Operations research and machine learning

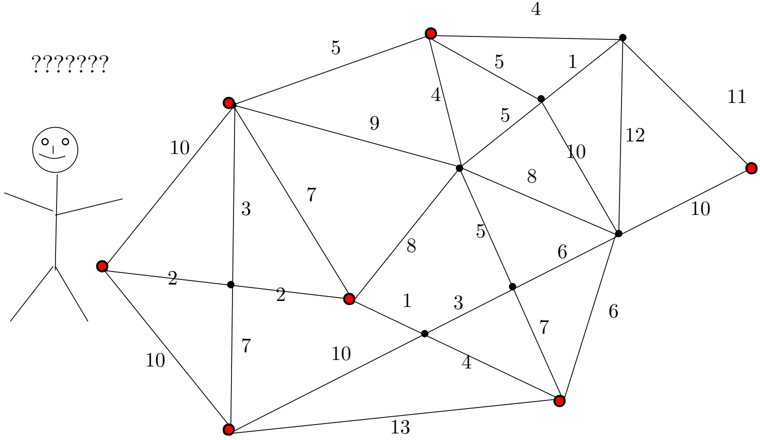
Axel Parmentier

CERMICS

May 16th, 2019

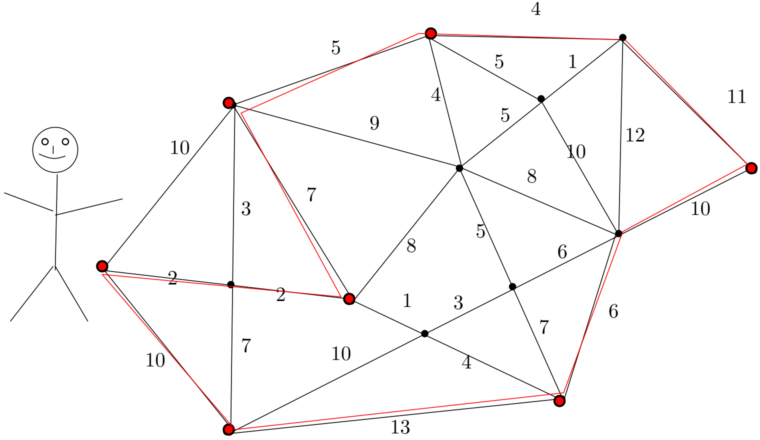
IRT SystemX

Traveling salesman problem



Mister Supersales must plan his tour

Traveling salesman problem



Mister supersale has planned his tour

Is there an optimal solution?

Is there an optimal solution?

Yes: finite set of solution

Can we enumerate all the solutions?

Is there an optimal solution?

Yes: finite set of solution

Can we enumerate all the solutions?

With 25 cities, we have $24! = 24 \times 23 \times 22 \times \cdots \times 2 \times 1$ possibilities, that is, around 6.204×10^{25} possibilities.

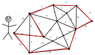
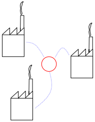
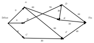
Using paper and pencil, testing 1 possibility per second, requires around 1.976×10^{16} years.

Testing 1 million possibilities per second with a computer, requires 19 billion years.

The traveling salesman problem is one of the most famous Operations Research problem.

Operations Research (OR):
mathematical discipline that deals with the optimal allocation of resources (typically in firms).

mathematical part of decision science

- Find the best tour 
- Plan the best timetable 
- Find the most resilient network 
- Fill a container optimally 
- Locate facilities/warehouses optimally 
- Schedule jobs on machines 

Operations Research in practice:

1. Model the question studied as an optimization
2. Choose/design an algorithm to solve the optimization problem
3. Interpret the results

Operations Research in practice:

1. Model the question studied as an optimization
2. Choose/design an algorithm to solve the optimization problem
3. Interpret the results

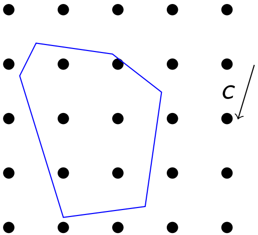
Roughly speaking:

- OR researchers focus on 2
- OR engineers / users know which algorithms work for 2 and spend their time on 1 and 3

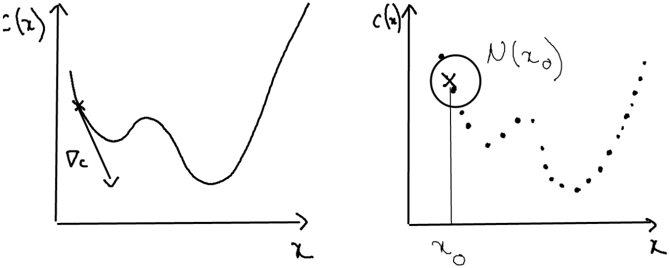
Scope of applications of the field: its versatile and powerful algorithms

Mixed integer linear programming

$$\begin{aligned} \min \quad & cx \\ \text{s.t.} \quad & Ax = b \\ & x \in \mathbb{R}_+^n \times \mathbb{Z}_+^p \end{aligned}$$



MILP with up to 10^6 variables solved to optimality in the industry



Define a neighborhood to replace gradient

What is Machine Learning

Machine learning uses data to:

1. extract information from data (unsupervised learning)

   "There is A and B"

2. make predictions (supervised learning)

   "If A happens, then B with happen"

3. take decisions (reinforcement learning)

   "If I want B to happen, then I should do A"

Machine learning uses data to:

1. extract information from data (unsupervised learning)

   "There is A and B"

2. make predictions (supervised learning)

   "If A happens, then B with happen"

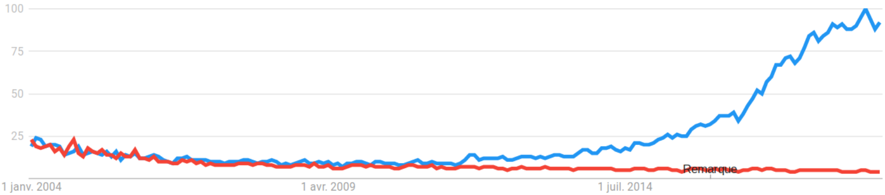
3. take decisions (reinforcement learning)

   "If I want B to happen, then I should do A"

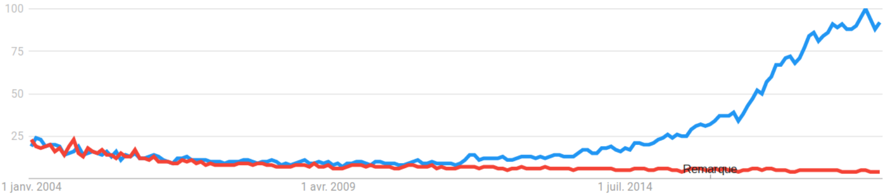
Which of OR and ML has the largest scope?

