)*

Tacit knowledge (user preferences)

Example:

"in real-life operations, the quality of a route is not exclusively defined by its theoretical length, duration, or cost"

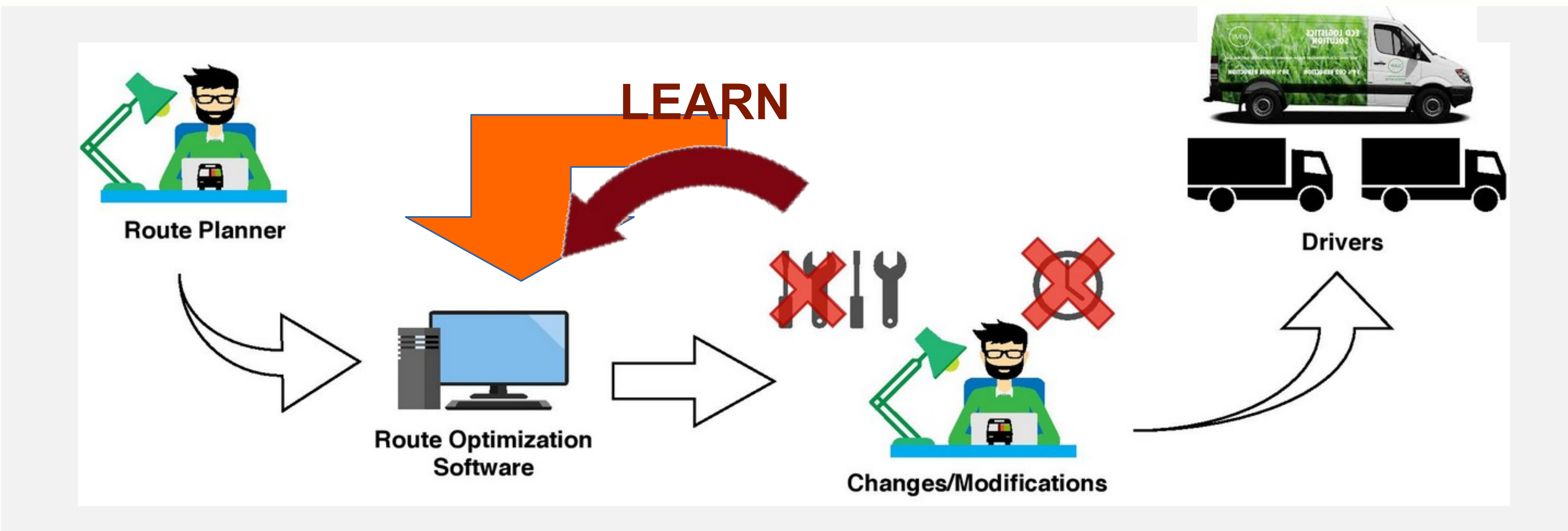
- Data:
  - stop list, zones
  - TSP solutions with "good, average, bad" labels

→ learn and route

Tacit knowledge (user preferences)

"Vehicle routing by learning from historical solutions"

[Rocsildes Canoy and Tias Guns, CP19], **Best student paper award**



GOAL: Learn preferences, reduce manual effort, adapt to changes over time!

Tacit knowledge (user preferences)

Small data: 6 months = 26 weeks = 130 week days (instances)

Tacit knowledge (user preferences)

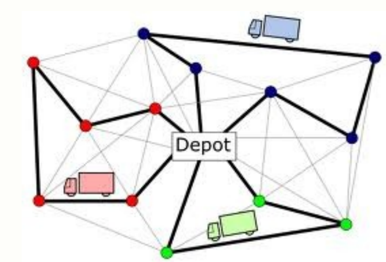
For <u>single</u> vehicles, in mobility data mining literature:

- Driver turn prediction *[Krumm, 2008]*
- Prediction of remainder of route early in the trip *[Ye et al., 2015]*
- Prediction of route given origin and destination *[Wang et al., 2015]*

Can we use similar techniques

to learn <u>preferences across routings</u> of multiple vehicles?

And can we <u>optimize</u> over them with constraint solving?



Learning and prediction part

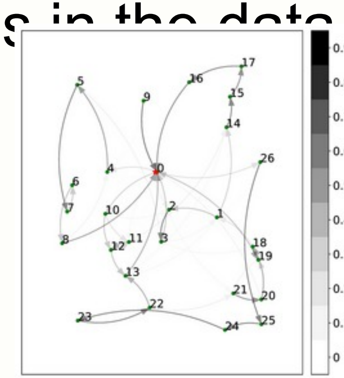
1st order Markov approximation:

$P([s1,s2,s3,...]) = P(s1)*P(s2|s1)*P(s3|s2)*...$

→ estimate the $P(s_y|s_x)$ by observing the transitions in the actually driven routes

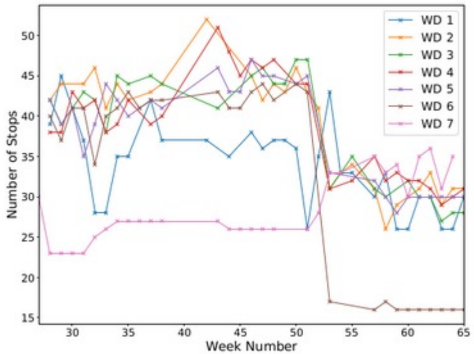
probability of transition = relative nr of observations in the data

$$t_{ij} = \frac{f_{ij} + \alpha}{N_i + \alpha d},$$



[Canoy, Guns, Vehicle routing by learning from historical solutions, CP19]

Concept drift: new/gone clients



When 'counting' the probabilities:

- can include a *prior* on each historic instance wrt. current day
- e.g. weighing of the instance:

$$\mathbf{F} = \sum_t w_t \mathbf{A}^t.$$

  - uniform = unit weight
  - by similarity = how much overlap in clients with current day
  - by time = more recent instances get higher weight
    incl. exponential smoothing

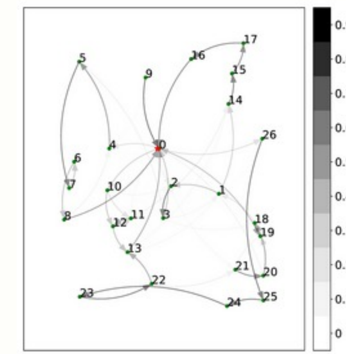
[Canoy, Guns, Vehicle routing by learning from historical solutions, CP19]

Constrained optimisation: what now?

Goal: find constrained <u>maximum likelihood</u> solution:

maximize P([s1,s2,s3,...) = P(s1)*P(s2|s1)*P(s3|s2)*...

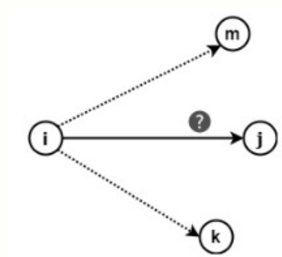
s.t. VRP([s1,s2,s3,...])



Standard probability computation trick: log-likelihood

$$\max \prod_{(i,j)\in X} \mathbf{Pr}(\text{next stop}=j \mid \text{current stop}=i),$$

$$= \max \sum_{(i,j)\in A} log(t_{ij})x_{ij}.$$



→ *VRP:* replace *distance matrix* by *negative log-likelihood* matrix!

