Optimize to learn to optimize: the Algorithm Configuration Problem

Claudia D'Ambrosio[1], Antonio Frangioni[2]
Gabriele Iommazzo[1,2], Leo Liberti[1]

[1]OptimiX Team, CNRS LIX, Ecole Polytechnique

[2]Operations Research Group, Dipartimento di Informatica, Universià di Pisa

Machine Learning NeEDS Mathematical Optimization

May 10, 2021

Outline

1. Motivation
2. The Algorithm Configuration Problem
3. Our ML+MO take on the ACP
4. Implementation: the ML model counts – a lot!
5. Implementation: the nitty-gritty details count – a lot!
6. A glimpse to computational results
7. Conclusions

Motivation: Mathematical Optimization is HARD

- Most Mathematical Optimization (MO) problems are hard in practice
- In theory "all $\mathcal{NP}$-hard problems equally difficult", but in practice hard means very different things:
  - difficult to find any feasible solution at all, and/or
  - difficult to prove that a given solution is optimal, and/or
  - difficult to do anything because you need a huge formulation
  - difficult to do anything because functions nonlinear/hard to compute

  $\Longrightarrow$ no single approach will always work, need specialised ones
- Any solver (and especially general-purpose ones) need many of them:
  - bounding techniques (relaxations, cuts, ...), branching, dominance, ...
  - heuristics (math-, fast, expensive, ...)
  - preprocessing, reformulations, ...

  $\Longrightarrow$ any efficient approach is many approaches in one

Motivation (cont.d)

- Obviously even worse for
  huge-scale problems with multiple nested forms of structure
  that require multi-level nested decomposition approaches

  `https://gitlab.com/smspp/smspp-project`

Motivation (cont.d)

- Obviously even worse for
  huge-scale problems with multiple nested forms of structure
  that require multi-level nested decomposition approaches
  `https://gitlab.com/smspp/smspp-project`
  . . . but that's another story
- Today it's general-purpose MO solvers and their 100s parameters
- Tuning them crucial for efficiency, but hard mostly manual process
  requiring skilled personnel with in-depth knowledge of the techniques
- This even assuming you really know what's going on, not always true
- Necessarily instance-specific because even "MILP" can be anything
- Clearly lots of scope to automatise experience-based process:
  if this does not say "ML", I do not know what does
- In fact, that's our (and a lot of other people's) take

Outline

1. Motivation
2. The Algorithm Configuration Problem
3. Our ML+MO take on the ACP
4. Implementation: the ML model counts – a lot!
5. Implementation: the nitty-gritty details count – a lot!
6. A glimpse to computational results
7. Conclusions

Algorithm Configuration Problem (ACP) [30]