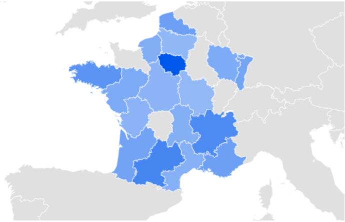
ML vs OR beauty pageant

Google trends data (popularity = proportion of the research)
Machine Learning (blue) vs Operations Research (red)



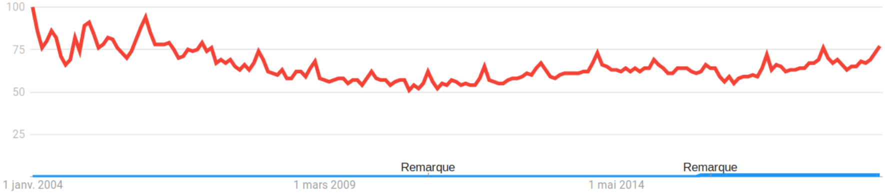
Google trends data (popularity = proportion of the research)
Machine Learning (blue) vs Operations Research (red)



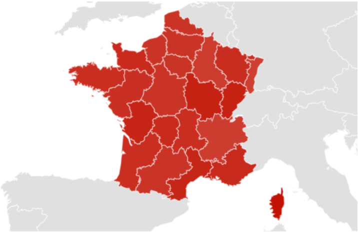


Trendy does not mean relevant: ML is not the king of prediction

Machine learning (blue) vs horoscope (red)



Machine learning (blue) vs horoscope (red)



Many "artificial intelligence" problems are operations research problems.

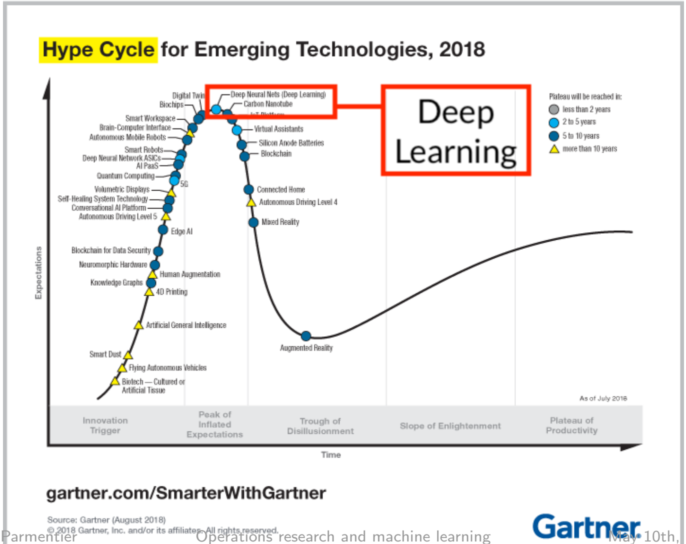
Why trends matters

Many "artificial intelligence" problems are operations research problems.

Many firms don't known the existence of OR:

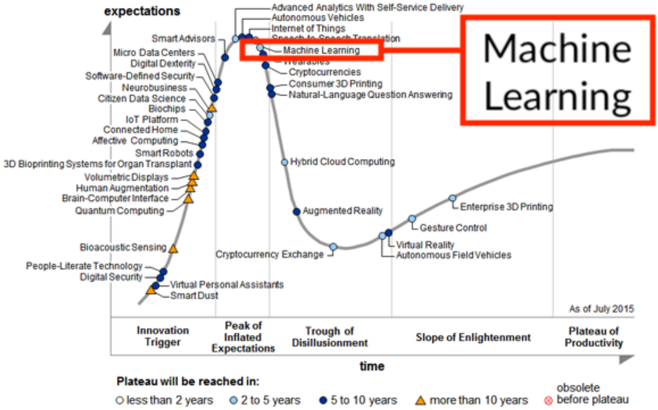
- hire ML consultants that don't know OR on OR problems

ML takes today our projects / students, can we just wait that ML gets old-fashioned?



ML is more than a trend: years at peak expectations

2015



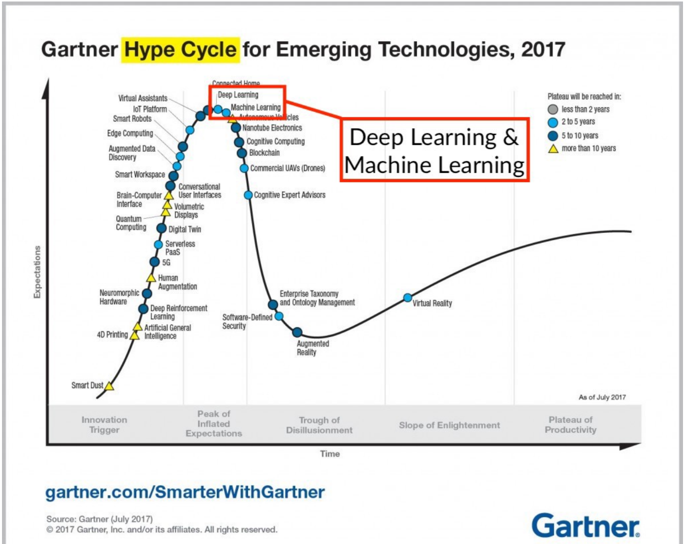
ML is more than a trend: years at peak expectations

2016

Source: Gartner (July 2016)