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij} \implies \min \sum_{(i,j)\in A} -log(t_{ij})x_{ij}.$$

Compatible with ALL vrp solvers

Concept drift, quality AFTER solving

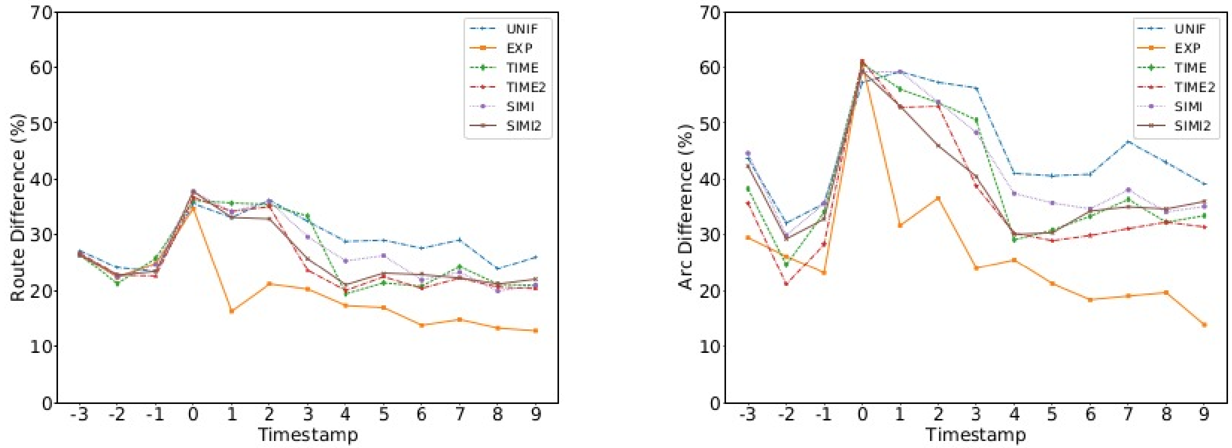


Fig. 7 **Route and arc difference during concept drift (drop in number of stops)**

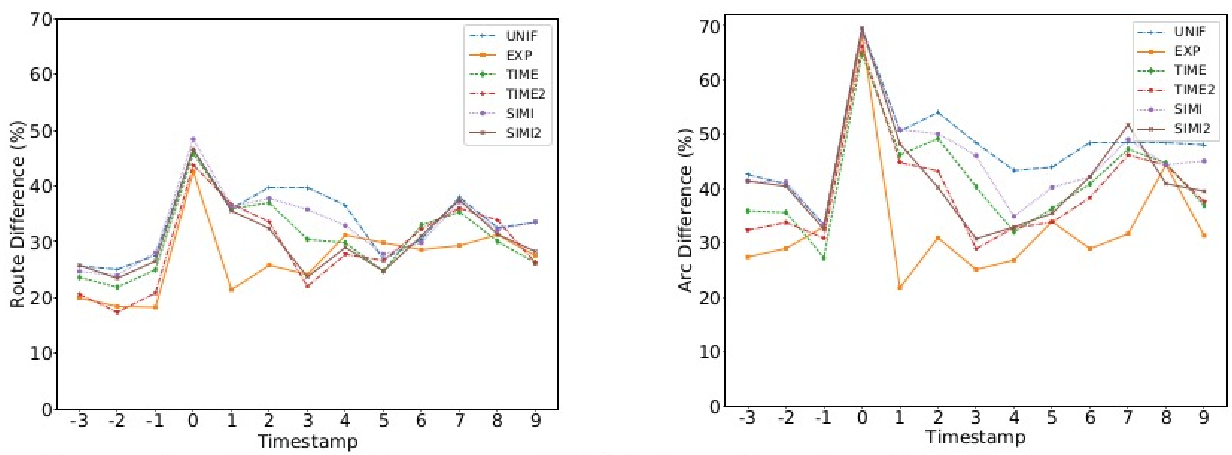


Fig. 8 **Route and arc difference during concept drift (rise in number of stops)**

[Canoy, Guns, Vehicle routing by learning from historical solutions, CP19]

Extension: Neural instead of Markov?

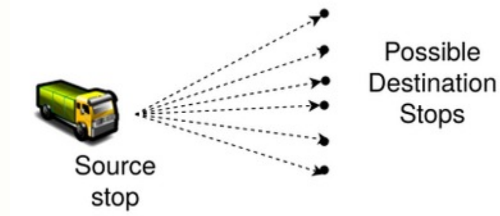
Opportunities: contextual *features* (day of week, nr vehicles...)

Challenges:

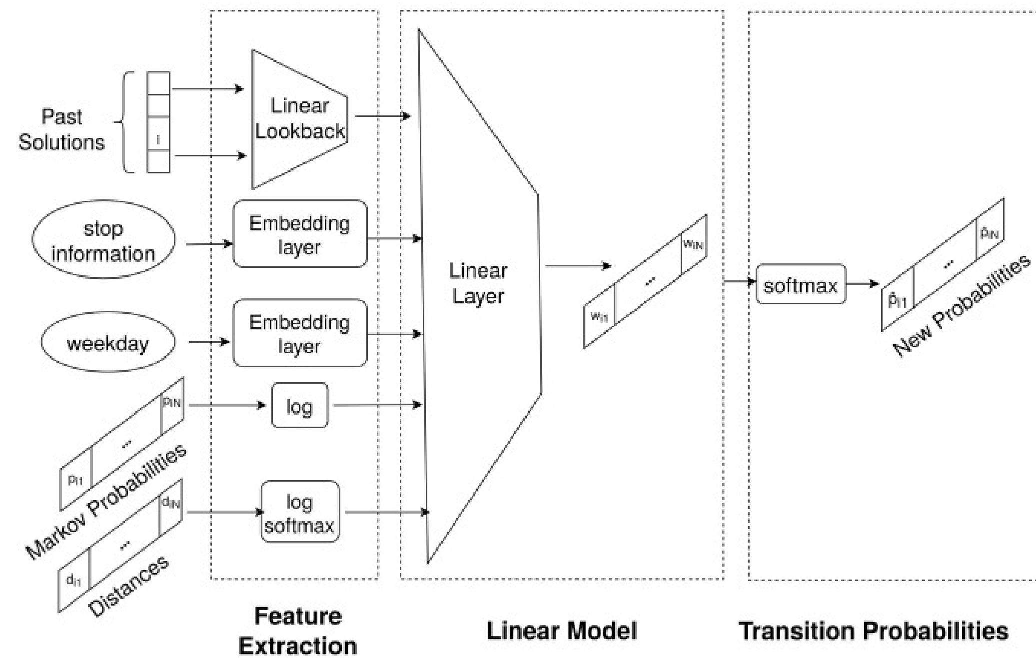
- Need to predict n*n outputs
- What input representation?
  - Encoding the (variable amount) of stops
  - Encoding all n*n distances
  - Encoding the temporal historic instances
- What loss function?
- Only few data (large networks will overfit)

[Mandi, Canoy, Bucarey, Guns, Data Driven VRP: ..., CP21]

Neural representation



- Key ideas:
  - domain-specific architecture,
  - 1 source→all stops
- Contextual features:



[Mandi, Canoy, Bucarey, Guns, Data Driven VRP: ..., CP21]

Loss function: Classification loss as proxy for arc-difference
(log likelihood of arcs in ‘preferred’ solution)

Summary comparison:

	CE	AD (%)	RD(%)	Distance
Markov Counting	2.44	18.55	17.26	418
Neural Net (Without linear lookback)	**1.04**	**18.04**	**17.02**	414
Distance based VRP	11.90	73.14	46.93	**366**

[Mandi, Canoy, Bucarey, Guns, Data Driven VRP: ..., CP21]

Learning the preferences

= imitation of user choices → copying, not *intelligence*?