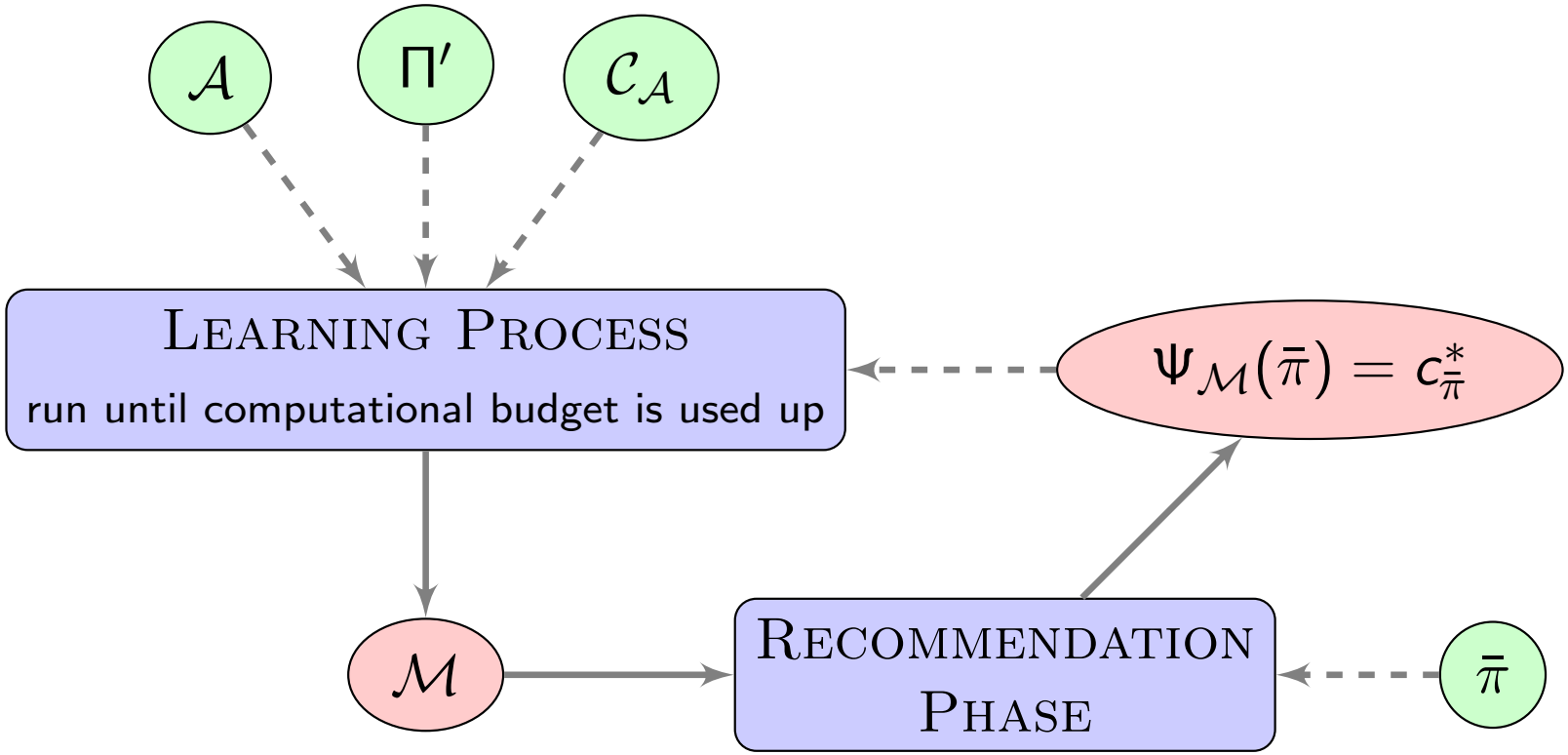
- Target algorithm $\mathcal{A}$ for problem Π = (infinite) set of instances
- Configuration c for $\mathcal{A}$ = finite (but not small) vector of parameters
- Set $\mathcal{C}_\mathcal{A}$ of feasible parameter configurations: typically bounded but nontrivial (continuous/categorical parameters, logic relationships)
- $p_\mathcal{A} : \Pi \times \mathcal{C}_\mathcal{A} \longrightarrow \mathbb{R}$ performance function measuring efficiency and effectiveness (nontrivial: find solution vs. solution quality vs. time)
- ACP: given $\bar{\pi} \in \Pi$ find $c^*_{\bar{\pi}} \in \mathcal{C}_\mathcal{A}$ providing optimal $p_\mathcal{A}(\bar{\pi}, c^*_{\bar{\pi}})$
  a meta optimization problem in our case
- "Simpler version": Algorithm Selection Problem (ASP)
- Can only be very hard (undecidable?)
- Wealth of other applications: hyperparameter tuning in ML, personalised medical treatment, ...

Algorithm Configuration Problem (ACP)

- Solution to ACP $=$ recommender $\Psi_{\mathcal{M}} : \Pi \rightarrow \mathcal{C}_{\mathcal{A}}$
  given $\pi \in \Pi$ returns optimal configuration $c^*_\pi \in \mathcal{C}_{\mathcal{A}}$
- $\Psi_{\mathcal{M}}$ need assess $p_{\mathcal{A}}$ over entire $\mathcal{C}_{\mathcal{A}}$, but
  - $\mathcal{C}_{\mathcal{A}}$ large, exhaustive evaluation of $p_{\mathcal{A}}$ on $\mathcal{C}_{\mathcal{A}}$ infeasible
  - $p_{\mathcal{A}}$ black-box: run $\mathcal{A}$ to evaluate $p_{\mathcal{A}}$ costly, defies the purpose
  - $\mathcal{A}$ may even be one-shot, non-repeatable process (medical treatment)