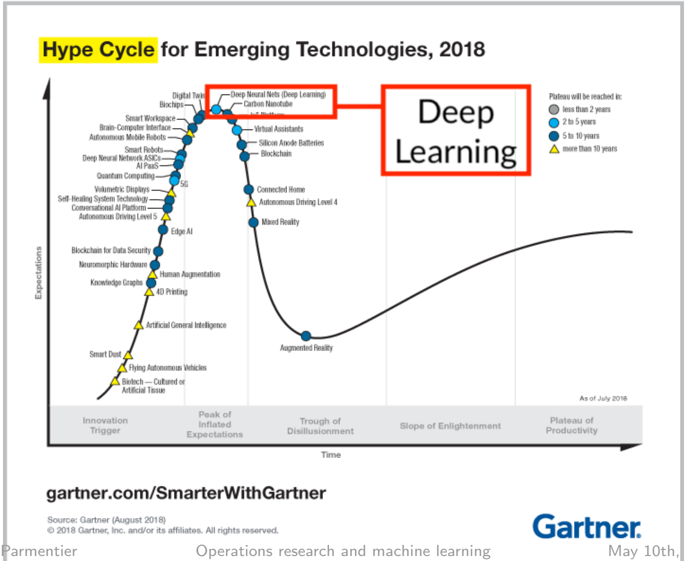
ML is more than a trend: years at peak expectations

2017



ML is more than a trend: years at peak expectations

2018



Basic machine learning should be in the toolbox of all scientists

Today:

- Experience is at the heart of scientific method to *validate model*
- Basic statistics is an important skill for every scientist/engineer

Today:

- Experience is at the heart of scientific method to *validate model*
- Basic statistics is an important skill for every scientist/engineer

Machine Learning provides statistics tools for the age of big data.

- Data won't disappear
- Making predictions will remain at the heart of scientific method
- Machine Learning provides "user friendly default methods"
- Basic machine learning will be:
  - an important skill for every scientist/engineer
  - required in the toolbox of the OR practitioner

Whether you want to

- implement your strategy
- learn from the data
- approximate your partial differential equation
- etc.

in the end, you will want to find a good/the best solution.

Whether you want to

- implement your strategy
- learn from the data
- approximate your partial differential equation
- etc.

in the end, you will want to find a good/the best solution.

and if you problem is of (even moderately) large scale:

- need an optimization algorithm
- need operations research if your problem is discrete

Today talk on what happens at the crossroad of OR and ML

1. OR for ML
2. ML for OR
3. Data driven optimization

Today talk on what happens at the crossroad of OR and ML

1. OR for ML
2. ML for OR
3. Data driven optimization

Machine learning in practice:

- Model the problem using a statistical model (e.g. neural network)
- Learn the model = solve an optimization problem
- Interpret the results

"Learning" in ML:

- Modeling: formulate an optimization problem (with good statistical properties)
- Optimization: solve it

Bertsimas "Machine Learning under a Modern Optimization Lens" papers

`http://www.mit.edu/~dbertsim/papers.html#MachineLearning`

Mathematical programming at the cross-road of OR and ML

"Learning" in ML:

- Modeling: formulate an optimization problem (with good statistical properties)
- Optimization: solve it

ML "good algorithms":

- good generalization,
- simple and easy implementation,
- fast convergence to an approximate solution

OR "good algorithms":

- optimality / solution quality
- apply to a wide class of problem (MILP)
- speed

And many shared objectives: scalability to large problems, fast convergence to an approximate solution

Table of Content

Part content

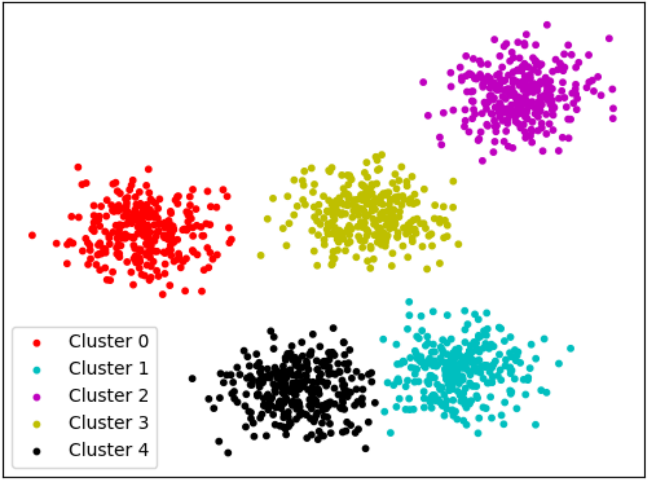
Given samples of data $x_1, \ldots, x_n$ in $\mathbb{R}^n$, detect some structure in the data

- Clustering
- Dimensionality reduction
- Learning a distribution.
- Anomaly detection
- Generative Adversarial Networks
- etc.

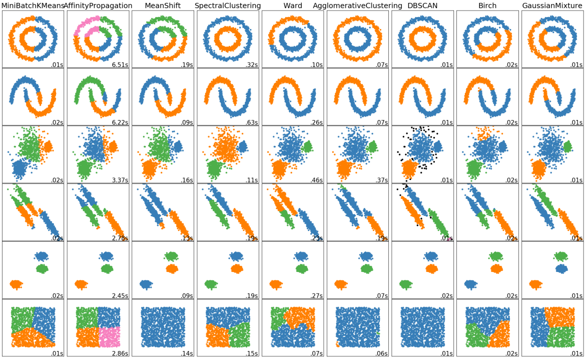
Extract structure from the data

Clustering

Given $x_1, \ldots, x_n$ in $\mathbb{R}^d$, partition them into k clusters of "similar" points.



Given $x_1, \ldots, x_n$ in $\mathbb{R}^d$, partition them into k clusters of "similar" points.



Compare different methods on your dataset using `sci-kit learn`

Supervised learning problem

Use previously seen data $(x_1, y_1), \ldots, (x_n, y_n)$ to

predict y given a new x

Practically

$$y = f_\theta(x)$$

with f_θ in a statistical model $\{f_\theta, \theta \in \Theta\}$

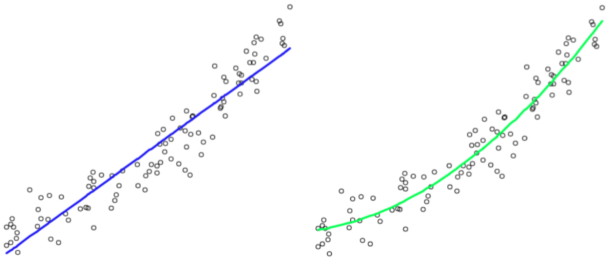
Use previously seen data $(x_1, y_1), \ldots, (x_n, y_n)$ to

predict y given a new x

Practically

$$y = f_\theta(x)$$

with f_θ in a statistical model $\left\{ f_\theta, \theta \in \Theta \right\}$



Supervised learning problem

Use previously seen data $(x_1, y_1), \ldots, (x_n, y_n)$ to

predict y given a new x

Practically

$$y = f_\theta(x)$$

with f_θ in a statistical model $\left\{f_\theta, \theta \in \Theta\right\}$

Learning the problem: find θ

$$\min_\theta \sum_{i=1}^n \|y_i - f_\theta(x_i)\|^2$$

Output types and supervised learning problems

Given $(x_1, y_1), \ldots, (x_n, y_n)$, learn

$$y = f_\theta(x)$$

Quantitative variables

$X \in \mathbb{R}$ or in a subset of $\mathbb{R}$

Qualitative variables

X in a non-ordered finite set.