Optimisation software is meant to do *better* than a user

(by considering larger nr of candidates and better resolving of conflicts)

I prefer route X even if it is 2 kilometers longer

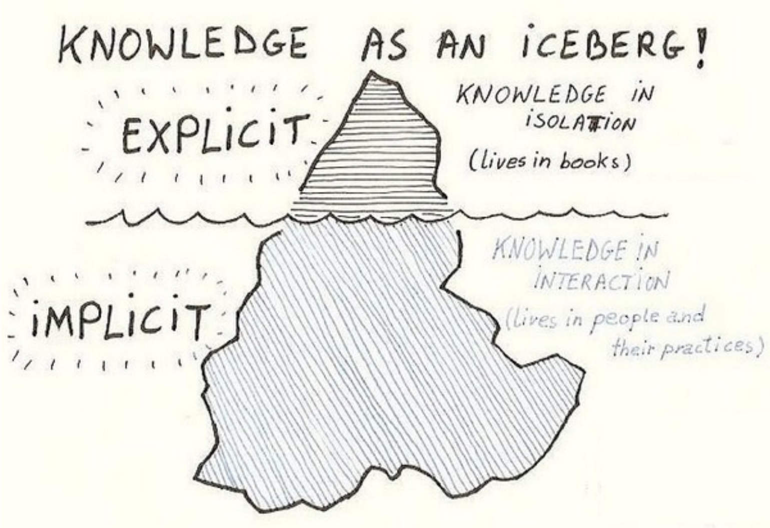
→ trade's off distance versus preference

Give control to combine both: $t'_{ij} = \beta t_{ij} + (1 - \beta) d_{ij}\,.$

Prediction + constraint solving

- Part explicit knowledge:
  in a formal language

- Part implicit knowledge:
  learned from data



» perception *(vision, natural language, ...)*

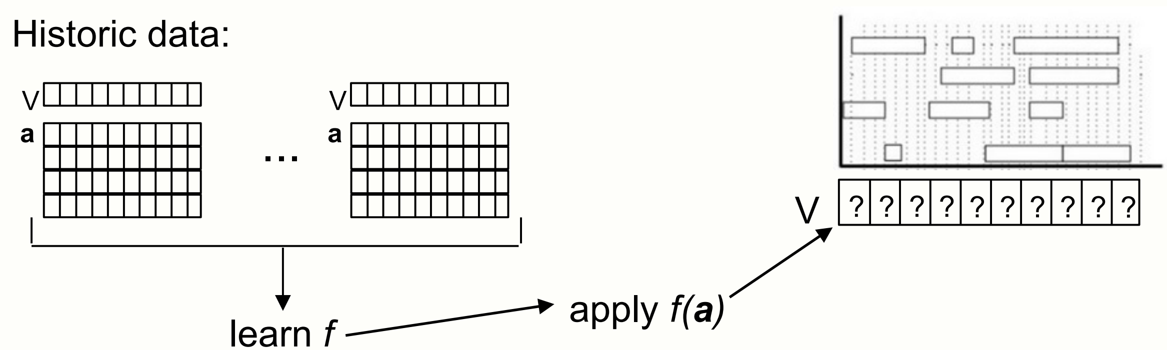
» tacit knowledge *(user preferences, social, ...)*

» **complex environment** *(demand, prices, ...)*

Time for end-to-end training!

Learn the objective function

Prediction + Optimisation (regression of weights)

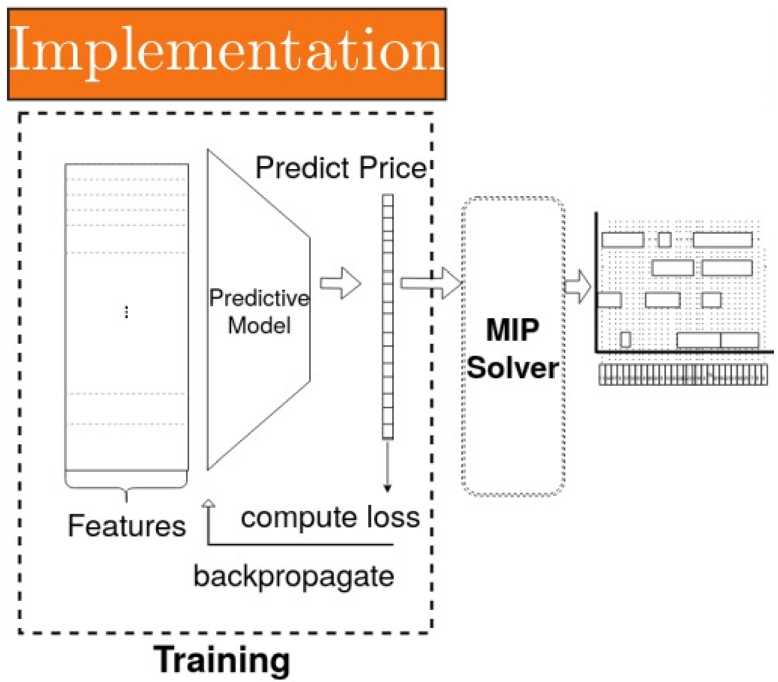


Predict: hourly energy prices + Optimize: energy-aware scheduling

Other examples:

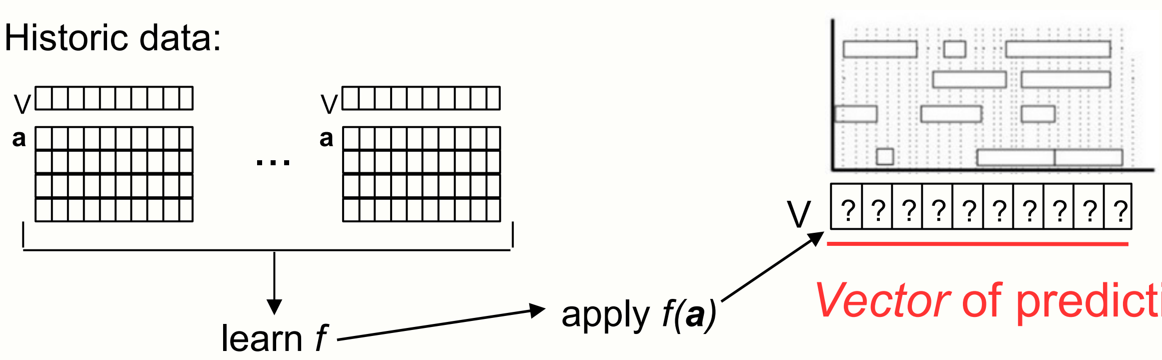
- Optimize steel plant production waste, by predicting steel defects
- Optimize money transport, by predicting value of coins at clients
- ...

prediction-focussed **regression**



Pre-trained neural network

MSE loss not the best proxy for *task* loss....