  $\Longrightarrow$ model $\mathcal{M}$: encodes knowledge about $p_{\mathcal{A}}$
- $\mathcal{M}$ worst case/average complexity theory: difficult and imprecise
- Data-driven approach: $\mathcal{M}$ is learnt out of experiments
- Requires learning phase, training set
- $\mathcal{M}$ itself can guide exploration of $\mathcal{C}_{\mathcal{A}}$ in learning phase

ACP: a general scheme



- $\Pi' \subset \Pi$ = training set of instances
- Initial subset of $\Pi' \times \mathcal{C}_{\mathcal{A}}$ typically by design of experiments
- Learning process often itself iterative, either $\mathcal{M}$ or different $\mathcal{M}'$ guides selection of new $(\bar{\pi}, \bar{c})$

ACP Taxonomy

- Fundamental elements:
  - model $\mathcal{M}$ and associated recommender $\Psi_{\mathcal{M}}$
  - whether $\Psi_{\mathcal{M}}(\pi)$ really depends on π (Per-Instance) or always gives same answer (Per-Problem)
  - whether $\mathcal{M}$ is fixed during $\mathcal{A}$ run (offline) or information from the run is dynamically included (online)
- Three main types of recommenders:
  - $\zeta_{\mathcal{A}} : \Pi \longrightarrow \mathcal{C}_{\mathcal{A}}$: directly recommend a configuration
    $\Longrightarrow \quad \Psi_{\mathcal{M}}(\bar{\pi}) = \zeta_{\mathcal{A}}(\bar{\pi})$
  - $\bar{p}_{\mathcal{A}} : \Pi \times \mathcal{C}_{\mathcal{A}} \longrightarrow \mathbb{R}$: compute approximation of $p_{\mathcal{A}}$
    $\Longrightarrow \quad \Psi_{\mathcal{M}}(\bar{\pi}) = \arg\max_{c \in \mathcal{C}_{\mathcal{A}}} \bar{p}_{\mathcal{A}}(\,\bar{\pi}\,,\,c\,)$
  - $\mathscr{C}_{\mathcal{A}} = \{\,(\,\pi\,,\,c^*_{\pi}\,),\, \pi \in \Pi^*\,\}$: configurations for representative instances
    $\Longrightarrow \quad \Psi_{\mathcal{M}}(\bar{\pi}) = c^*_{\pi^*}$ where $\pi^* \in \arg\min_{\pi \in \Pi^*} \{\,||\,\pi - \bar{\pi}\,||\,\}$
    (PP special case where $\mathscr{C}_{\mathcal{A}}$ singleton)
- $\bar{p}_{\mathcal{A}} + \mathscr{C}_{\mathcal{A}}$: compute $\bar{p}_{\mathcal{A}}(\,\bar{\pi}\,,\,c\,)$ only for $c \in \mathscr{C}_{\mathcal{A}}$, choose best

ACP/ASP: a bird's eye on the literature

	PI		PP
$\mathcal{M}$	offline	online	offline
$\zeta_{\mathcal{A}}$	[9]*, [7, 10], [24]		
$\bar{p}_{\mathcal{A}}$	[6, 8, 20, 31, 32], [23]		
$\mathscr{C}_{\mathcal{A}}$ per-instance		[5, 29]	
$\mathscr{C}_{\mathcal{A}}$ clusters	[12, 16]*, [25]		
$\bar{p}_{\mathcal{A}} + \mathscr{C}_{\mathcal{A}}$	[17, 31, 32]	[17]	
$\mathscr{C}_{\mathcal{A}}$ singleton			[1, 2, 3, 4, 14, 15] [21, 22, 27, 28]

- Our contributions in red (* = ASP only)
- Our approach per-instance offline, but online would be possible
- In this talk: only $\bar{p}_{\mathcal{A}}$

Outline

1. Motivation
2. The Algorithm Configuration Problem
3. Our ML+MO take on the ACP
4. Implementation: the ML model counts – a lot!
5. Implementation: the nitty-gritty details count – a lot!
6. A glimpse to computational results
7. Conclusions