- Quantitative or qualitative *inputs*: pre-processing
- Quantitative or qualitative *outputs*: influence the learning method

Output type	Quantitative	Qualitative
Problem type	*Regression*	*Classification*

Binary classification, where $y \in \{0, 1\}$, is especially useful

$$y = f_\theta(x)$$

Kind of x that can be used:

- vectors in $\mathbb{R}^d$
- qualitative variables
- time series
- images
- videos
- graphs

$$y = f_\theta(x)$$

Kind of x that can be used:

- vectors in $\mathbb{R}^d$
- qualitative variables
- time series
- images
- videos
- graphs

In practice:

$$y = f_\theta(x)$$

Kind of x that can be used:

- vectors in $\mathbb{R}^d$
- qualitative variables
- time series
- images
- videos
- graphs

In practice:

tune you features $\phi_k(x)$:

$$y = \sum_k \theta_k \phi_k(x)$$

Inputs and supervised learning problem

$$y = f_\theta(x)$$

Kind of x that can be used:

- vectors in $\mathbb{R}^d$
- qualitative variables
- time series
- images
- videos
- graphs

In practice:

tune you features $\phi_k(x)$:

$$y = \sum_k \theta_k \phi_k(x)$$

use sparse learning (e.g. lasso) to *identify the most relevant features*

$$\min_\theta \sum_{i=1}^n \|y_i - f_\theta(x_i)\|^2 + \lambda|\theta|$$

A problem that you can model by *dynamic programming*

$$V_t(x) = \max_a \sum_{x'} p(x,x')\Big(c(x,a,x') + V_{t+1}(x')\Big)$$

but not solve because x lives in an exponentially large state space

A problem that you can model by *dynamic programming*

$$V_t(x) = \max_a \sum_{x'} p(x,x')\Big(c(x,a,x') + V_{t+1}(x')\Big)$$

but not solve because x lives in an exponentially large state space

Reinforcement learning is an approximate dynamic programming method that

- uses an value function of the form $f_\theta(x) \simeq V_t(x)$
- learn θ through simulation

Part content

Main message

Machine Learning can predict many interesting informations that you can exploit in your solution schemes.

Main message

Machine Learning can predict many interesting informations that you can exploit in your solution schemes.

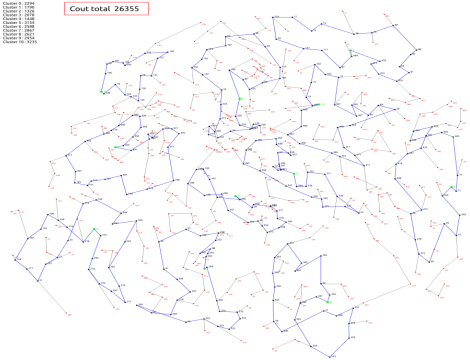
Today, examples on:

- Heuristics
- MILPs

Learn the structure of the instance for cheap initialization

Many OR problems are partitioning problems

- sometimes, a rather good heuristic can be found using *clustering*
- initialize a heuristic

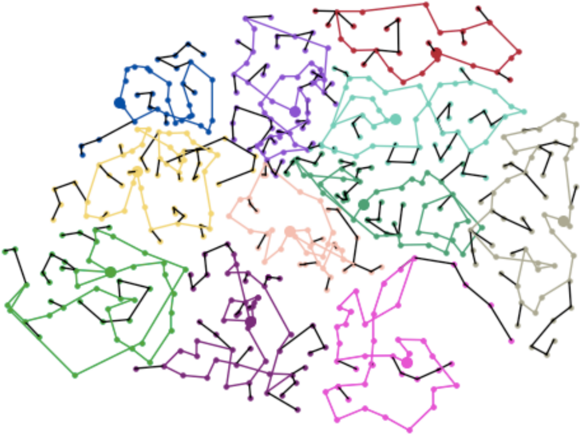


Many OR problems are partitioning problems

- sometimes, a rather good heuristic can be found using *clustering*
- initialize a heuristic

Many OR problems are partitioning problems

- sometimes, a rather good heuristic can be found using *clustering*
- initialize a heuristic



Learning parameters: well known in the heuristics toolbox

Learning where to initialize a local search

- Zhou, Y., Hao, J. K., & Duval, B. (2016). Reinforcement learning based local search for grouping problems: A case study on graph coloring. Expert Systems with Applications, 64, 412-422.

- Initialize a solution with a given prior probability
- Run a local search form this initial solution
- Update the prior probability

Learning parameters: well known in the heuristics toolbox

Learning where to initialize a local search

- Zhou, Y., Hao, J. K., & Duval, B. (2016). Reinforcement learning based local search for grouping problems: A case study on graph coloring. Expert Systems with Applications, 64, 412-422.

- Initialize a solution with a given prior probability
- Run a local search form this initial solution
- Update the prior probability

Learning which neighborhood to use in a large neighborhood search

Learning where to initialize a local search

- Zhou, Y., Hao, J. K., & Duval, B. (2016). Reinforcement learning based local search for grouping problems: A case study on graph coloring. Expert Systems with Applications, 64, 412-422.

- Initialize a solution with a given prior probability
- Run a local search form this initial solution
- Update the prior probability

Learning which neighborhood to use in a large neighborhood search

Many of these ideas already in

- Gardi, F., Benoist, T., Darlay, J., Estellon, B., & Megel, R. (2014). Mathematical Programming Solver Based on Local Search. John Wiley & Sons.

Learning property of good solutions enables to know where to search.

Learn where to search in local search

Learning property of good solutions enables to know where to search.

Heuristics for the VRP:

- Arnold, F., & Sörensen, K. (2018). What makes a VRP solution good? The generation of problem-specific knowledge for heuristics. Computers & Operations Research.
- Arnold, F., & Sörensen, K. (2019). Knowledge-guided local search for the vehicle routing problem. Computers & Operations Research, 105, 32-46.