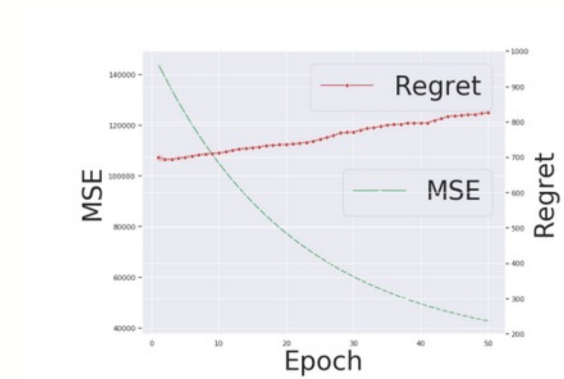
Joint inference: trades off the individual predictions

Vector of predictions

Why?

- MSE = average of individual errors of the vector
- Joint inference = *joint* error

  → some errors worse than others!



Which errors worse?

is combinatorial, need to *solve* to know

features   true cost vector

$$\underset{\omega}{\operatorname{argmin}}\, \mathbb{E}\left[regret\left(\underbrace{m(\overline{x_i};\omega)}_{\text{predicted cost vector}}, \overline{c_i}\right)\right]$$

network params

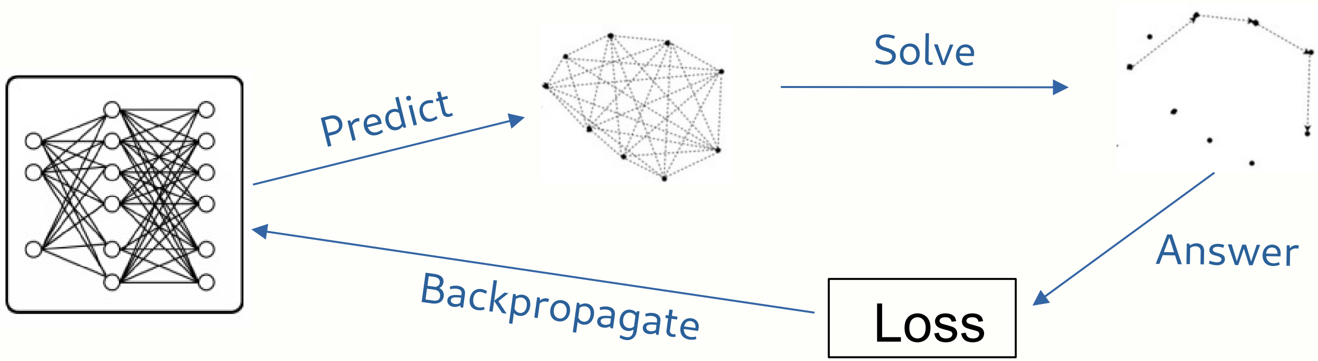
predicted cost vector

$$\begin{aligned} regret(\hat{c}, c) &= f(\hat{v}, c) - f(v^*, c) \\ with \quad v^* &= argmin_{v \in V} f(v, c) \\ \hat{v} &= argmin_{v \in V} f(v, \hat{c}) \end{aligned}$$

Challenges:
- no explicit gradient
- V is implicit, exponential size
- argmin f may be NP-hard

[Smart Predict-and-Optimize for Hard Combinatorial Optimization Problems, Jayanta Mandi, Emir Demirovic, Peter Stuckey, Tias Guns. AAAI20]

Learning approaches (gradient descent)



Key challenges:

1) suitable loss function? (non-differentiable solver)

2) scalability due to repeated solving:

Related work for discrete optimisation

- Differentiating KKT of a relaxed (QP) problem [Wilder, B., Dilkina, B., & Tambe, M. (2019, July)., Ferber, A., Wilder, B., Dilkina, B., & Tambe, M. (2020, April)]
- Differentiating HSD of a relaxed (LP) problem [Mandi, J., & Guns, T. (2020)]
- Subgradient of a surrogate loss [Elmachtoub, A. N., & Grigas, P. (2022), Mulamba, M. & Mandi, J. & Diligenti, M. & Lombardi, M. & Bucarey, V. & Guns, T.]
- Differentiation by perturbation [Pogančić, Marin Vlastelica, et al. (2020), Niepert, M., Minervini, P., & Franceschi, L. (2021)]

Decision-focused learning

Suitable loss function?

Key observation:

“The objective function induces a ranking over feasible solutions”

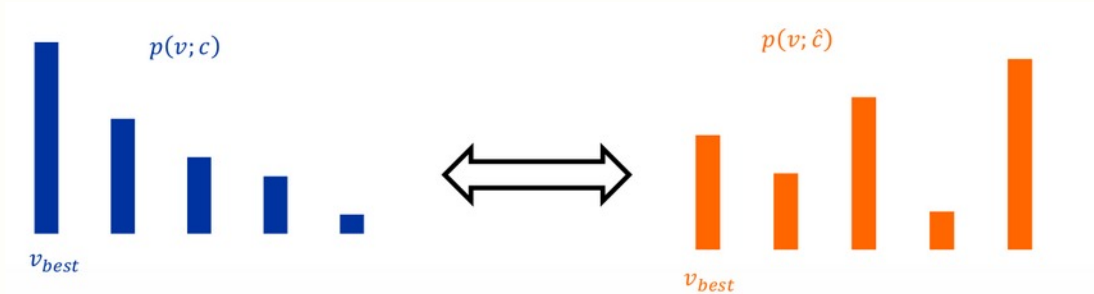
	Obj with true costs	Obj with predicted
Sol 1 [a,c,b,d,a]	12 (rank: 1)	14 (rank: 3)
Sol 2 [a,b,c,d,a]	15 (rank: 2)	10 (rank: 1)
Sol 3 [a,c,d,b,a]	16 (rank: 3)	11 (rank: 2)
Sol 4 [a,d,b,c,a]	23 (rank: 4)	16 (rank: 4)
Sol 5 [a,d,c,b,a]	28 (rank: 5)	18 (rank: 5)

[Mandi, Mulamba, Bucarey, Guns, Decision-focused learning: through the lens of learning to rank, ICML2022]

Decision-focused learning

Assume a set of feasible solutions S.

“The objective function induces a ranking over feasible solutions”



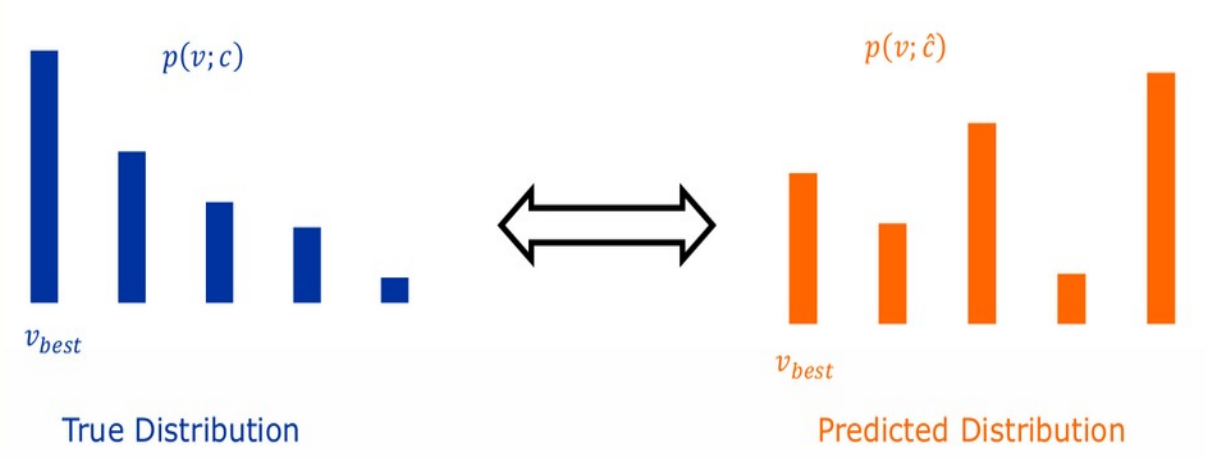
=> We can now use techniques from the much more mature ‘Learning to Rank’ field in ML!