Our approach to the ACP

- Assumptions:
  - (mostly harmless) $\mathcal{C}_{\mathcal{A}} = \{ c \in \{0,1\}^q \mid Ac \leq d \}$: only categorical parameters, MILP representation of logical constraints
  - (not entirely harmless) performance function = gap within time-limit, special scaling when no solution $= \infty$
  - (not at all harmless) encoding $e(\pi) \in \mathbb{R}^m$ for all $\pi \in \Pi =$ knowing which features of the instance influence the behaviour of $\mathcal{A}$
- Two-phases process:
  - Performance Map Learning Phase (PMLP): learn an approximation $\bar{p}_{\mathcal{A}}$ of $p_{\mathcal{A}}$ via "any" supervised ML technique on the training set

$$\mathcal{S} = \{ (\underbrace{e(\pi_i), c_i}_{x_i}, \underbrace{p_{\mathcal{A}}(\pi_i, c_i)}_{f(x_i)}) \} \subseteq \underbrace{\mathbb{R}^m \times \mathcal{C}_{\mathcal{A}}}_{\mathcal{X}} \times \underbrace{\mathbb{R}}_{\mathcal{Y}}$$

  - Configuration Space Search Problem (CSSP): solve

$$\text{CSSP}(\bar{\pi}) \qquad c^* \in \arg\max_{c \in \mathcal{C}_{\mathcal{A}}} \bar{p}_{\mathcal{A}}(\bar{\pi}, c)$$

    using the most appropriate MO technique

Our specific twists

- PP is not enough, it just isn't for general-purpose MO solvers
  $\Longrightarrow$ has to make nontrivial decisions when $\bar{\pi}$ arrives
- Cannot assume q small, it just isn't for general-purpose MO solvers
  $\Longrightarrow$ cannot treat $\bar{p}_{\mathcal{A}}$ as black-box
  (derivative-free optimization just does not scale) $\Longrightarrow$
  pry open the black-box and use powerful available MO tools
- Idea not entirely new [26] but successful applications limited, application to ACP context new
- Optimizing to learn to optimize: use MO techniques to improve on MO techniques via ML

Easier said than done

- Several significant implementation challenges
- Each ML paradigm gives rise to entirely different CSSP, requiring a separate implementation effort and specific tools
- Trade-offs in ML paradigm: more "complex" ones may yield higher generalization but result in more difficult CSSP
- Which performance function to be learned (feasibility vs. optimality)?
- Traditional ML loss metrics do not necessarily reflect the purpose: the "right" $\bar{p}_{\mathcal{A}}$ ideally has the same global minima as $p_{\mathcal{A}}$ even if the function values are very different
- Dimensionality reduction techniques (FS) modify/shrink $x = (e, c) \implies$ eliminate configuration variables $\implies$ c^* may not be immediate to reconstruct out of the CSSP solution

Outline

1. Motivation
2. The Algorithm Configuration Problem
3. Our ML+MO take on the ACP
4. Implementation: the ML model counts – a lot!
5. Implementation: the nitty-gritty details count – a lot!
6. A glimpse to computational results
7. Conclusions

PMLP: Support Vector Regression

- Several techniques (e.g., linear SVR) yield $\bar{f}(x) = wx + b$
- Cannot be used in a PI approach: $x = (\, e\,,\, c\,) \implies$
  $\bar{f}(\, e\,,\, c\,) = w_c c + w_e e + b \implies$
  the solution of CSSP is always the same for each π
- Must use nonlinear SVR: with kernel trick this is

$$\begin{aligned} \min & \sum_i \sum_j \alpha_i \kappa(x_i, x_j) \alpha_j + \varepsilon \sum_i |\alpha_i| - \sum_i y_i \alpha_i \\ & \sum_i \alpha_i = 0 \;\;,\;\; -C \le \alpha_i \le C \qquad \forall i \end{aligned}$$

  yielding $\bar{f}(x) = \sum_i \alpha_i^* \kappa(x_i, x) + b^*$
- Actual form of CSSP then depends on kernel; with Gaussian one

$$\text{CSSP}(\bar{\pi}) \quad \min_{c \in \mathcal{C}_{\mathcal{A}}} \; \sum_i \alpha_i^* \, e^{-\gamma^* \| (\, e_i\,,\, c_i\,) - (\, e(\bar{\pi})\,,\, c\,) \|_2^2} + b^*$$

  a linearly constrained nonconvex MINLP

PMLP: Neural Networks (NN)