Learning property of good solutions enables to know where to search.
Heuristics for the VRP:

- Arnold, F., & Sörensen, K. (2018). What makes a VRP solution good? The generation of problem-specific knowledge for heuristics. Computers & Operations Research.
- Arnold, F., & Sörensen, K. (2019). Knowledge-guided local search for the vehicle routing problem. Computers & Operations Research, 105, 32-46.

Methodology:

- Generate a large set of instances and feasible solutions
- Train a supervised learning classifier to predict if a feasible solution is near optimal or not
- Use this knowledge to guide local-search

Supervised learning prediction on MILPs

$$\begin{aligned} \min \;& cx \\ \text{s.t.}\;& Ax \leq b \\ & x \in \mathbb{Z}^d \times \mathbb{R}^{n-d} \end{aligned}$$

Can we predict accurately:

- Optimal solution of MILPs?

Supervised learning prediction on MILPs

$$\begin{aligned} \min \;& cx \\ \text{s.t.}\;& Ax \leq b \\ & x \in \mathbb{Z}^d \times \mathbb{R}^{n-d} \end{aligned}$$

Can we predict accurately:

- Optimal solution of MILPs? no
- Value of MILPs? no
- Solution time? no

or at least not yet

Supervised learning prediction on MILPs

$$\begin{aligned} \min \quad & cx \\ \text{s.t.} \quad & Ax \leq b \\ & x \in \mathbb{Z}^d \times \mathbb{R}^{n-d} \end{aligned}$$

Can we predict accurately:

- Optimal solution of MILPs? no
- Value of MILPs? no
- Solution time? no

or at least not yet

Use supervised learning to predict in fractions of a second interesting statistics on MILPs

Survey:

- Bengio, Y., Lodi, A., & Prouvost, A. (2018). Machine Learning for Combinatorial Optimization: a Methodological Tour d'Horizon. arXiv preprint arXiv:1811.06128.

What can we predict on general MILPs ?

Before launching the solver: interesting parameters

- Which Dantzig-Wolfe decomposition should be used?
  Kruber, M., Lübbecke, M. E., & Parmentier, A. (2017, June). Learning when to use a decomposition. In International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (pp. 202-210). Springer, Cham.

After a fraction of the solver time:

- If the solver will prove optimality within the time limit:
  Fischetti, M., Lodi, A., & Zarpellon, G. (2018). Learning MILP Resolution Outcomes Before Reaching Time-Limit.

Heuristic decisions all along the solution scheme:

- Learning where and how to branch:
  Lodi, A., & Zarpellon, G. (2017). On learning and branching: a survey. TOP, 25(2), 207-236.

A MILP $\mathcal{P}$:

$$\begin{aligned} \min \quad & cx \\ \text{s.t.} \quad & Ax = b \\ & x \in \mathbb{R}_+^n \times \mathbb{Z}_+^p \end{aligned}$$

- DWR for Mixed Integer Programs
- Solved by column generation

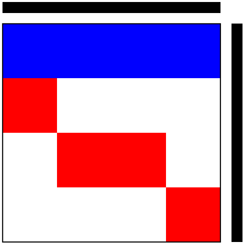


Figure: Decomposition $\mathcal{D}$ of $\mathcal{P}$

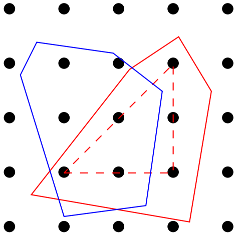
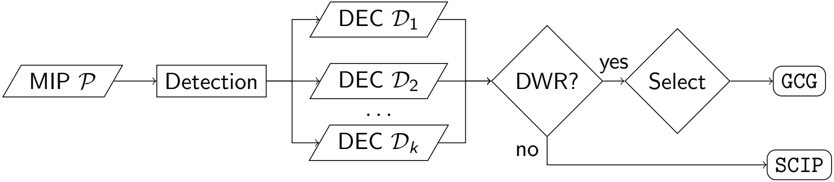


Figure: Tighter relaxation

Automatic Decomposition in GCG



A MIP can be forced in several types of decomposition:

- Border
- Staircase
- etc.

GCG performance highly depends on how well the decomposition catches the problem structure.

Our work: a supervised learning approach to *select the best decomposition* (using no decomposition is often the best answer).

Supervised learning binary classification problem

Binary classification problem

If it remains solution time t, is it better to solve using `GCG` with $\mathcal{D}$ or to use `SCIP` (no decomposition)?

Supervised learning binary classification problem

Binary classification problem

If it remains solution time t, is it better to solve using `GCG` with $\mathcal{D}$ or to use `SCIP` (no decomposition)?

$$\boxed{\begin{array}{c}\text{Input}\\ (\mathcal{P},\mathcal{D},t)\end{array}} \xrightarrow[\text{map } \phi]{\text{Feature}} \boxed{\begin{array}{c}\text{Features vector}\\ \phi(\mathcal{P},\mathcal{D},t)\in\mathbb{R}^d\end{array}} \xrightarrow[f_\theta]{\text{Classifier}} \boxed{\begin{array}{c}\text{Output}\\ f_\theta\left(\phi\left(\mathcal{P},\mathcal{D},t\right)\right)\end{array}}$$

Define ϕ
more than 80 features

Choose f *family*
Learn θ

Define features ϕ

Examples of features used:

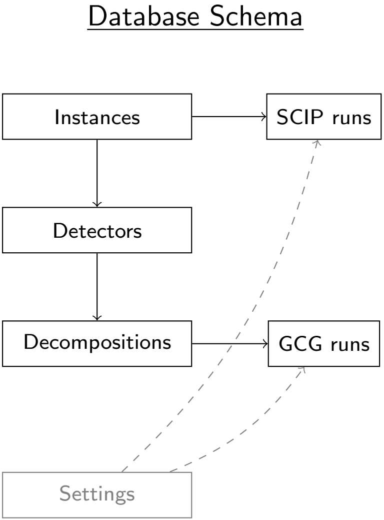
- Time t
- Nb variables/constraints
- Variable types
- Constraint types
- Products of features
- Nb linking variables / constraints
- Nb blocks
- min, max, mean block size
- Detector used (indicator)
- Detection quality metrics

`GCG` decomposition selection tool uses empirical detection quality metrics.