[Mandi, Mulamba, Bucarey, Guns, Decision-focused learning: through the lens of learning to rank, ICML2022]

Listwise Learning 2 Rank for DFL

Discrete exponential distribution in solution space

$$p(v;c) = \begin{cases} \frac{1}{Z} exp(-f(v,c)/\tau) & v \in V \\ 0 & v \notin V \end{cases}$$



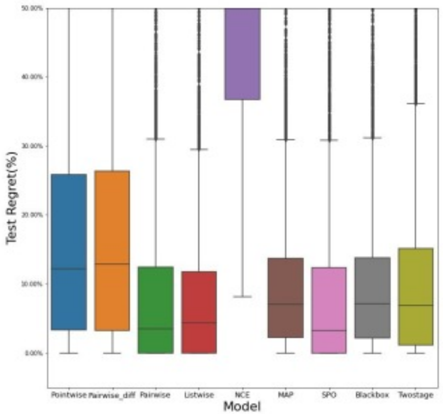
We obtain 2 empirical distributions (one for true costs, one for predicted) over a finite sample of feasible solutions S

=> Can use the standard Kullback-Leibler Divergence loss!

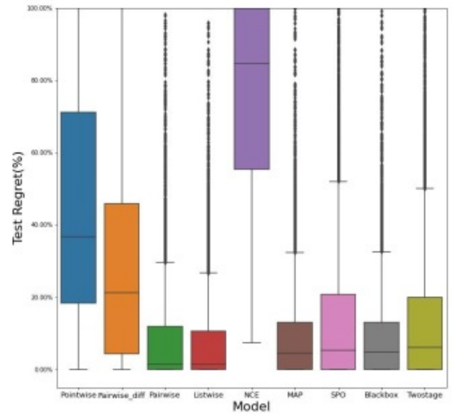
[Mandi, Mulamba, Bucarey, Guns, Decision-focused learning: through the lens of learning to rank, ICML2022]

Results

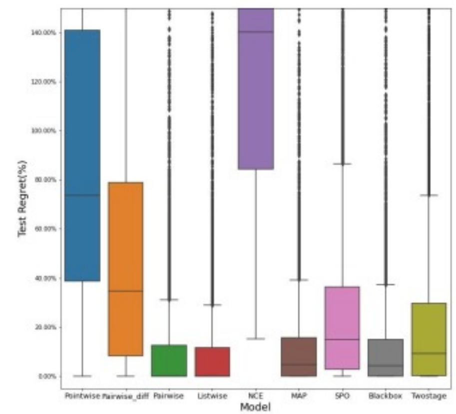
Shortest Path Problem



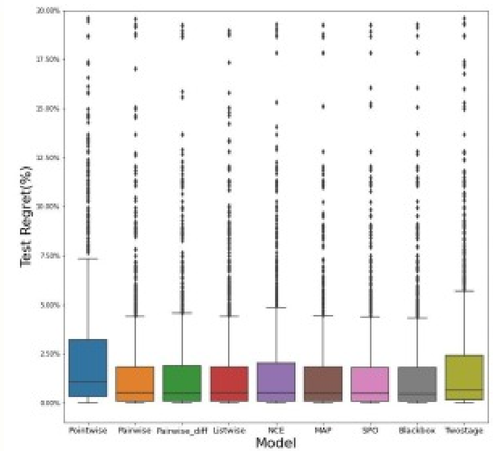
Degree 4

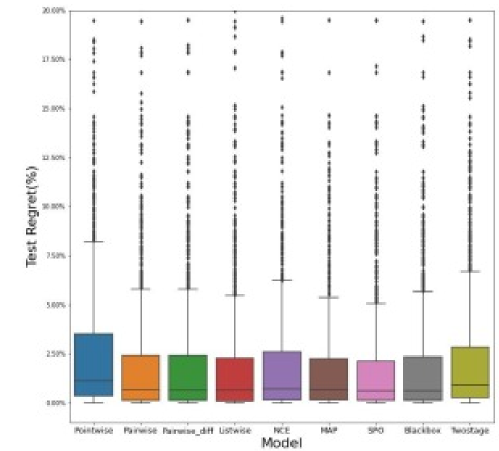


Degree 6



Degree 8

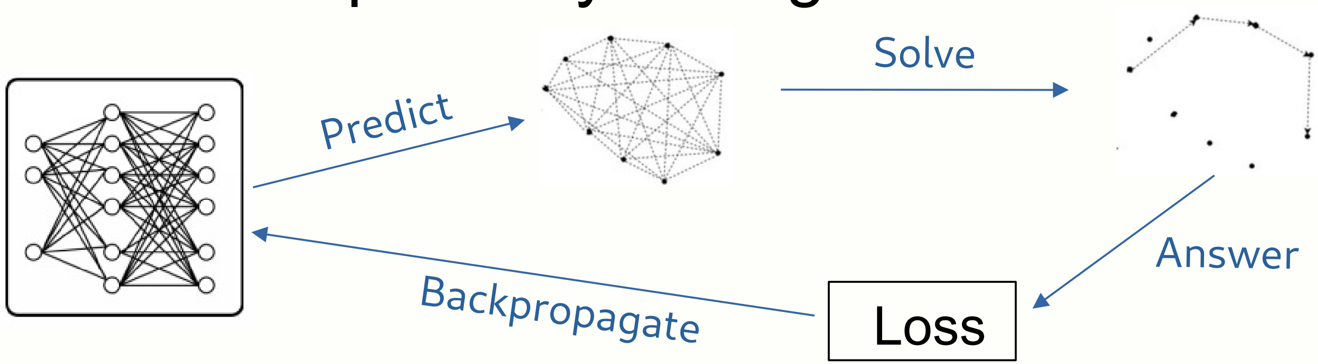




Scheduling Problem

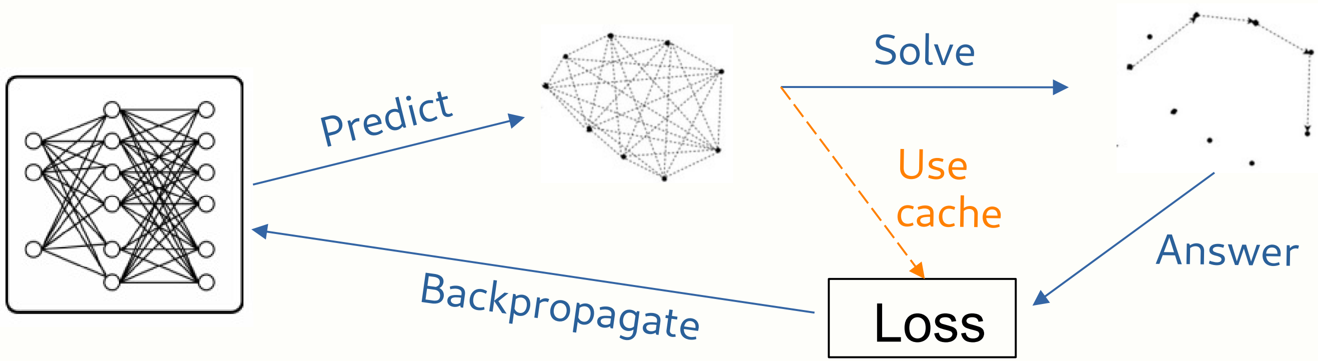
Decision-focused learning with L2R

2nd Key bottleneck: repeatedly calling the solver



Decision-focused learning with L2R

2nd Key bottleneck: repeatedly calling the solver



Can use *cached solutions* as approximate solver!!