- NN [19]: digraph $G = (I \cup H \cup \{o\}, A)$, arc weights w, node biases b, activation functions σ_n (linear, ReLU, sigmoid, ...)
- $a_n = x_n$ for $n \in I$, $a_n = \sigma_n(\sum_u w_{un} a_u + b_n)$ otherwise
- In our case, $I = E \cup C$ (features and controls)
- CSSP($\bar{\pi}$) in general a nonconvex MINLP

$$
\begin{aligned}
\max\ & \sigma_o(\textstyle\sum_u w^*_{uo} a_u + b^*_o) \\
& a_n = c_n && n \in C \\
& a_n = e(\bar{\pi})_n && n \in E \\
& a_n = \sigma_n(\textstyle\sum_u w^*_{un} a_u + b^*_n) && n \in H \\
& c \in \mathcal{C}_{\mathcal{A}}
\end{aligned}
$$

- With σ = linear or ReLU actually a MILP, although not necessarily "easy" (nasty big-M [18] + many extra variables)

PMLP: "poorman's NN" = Logistic Regression (LR)

- Conditional probability $\mathrm{P}(\mathcal{Y} = y \| \mathcal{X} = x)$ of Bernoulli variable $\mathcal{Y}$ [13]: $\bar{f}(x) = \frac{1}{1+e^{-z}}$ if $\exists w \in \mathbb{R}^d, b \in \mathbb{R}$ s.t. $z = wx + b$
- Convex training program to find best w^*, b^*
- CSSP($\bar{\pi}$) a linearly constrained MINLP with concave objective

$$\max_{c \in \mathcal{C}_{\mathcal{A}}} \left\{ \ln \left(\sigma(c) = \frac{1}{1 + e^{-(w^*(\, e(\bar{\pi})\, ,\, c\,) + b^*\,)}} \right) \right\}$$

- A very simple NN with $H = \emptyset$ (and it probably shows)
- However, interesting different probabilistic-motivated CSSP forms

$$\max_{c \in \mathcal{C}_{\mathcal{A}}\, ,\, r \in [0,1]} \left\{ r \ln(\sigma(c)) + (1 - r)(1 - \ln(\sigma(c))) + r \; ; \; \ldots \right\}$$

($r \approx$ estimate of the conditional probability)

- May work better [24] but nastier to solve

PMLP: Decision Tree (DT)

- Decision Tree (DT): binary tree $T = (V_I \cup V_L, A)$, labels λ_n on leaves $n \in V_L$, threshold/index function $\tau/\rho : V_I \to I$ otherwise
- In our case, $V_I = V_E \cup V_C$ (features and controls) depending on $\rho(n)$
- Training = construct T^*, λ^*, τ^* and ρ^*
- CSSP($\bar{\pi}$): $T^* \to T' = (V_C^* \cup V_L^*, A')$ (may be much smaller, entire sub-trees not reachable) as $\tau^*(n)$ for $n \in V_E$ can be verified statically
- CSSP($\bar{\pi}$) a MILP: path variables $y_{p[n],n} \in \{0, 1\}$ ($y_{p[r],r} = 1$)

$$
\begin{aligned}
\max \ & \textstyle\sum_{n \in V_L^*} y_{p[n],n} \lambda_n^* & \\
& y_{p[n],n} = y_{n,n_-} + y_{n,n_+} & n \in V_C^* \\
& y_{p[n],n} - c_{\rho^*(n)} \leq y_{n,n_-} & n \in V_C^* \\
& y_{p[n],n} + c_{\rho^*(n)} - 1 \leq y_{n,n_+} & n \in V_C^* \\
& c \in \mathcal{C}_{\mathcal{A}} &
\end{aligned}
$$

- No nasty big-M constraints, but $c \in \{0, 1\}$ not general

Outline

1. Motivation
2. The Algorithm Configuration Problem
3. Our ML+MO take on the ACP
4. Implementation: the ML model counts – a lot!
5. Implementation: the nitty-gritty details count – a lot!
6. A glimpse to computational results
7. Conclusions

Challenges: choice of $p_{\mathcal{A}}$

- $p_{\mathcal{A}} \approx$ gap for fixed time limit (60s), but details count
- Solver ran 3 times with $\neq$ seeds to weed out impact of random choices, $\varrho(\pi, c) =$ middle value
- $\varrho(\pi, c)$ can be large ($\infty =$ no solution), compute max $\bar{\varrho}$ for $\varrho \leq$ 1e+5
- re-set all $\varrho >$ 1e+5 to max $\bar{\varrho} + 100$
- re-scale ϱ in $[0, 1]$
- Not terribly general (but proper scaling important in ML)
- Extending to non-fixed time limit?