Learn the problem: generate database

- database
- python interface
- SCIP version 3.2.1, GCG 2.1.1.
  i7-2600 3.4GHz PC,
  8MB cache, 16GB RAM
- $\sim$ 135 days computing time

Crucial part, because a supervised learning scheme will work on certain kinds of inputs only if such inputs are in the training set



Distribution of Instances

SCIP results	all	clr	stcv	cpmp	sdlb	ctst	gap	ntlb	ltsz	bp	rap	stbl	cvrp	miplib
		structured												non-str
instances	400	25	25	25	25	25	25	25	25	25	25	25	25	100
opt. sol.	65.5%	19	3	18	10	25	23	25	25	6	12	22	6	68
feas. sol.	31.5%	6	21	7	11	-	2	-	-	19	12	3	19	26
no sol.	3.0%	-	1	-	4	-	-	-	-	-	1	-	-	6

Structured Instances

coloring (clr)
set covering (stcv)
capacitated p-median (cpmp)
survivable fixed telecom
network design (sdlb)
cutting stock (ctst)
generalized assignment (gap)
network design (ntlb)
resource allocation (rap)
capacitated vehicle routing (cvrp)
lot sizing (ltsz)
bin packing (bp)
stable set (stbl)

Splitting Training and Testset

Reminder: datapoints $(\mathcal{P}, \mathcal{D}, t)$

Split training and test set by mip instances, to avoid a biased estimator.

Distribution of decompositions per MIP instance:

- Average: ~ 15.3
- Standard Deviation: ~ 9.0

	Instances	Decompositions
Training	269 ($\sim 2/3$)	4434
Test	131 ($\sim 1/3$)	2069

Overall Performance

- Test set of 131 MIP instances, 99 structured and 32 unstructured.
- `GCG` better than `SCIP` on 34 instances.

Nearest neighbor classifier of `scikit-learn` library.

Instances	All				Structured				Non-structured			
Solver	`SCIP`	`GCG`	SL	OPT	`SCIP`	`GCG`	SL	OPT	`SCIP`	`GCG`	SL	OPT
No opt. sol.	52	66	44	39	39	37	31	26	13	29	14	13
CPU time (h)	111.3	142.6	93.1	85.7	83.5	82.2	65.9	58.5	27.8	56.8	29.2	27.2
Geo. mean (s)	127.1	370.4	78.6	67.8	73.4	146.9	39.2	32.2	672.9	5145.0	766.0	646.5

- `SCIP`: apply `SCIP` to all instances
- `GCG`: apply `GCG` with build-in selection tool
- SL: our supervised learning scheme
- OPT: best decomposition selected each time

Avoid using GCG when there is no appropriate structure.

For $(\mathcal{P}, \mathcal{D}, t)$: Is GCG on $(\mathcal{P}, \mathcal{D})$ better than SCIP on $\mathcal{P}$?

		All instances		Structured		Non-structured	
Classifier	Pred.	SCIP 74.0%	GCG 26.0%	SCIP 68.7%	GCG 31.3%	SCIP 90.6%	GCG 9.4%
RBF Unbal.	SCIP	TN	FN				
	GCG	FP	TP				
KNN distance.	SCIP						
	GCG						
RF Unbal.	SCIP						
	GCG						
RF Bal.	SCIP						
	GCG						

Solver Selection Accuracy

Avoid using GCG when there is no appropriate structure.

For $(\mathcal{P}, \mathcal{D}, t)$: Is GCG on $(\mathcal{P}, \mathcal{D})$ better than SCIP on $\mathcal{P}$?

		All instances		Structured		Non-structured	
		SCIP	GCG	SCIP	GCG	SCIP	GCG
Classifier	Pred.	74.0%	26.0%	68.7%	31.3%	90.6%	9.4%
RBF	SCIP	73.3%	19.1%	66.7%	23.2%	90.6%	9.4%
Unbal.	GCG	3.8%	3.8%	5.1%	5.1%	0.0%	0.0%
KNN	SCIP	69.5%	9.9%	64.6%	11.1%	84.4%	6.3%
distance.	GCG	6.9%	13.7%	7.1%	17.2%	6.3%	3.1%
RF	SCIP	63.4%	11.5%	55.6%	13.1%	87.5%	6.3%
Unbal.	GCG	10.7%	14.5%	13.1%	18.2%	3.1%	3.1%
RF	SCIP	60.3%	10.7%	50.5%	11.1%	90.6%	9.4%
Bal.	GCG	13.7%	15.3%	18.2%	20.2%	0.0%	0.0%

Avoid using GCG when there is no appropriate structure.

For $(\mathcal{P}, \mathcal{D}, t)$: Is GCG on $(\mathcal{P}, \mathcal{D})$ better than SCIP on $\mathcal{P}$?

On many supervised learning problems,

- a simple approach gives a quite good performance
- a much more involved approach does only slightly better

Keep it simple!

Suppose we have a tractable MILP formulation for some given problem.

Make fast predictions on new instances using supervised learning

We can predict more on specific problems

Suppose we have a tractable MILP formulation for some given problem.

Make fast predictions on new instances using supervised learning

Prediction of the value:

- Fischetti, M., & Fraccaro, M. (2018). Machine learning meets mathematical optimization to predict the optimal production of offshore wind parks. Computers & Operations Research.

We can predict more on specific problems

Suppose we have a tractable MILP formulation for some given problem.

Make fast predictions on new instances using supervised learning

Prediction of the value:

- Fischetti, M., & Fraccaro, M. (2018). Machine learning meets mathematical optimization to predict the optimal production of offshore wind parks. Computers & Operations Research.

Predictions of (parts of) a good solution:

- Larsen, E., Lachapelle, S., Bengio, Y., Frejinger, E., Lacoste-Julien, S., & Lodi, A. (2018). Predicting solution summaries to integer linear programs under imperfect information with machine learning. arXiv preprint arXiv:1807.11876.
- Larsen, E., Lachapelle, S., Bengio, Y., Frejinger, E., Lacoste-Julien, S., & Lodi, A. (2019). Predicting Tactical Solutions to Operational Planning Problems under Imperfect Information. arXiv preprint arXiv:1901.07935.

In many industrial context, solve every day variants of the same problem

- We have an exact MILP solver
- Too slow for operational use

Can we combine OR and ML to have a better solver?

Bayesian optimization for advanced parameter selection?

Can use reinforcement learning to learn along a branching scheme ?