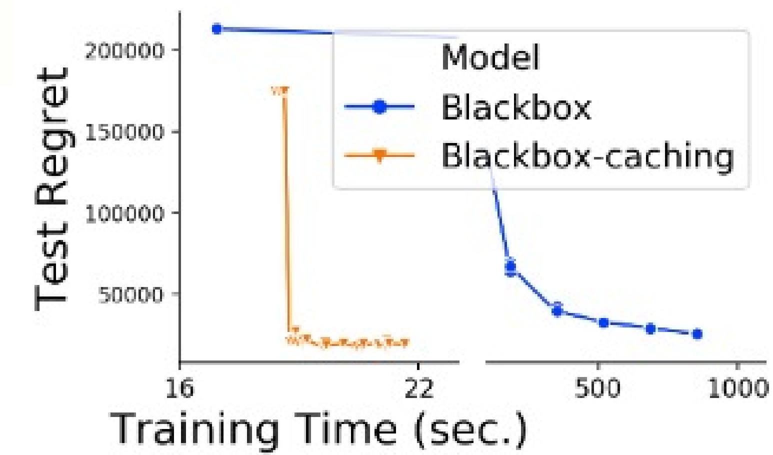
These cached solutions are the feasible set S

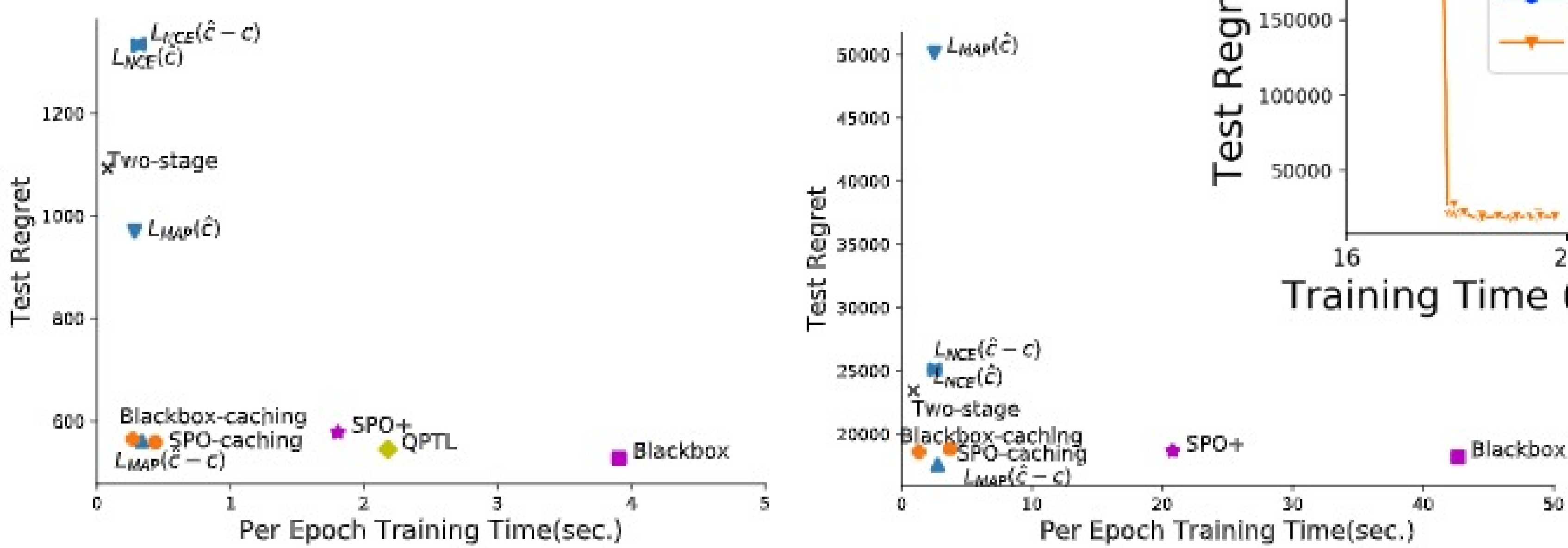
(also: sampling schemes: call the solver only 10% of the times)

[Mulamba, Mandi, Bucarey, Guns, Contrastive Losses and Solution Caching for Predict-and-Optimize, IJCAI2021]

Results

Caching scheme compatible with **all** methods that call a blackbox solver *(call the cache instead, 90% of time)*





(a) Knapsack-120 (b) Energy-3

Implementation in gradient descent loop

Standard:

Algorithm 1: Stochastic gradient descent

Input : training data $\mathcal{D} = \{X, y\}_{i=1}^{n}$, learning rate γ

initialize θ (neural network weights)
for *epochs* **do**
  for *batches* **do**
    sample batch $(X, y) \sim \mathcal{D}$
    $\hat{y} \leftarrow g(z, \theta)$ (forward pass: compute predictions)
    Compute loss $L(y, \hat{y})$ and gradient $\frac{\partial L}{\partial \theta}$
    Update $\theta = \theta - \gamma \frac{\partial L}{\partial \theta}$ through backpropagation (backward pass)
  end
end

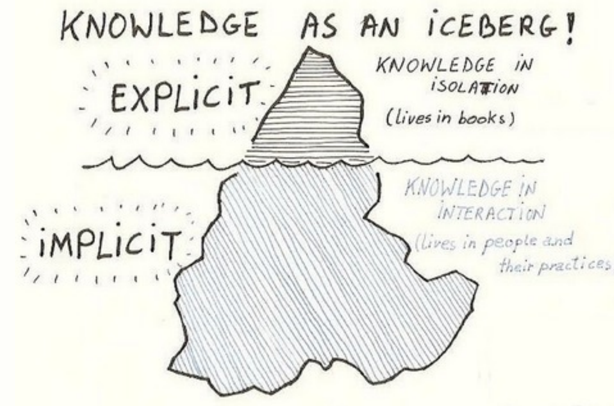
with Listwise ranking:

Algorithm 3: Stochastic gradient descent with KL on solutions

Input : training data $\mathcal{D} = \{X, y\}_{i=1}^{n}$, architecture g, learning rate γ, sample rate r