Challenges: PMLP loss metrics

- Prediction error $\rho_i = p_i - \bar{p}_i = p_{\mathcal{A}}(\pi_i, c_i) - \bar{p}_{\mathcal{A}}(\pi_i, c_i)$
- Standard ML loss $= \sum_i \ell(p_i, \bar{p}_i)$, some choices for ℓ:
  - MAE $= |\rho_i|$: robust to outliers, sparse solutions, nonsmooth
  - MSE $= (\rho_i)^2$: different for $\gg 1$ and $\ll 1$, sensitive to outliers, smooth
  - others with intermediate traits (log cosh loss, elastic net, ...)

  None of them really measures correspondence of global minima
- CMAE$_\delta$ for $\delta \in [0, 1]$ (requires p_i scaled in $[0, 1]$)

$$\ell_\delta(p_i, \bar{p}_i) = \begin{cases} \rho_i(1 + 1/(1 + \exp(\rho_i))) & \text{if } p_i \leq \delta \text{ and } \rho_i < 0 \\ \rho_i(1 + 1/(1 + \exp(-\rho_i))) & \text{if } p_i \geq 1 - \delta \text{ and } \rho_i > 0 \\ \rho_i & \text{if } \delta \leq p_i \leq 1 - \delta \\ 0 & \text{otherwise} \end{cases}$$

  Penalize overestimates but allow underestimates "on δ-minima" ($p_i \leq \delta$), vice-versa "on δ-maxima" ($p_i \geq 1 - \delta$), MAE elsewhere
- Rather crude, but at least tries

Challenges: CSSP solution completion

- Ideally, CSSP defined on $\mathcal{C}_{\mathcal{A}} = \{ c \in \{0, 1\}^q \mid Ac \leq d \}$
- But after FS, $\mathcal{C}_{\mathcal{A}} = \{ c' \in \{0, 1\}^{q'} \mid A'c' \leq d' \}$, $q' < q$
- Work on single parameter ω hit-encoded (q_ω bits summing to ≤ 1)
- $q_\omega - q'_\omega < 2 \implies$ leftout variables values determined univocally
- $q_\omega - q'_\omega \geq 2 \implies$ leftout variables values determined heuristically
- $\mathcal{S}$ encoded by matrix $M_{\mathcal{S}}$; after FS, $M'_{\mathcal{S}'}$ (less columns and rows)
- Heuristic 1: given c^*, separately for each ω
  1. group rows of $M'_{\mathcal{S}'}$ by all q_ω variables
  2. for each group apply `mean` or `min` to $p_{\mathcal{A}} \longrightarrow$ new column $p'_{\mathcal{A}}$
  3. delete rows in grouped matrix where selected variable values are $\neq c^*$
  4. leftout variable values: choose grouped matrix row with minimum p'
- Heuristic 2: group rows of $M'_{\mathcal{S}'}$ by q' variables, for each group apply `min` and retrieve row with min $p_{\mathcal{A}}$, use leftout variable values of that row to complete c^*

Outline

1. Motivation
2. The Algorithm Configuration Problem
3. Our ML+MO take on the ACP
4. Implementation: the ML model counts – a lot!
5. Implementation: the nitty-gritty details count – a lot!
6. A glimpse to computational results
7. Conclusions

Experimental setup: SVR

- $\mathcal{A}$: IBM ILOG CPLEX (100+ algorithmic parameters)
- $\mathcal{C}_{\mathcal{A}}$: 2304 combinations of 9 hit-encoded CPLEX params ($q = 23$)
- All combinations tested on all instances (conceptual experiment)
- FS: "kind" ($q = 14$) or "aggressive" ($q = 10$)
- Π: Hydro UC instances [11], hand-picked $e(\pi)$ of 43 components
- Π': 187 in-sample instances + 63 out-of-sample ones
- training set size $\approx$ 11000 rows; losses MAE and CMAE$_\delta$ (hyperparameter selection)
- CSSP solver: Bonmin (heuristic for nonconvex problems)