Machine learning uses data to:

3. take decisions (reinforcement learning)

"If I want B to happen, then I should do A"

Machine learning uses data to:

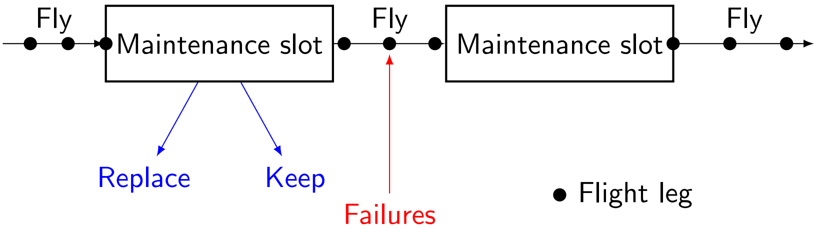
3. take decisions (reinforcement learning)

"If I want B to happen, then I should do A"

- Operations Research is decision science (originally without data)
- Data changes our problem

Taking decisions based on data requires to tackle with

- data (ML)
- curse of dimensionality (OR)



Data available: airplane signals recorded at 1 Hz. Previous failures.

Objective : Find an optimal maintenance planning minimizing the expected costs

A distribution $(X_v)_{v \in V}$ factorizes as a directed graphical model on a digraph $D = (V, A)$ if

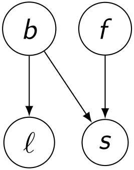
$$\mathbb{P}(X_V = x_V) = \prod_{v \in V} p_{v|\mathrm{prt}(v)}(x_v, x_{\mathrm{prt}(v)})$$

where D is acyclic and

$$\mathbb{P}(X_v = x_v | X_{\mathrm{prt}(v)} = x_{\mathrm{prt}(v)}) = p_{v|\mathrm{prt}(v)}(x_v | x_{\mathrm{prt}(v)})$$

Example:

- battery b
- fuel f
- lights ℓ
- engine start s



Probabilistic graphical models – definition and inference problem

A distribution $(X_v)_{v \in V}$ factorizes as a directed graphical model on a digraph $D = (V, A)$ if

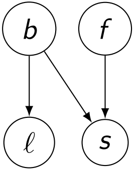
$$\mathbb{P}(X_V = x_V) = \prod_{v \in V} p_{v|\mathrm{prt}(v)}(x_v, x_{\mathrm{prt}(v)})$$

where D is acyclic and

$$\mathbb{P}(X_v = x_v | X_{\mathrm{prt}(v)} = x_{\mathrm{prt}(v)}) = p_{v|\mathrm{prt}(v)}(x_v | x_{\mathrm{prt}(v)})$$

Example:

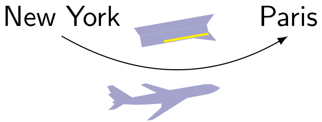
- battery b
- fuel f
- lights ℓ
- engine start s



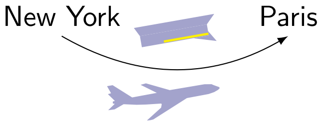
Inference problem: compute $\mathbb{E}(f(X_v))$ or $\mathbb{P}(X_v = x_v | X_E = x_E)$.

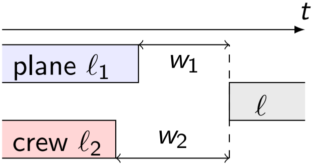
Example: probability that the fuel tank is empty given that lights work and engine does not start?

Richer example of Inference problem

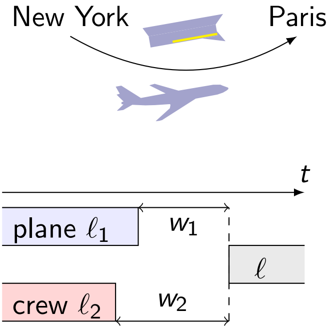


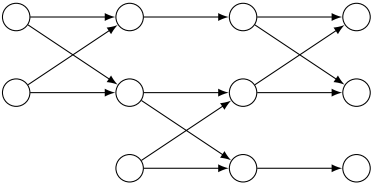
Richer example of Inference problem





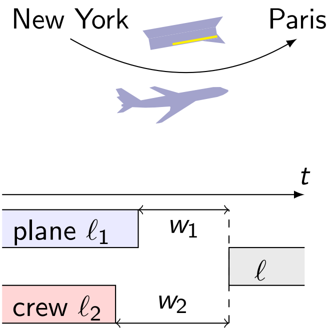
Richer example of Inference problem

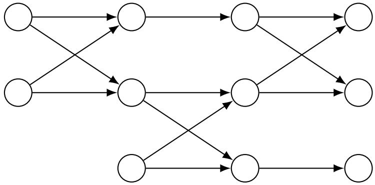




$$\prod_{v \in V} p_{v|\mathrm{prt}(v)}$$

Richer example of Inference problem





$$\prod_{v \in V} p_{v|\mathrm{prt}(v)}$$

We cannot hope to work directly with $\mu_V(x_V) = \mathbb{P}(X_V = x_V)$

As V is large, work with collection of moments μ_C with $|C|$ small

$$\mathcal{M} = \left\{(\mu_C)_{C\in\mathcal{V}} \colon \ : \exists \mu_V, \forall C, \forall x_C, \mu_C(x_C) = \sum_{x_{V\setminus C}} \mu_V(x_C, x_{V\setminus C})\right\}$$

As V is large, work with collection of moments μ_C with $|C|$ small

$$\mathcal{M} = \left\{ (\mu_C)_{C \in \mathcal{V}} : \ : \exists \mu_V, \forall C, \forall x_C, \mu_C(x_C) = \sum_{x_{V \backslash C}} \mu_V(x_C, x_{V \backslash C}) \right\}$$

Theorem (e.g. Theorem 3.4 in Wainwright and Jordan (2008))

Denoting $\theta = (\log(p_{v|\mathrm{prt}(v)}))$, an optimization solution μ^* of the convex optimization problem

$$\max_{\mu \in \mathcal{M}} \langle \theta | \mu \rangle + H(\mu)$$

is such that $\mathbb{P}(X_C = x_c) = \mu_C(x_C)$, where H is the entropy function.