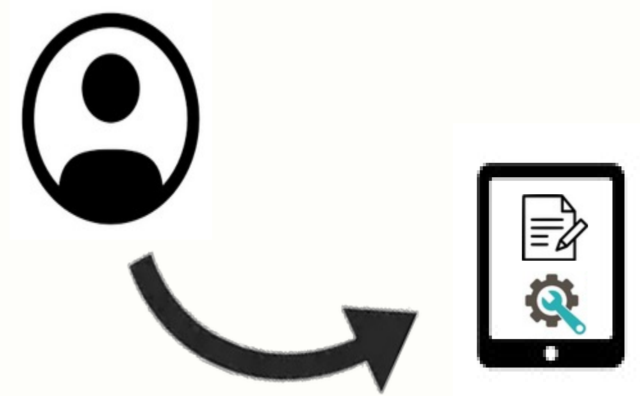
initialize θ (neural network weights of g)
sols $\leftarrow \{solver(y) \mid (X, y) \in \mathcal{D}\}$ (initialize with true solutions)
for *epochs* **do**
  for *batches* **do**
    sample batch $(X, y) \sim \mathcal{D}$
    $\hat{y} \leftarrow g(X, \theta)$ (forward pass: compute predictions)
    if *random()* $\leq r$ **then**
      sols $\leftarrow$ sols $\cup \{solver(\hat{y})\}$
    end
    Compute loss L $=$ KL$(distr(y, sols), distr(\hat{y}, sols))$ and grad. $\frac{\partial L}{\partial \theta}$
    Update $\theta = \theta - \gamma \frac{\partial L}{\partial \theta}$ through backpropagation (backward pass)
  end
end

Key take-aways:



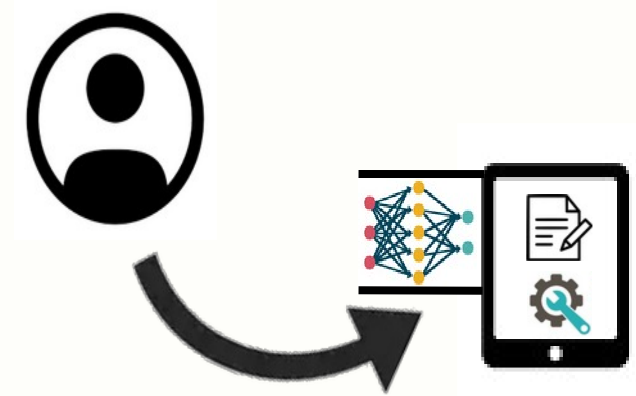
- <u>Explicit</u> knowledge: use solver
- <u>Implicit</u> knowledge: do learning
- Joint inference / collective classification:
  maximize log likelihood!
- Keep revisiting the solving AND the learning, hybridize and use properties of one in the other!
- Prediction + Optimisation with decision-focussed learning possible

Bigger picture



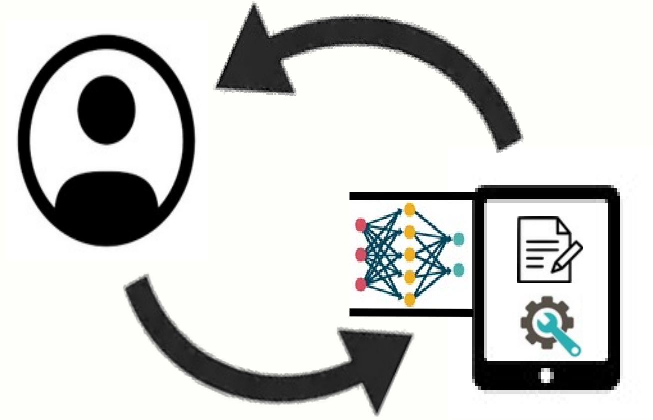
Bigger picture

- Learning implicit user preferences
- Learning from the environment



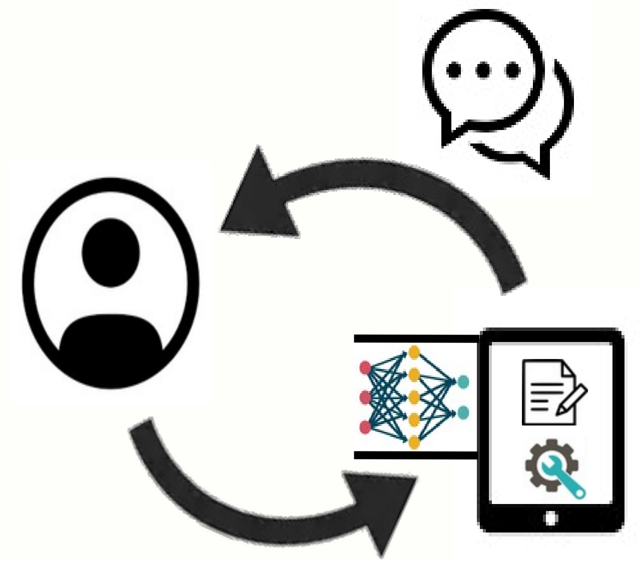
Bigger picture

- Learning implicit user preferences
- Learning from the environment
- Explaining constraint solving

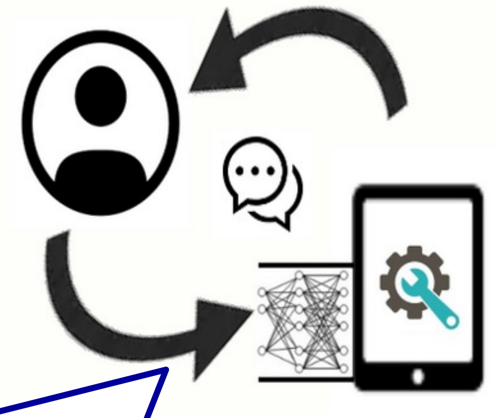


Bigger picture

- Learning implicit user preferences
- Learning from the environment
- Explaining constraint solving
- Stateful interaction