Experiment metrics

- Global optimum c^{**} of CSSP, Bonmin solution c^{*} of CSSP
- Comparison with default CPLEX configuration $c^{\mathtt{cpx}}$
- "% glob": $c^{**} = c^{*}$ frequency, "gap": relative error when $c^{**} \neq c^{*}$
- For all configs, p = true gap and p^{ml} = rescaled gap
- Separately, primal / dual gap ϱ_{prim} / ϱ_{dual}, % of feasible solutions
- "%w" = wins, "%w+d" = wins + draws, "%w-d" = wins over non-draws, "gap d" = relative gap of c^{*} (or $c^{\mathtt{cpx}}$, same) w.r.t. c^{**}
- When something unspecified, average on all possible choices

Solving the CSSP, FS impact

	FS	% glob	gap	time
IS	no	83.69	2.01e-2	15.36
	kind	87.70	2.76e-2	10.73
	agg	84.49	3.12e-2	5.95
OS	no	85.32	1.59e-2	13.44
	kind	90.48	2.01e-2	12.23
	agg	93.25	1.96e-2	6.15

- FS improves "% glob" and reduces time (smaller problems)
- local optima c^* never too worse than c^{**} (at most 3.2%)

Solving the CSSP, metric impact

	metric	% glob	gap	time
IS	$CMAE_{.2}$	84.85	3.00e-2	11.41
	$CMAE_{.4}$	83.07	2.80e-2	10.34
	MAE	87.34	2.26e-2	10.91
OS	$CMAE_{.2}$	93.12	2.32e-2	10.68
	$CMAE_{.4}$	88.89	1.75e-2	10.08
	MAE	88.36	1.20e-2	10.75

- No impact on time
- Minor difference on solution quality, no clear trend

Recommender performance, FS impact

	p	FS	%w	%w+d	%w-d	gap d
IS	p	no	47.06	96.12	92.39	0
		kind	48.91	98.02	96.12	0
		agg	48.98	97.79	95.70	0
	p^{ml}	no	78.74	92.11	90.89	2.79e-1
		kind	81.60	93.94	93.08	1.52e-1
		agg	83.16	95.19	94.53	1.23e-1
OS	p	no	33.73	83.73	67.50	0
		kind	36.64	82.61	68.06	0
		agg	33.86	77.05	59.78	0
	p^{ml}	no	59.13	76.59	71.77	3.39e-1
		kind	60.71	77.05	72.54	2.83e-1
		agg	55.82	71.89	66.55	3.28e-1

- FS most often improves
- Draws are invariably on "easy" instances solved to optimality
- Results much better in "ML metric": you get what you learn

Recommender performance, metric impact

	p	metric	%w	%w+d	%w-d	avg d
IS	p	$CMAE_{.2}$	47.98	97.09	94.30	0
		$CMAE_{.4}$	48.46	97.59	95.27	0
		MAE	47.86	96.73	93.62	0
	p^{ml}	$CMAE_{.2}$	80.99	93.35	92.41	1.97e-1
		$CMAE_{.4}$	81.61	93.79	92.93	1.51e-1
		MAE	80.54	93.94	92.99	2.20e-1
OS	p	$CMAE_{.2}$	34.74	79.72	63.34	0
		$CMAE_{.4}$	35.63	81.39	65.99	0
		MAE	36.16	85.36	71.78	0
	p^{ml}	$CMAE_{.2}$	59.08	76.01	71.16	3.04e-1
		$CMAE_{.4}$	58.29	72.57	68.00	2.36e-1
		MAE	62.26	79.45	75.16	3.68e-1

- Metric does move something (only used in hyperparameter selection)
- Not much and not always in the right direction