As V is large, work with collection of moments μ_C with $|C|$ small

$$\mathcal{M} = \left\{ (\mu_C)_{C \in \mathcal{V}} : \; : \exists \mu_V, \forall C, \forall x_C, \mu_C(x_C) = \sum_{x_{V \backslash C}} \mu_V(x_C, x_{V \backslash C}) \right\}$$

Theorem (e.g. Theorem 3.4 in Wainwright and Jordan (2008))

Denoting $\theta = (\log(p_{v|\mathrm{prt}(v)}))$, an optimization solution μ^* of the convex optimization problem

$$\max_{\mu \in \mathcal{M}} \langle \theta | \mu \rangle + H(\mu)$$

is such that $\mathbb{P}(X_C = x_c) = \mu_C(x_C)$, where H is the entropy function.

- Fenchel duality
- Efficient algorithms
- Techniques to build approximations ($\mathcal{M}$ and H)

Learning parameters: knowing the digraph $D = (V, A)$, learning $p_{v|\mathrm{prt}(v)}$:

- $p_{v|\mathrm{prt}(v)}$ is generally small dimensional
- does not require much data
- covers OR applications presented

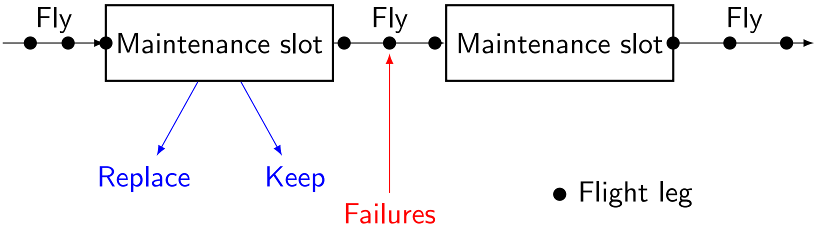
Learning the structure: much harder

Three main kind of problems:

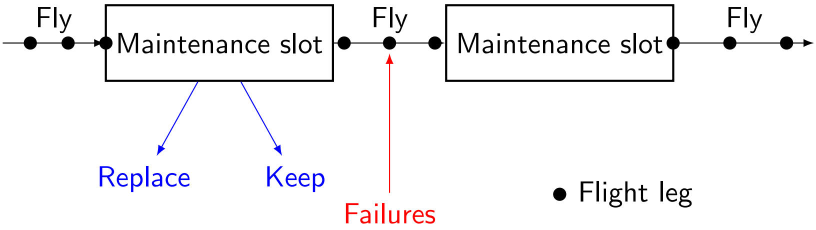
- *Inference*: Compute $\mathbb{E}(f(X_v))$
- *Learning*: learn the statistical model from data
- *Decision*: stochastic optimization

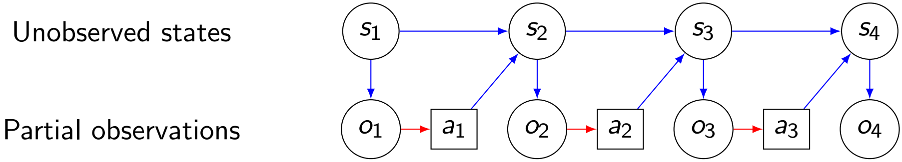
References:

- Wainwright, M. J., & Jordan, M. I. (2008). Graphical models, exponential families, and variational inference. Foundations and Trends in Machine Learning, 1(1–2), 1-305.
- Koller, D., & Friedman, N. (2009). Probabilistic Graphical Models: Principles and Techniques (Adaptive Computation and Machine Learning series). MIT Press, Aug, 31, 2009.

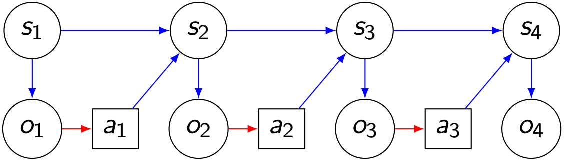

Fly
Maintenance slot
Fly
Maintenance slot
Fly
Replace
Keep
Failures
Flight leg

Back to predictive maintenance example



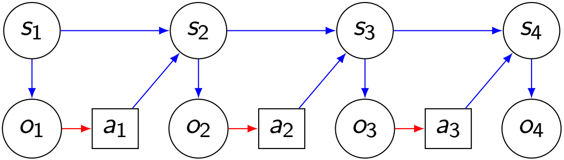


Predictive maintenance example



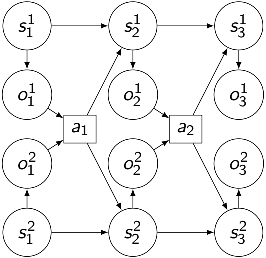
$$\mathbb{P}_\delta(X_V = x_V) = \prod_{v \in V^{\mathrm{s}}} p(x_v | x_{\mathrm{prt}(v)}) \prod_{v \in V^{\mathrm{a}}} \delta_{v|\mathrm{prt}(v)}(x_v | x_{\mathrm{prt}(v)}).$$

Predictive maintenance example



$$\mathbb{P}_\delta(X_V = x_V) = \prod_{v \in V^{\mathrm{s}}} p(x_v | x_{\mathrm{prt}(v)}) \prod_{v \in V^{\mathrm{a}}} \delta_{v|\mathrm{prt}(v)}(x_v | x_{\mathrm{prt}(v)}).$$

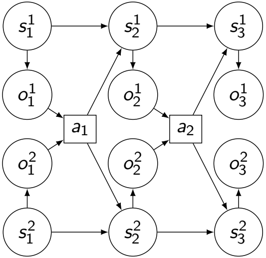
$$\max_{\delta \in \Delta} \quad \mathbb{E}_\delta\left(\sum_{v \in V^\ell} r_v(X_v) \right).$$



MILP approach to influence diagrams:

- Valid inequalities leveraging independence structure
- Linear program on soluble influence diagrams

Influence diagrams



MILP approach to influence diagrams:

- Valid inequalities leveraging independence structure
- Linear program on soluble influence diagrams

- 16:50, Salle C.103, Victor Cohen, Linear Programming for Decision Processes with Partial Information
- Cohen, V., & Parmentier, A. (2018). Linear Programming for Decision Processes with Partial Information. arXiv preprint arXiv:1811.08880.
- P., A., Cohen, V., Leclère, V., Obozinski, G., & Salmon, J. (2019). Mathematical programming for influence diagrams. arXiv preprint arXiv:1902.07039.

Conclusion

As an OR practitioner, know which kind of problem ML can solve

- Unsupervised learning
- Supervised learning
- Reinforcement learning

ML can help you take heuristic decisions that speed-up your OR algorithm

Probabilistic graphical models are an interesting tool in the context of data driven optimization

Thank you

Post-doc positions at Cermics on the topic