CHAT-Opt:
Conversational **H**uman-**A**ware **T**echnology for **Opt**imisation

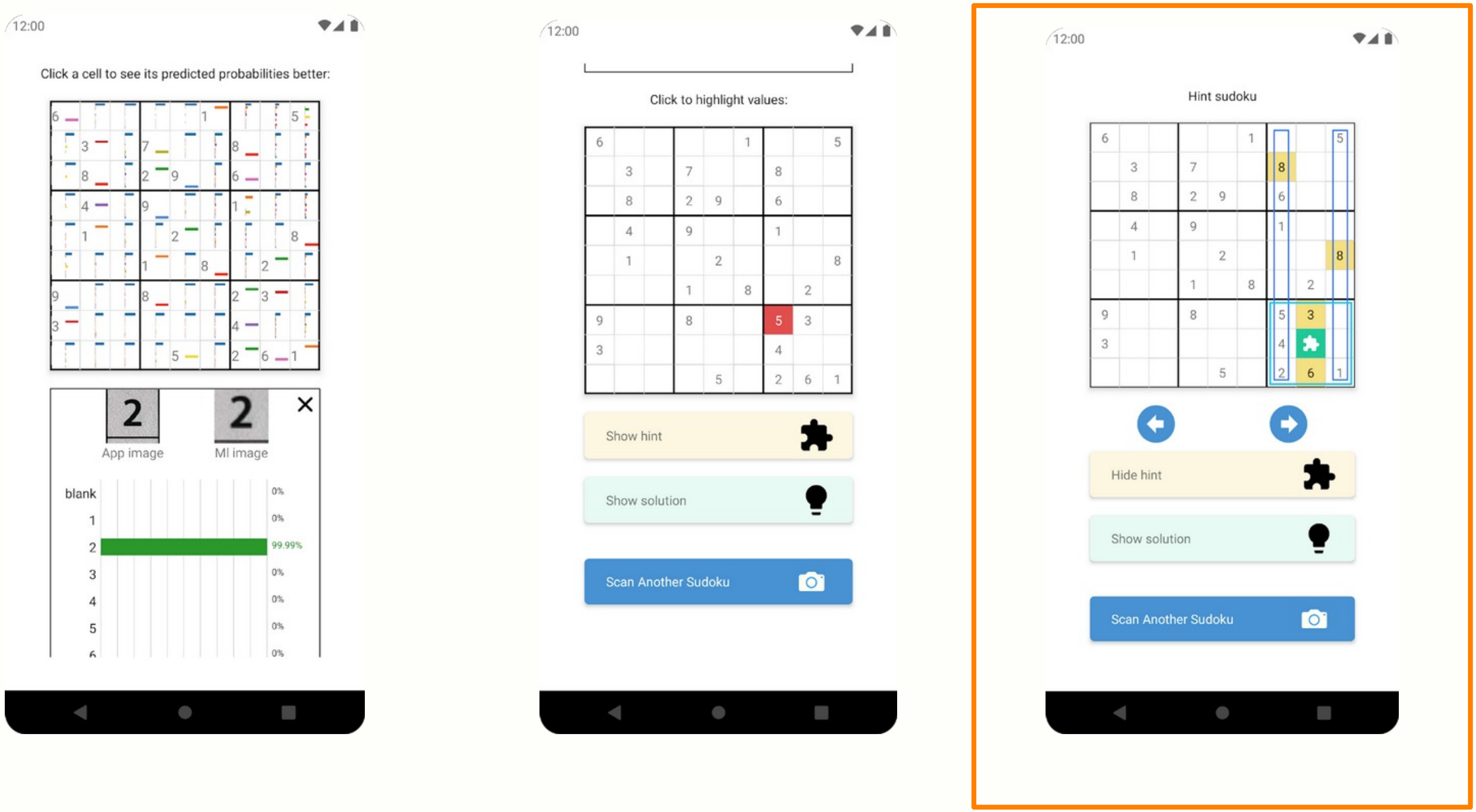


Towards **co-creation** of constraint optimisation solutions

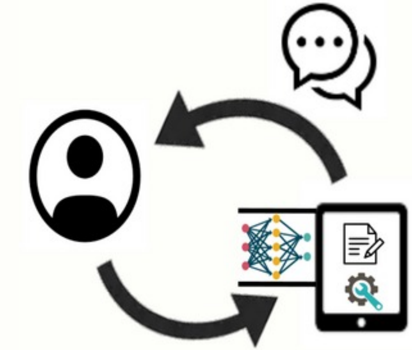
- Solver that learns from user and environment
- Towards conversational: explanations and stateful interaction

Hiring post-docs!

Sudoku Assistant, explanation steps



Conversational **H**uman-**A**ware **T**echnology for **Opt**imisation



Easy to integrate (Python) libraries

- Easy integration with Machine Learning libraries

  => Python and numpy arrays

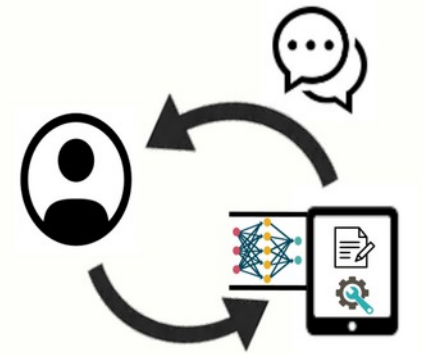
- Efficient repeated solving

  => Incremental

- Use CP/SAT/MIP or any combination

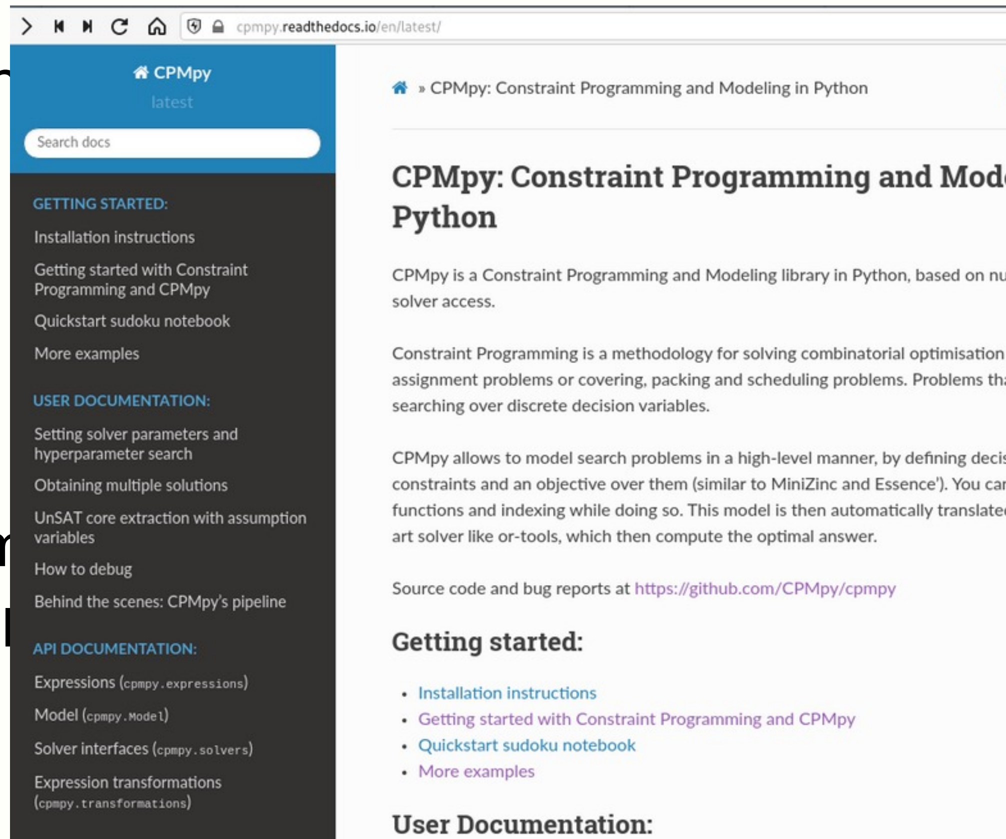
  => solver independent and multi-solver

Conversational **H**uman-**A**ware **T**echnology for **Opt**imisation



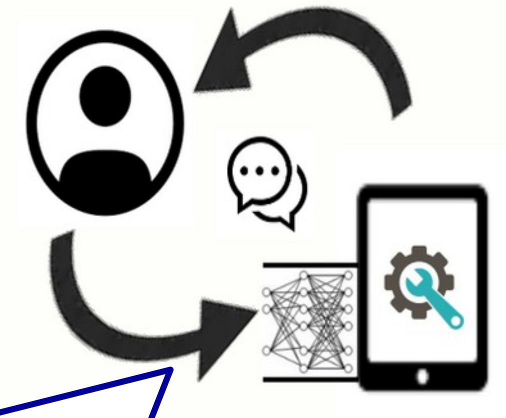
Easy to integrate (Python) libraries

- Easy integration with Machin
  => Python and numpy arrays
- Efficient repeated solving
  => Incremental
- Use CP/SAT/MIP or any com
  => solver independent and mul



https://github.com/CPMpy/cpmpy

CHAT-Opt:
Conversational **H**uman-**A**ware **T**echnology for **Opt**imisation



Towards **co-creation** of constraint optimisation solutions

- Solver that learns from user and environment
- Towards conversational: explanations and stateful interaction

https://people.cs.kuleuven.be/~tias.guns

@TiasGuns

Hiring post-docs!