Recommender performance, heuristic impact

set	p	heuristic	%w	%w+d	%w-d	gap d
IS	p	none	47.06	96.12	92.40	0
		clustF_h1_gbMean	49.67	98.80	97.65	0
		clustF_h1_gbMin	48.86	97.86	95.82	0
		clustF_h2	48.20	97.19	94.51	0
		noClust_h1_gbMean	48.86	97.86	95.82	0
		noClust_h1_gbMin	48.86	97.86	95.82	0
		noClust_h2	48.06	97.19	94.50	0
	p^{ml}	none	78.48	92.25	91.00	2.88e-1
		clustF_h1_gbMean	83.42	94.52	93.83	5.67e-2
		clustF_h1_gbMin	81.95	93.85	93.02	1.43e-1
		clustF_h2	82.29	95.25	94.55	1.62e-1
		noClust_h1_gbMean	81.95	93.85	93.02	1.43e-1
		noClust_h1_gbMin	81.95	93.85	93.02	1.43e-1
		noClust_h2	82.55	95.19	94.49	1.88e-1
OS	p	none	35.32	85.71	71.30	0
		clustF_h1_gbMean	35.91	80.95	65.72	0
		clustF_h1_gbMin	34.92	81.15	65.20	0
		clustF_h2	32.94	75.99	58.21	0
		noClust_h1_gbMean	34.92	81.15	65.20	0
		noClust_h1_gbMin	34.92	81.15	65.20	0
		noClust_h2	34.52	78.17	61.55	0
	p^{ml}	none	61.90	78.17	74.05	3.01e-1
		clustF_h1_gbMean	59.72	74.60	70.15	2.97e-1
		clustF_h1_gbMin	58.93	76.19	71.19	2.97e-1
		clustF_h2	53.17	70.04	63.95	3.86e-1
		noClust_h1_gbMean	58.93	76.19	71.19	2.97e-1
		noClust_h1_gbMin	58.93	76.19	71.19	2.97e-1
		noClust_h2	56.75	73.02	67.81	3.49e-1

- It has some effects (not really clear)

Recommender performance in "natural" metrics

set	FS	metric	%feas c^*_{cssp}	%feas c_{cpx}	$\varrho^{\text{cssp}}_{\text{prim}}$	$\varrho^{\text{cpx}}_{\text{prim}}$	$\varrho^{\text{cssp}}_{\text{dual}}$	$\varrho^{\text{cpx}}_{\text{dual}}$
IS	no	$\text{CMAE}_{.2}$	96.79	100	1.50e-1	5.81e-1	9.71e-2	1.02e-1
		$\text{CMAE}_{.4}$	98.40		1.14e-1		7.25e-2	
		MAE	98.93		1.06e-1		4.70e-2	
	kind	$\text{CMAE}_{.2}$	98.75		6.96e-2		3.79e-2	
		$\text{CMAE}_{.4}$	98.48		7.09e-2		5.02e-2	
		MAE	96.43		7.18e-2		4.16e-2	
	agg	$\text{CMAE}_{.2}$	98.04		7.85e-2		5.18e-2	
		$\text{CMAE}_{.4}$	97.59		9.27e-2		6.68e-2	
		MAE	97.59		7.745e-2		5.00e-2	
OS	no	$\text{CMAE}_{.2}$	87.30	100	7.00e-2	4.63e-1	7.93e-2	8.12e-2
		$\text{CMAE}_{.4}$	88.89		1.31e-1		9.69e-2	
		MAE	93.65		1.37e-1		6.61e-2	
	kind	$\text{CMAE}_{.2}$	89.68		1.47e-1		6.47e-2	
		$\text{CMAE}_{.4}$	91.01		1.54e-1		1.28e-1	
		MAE	92.86		1.49e-1		1.08e-1	
	agg	$\text{CMAE}_{.2}$	89.42		2.05e-1		9.06e-2	
		$\text{CMAE}_{.4}$	88.10		2.08e-1		9.87e-2	
		MAE	88.62		1.79e-1		7.73e-2	

- CPLEX default emphasises feasibility, we got what we learnt
- We always do better in all other metrics

Recommender performance in "natural" metrics (cont'd)

	FS	metric	%feas c^*_{bm}	%feas c_{cpx}	ϱ^{bm}_{prim}	ϱ^{cpx}_{prim}	ϱ^{bm}_{dual}	ϱ^{cpx}_{dual}
IS	no	none	97.46	100	1.23e-1	5.81e-1	7.44e-2	1.02e-1
	kind	clustF_h1_gbMean	98.53		7.08e-2		4.27e-2	
		clustF_h1_gbMin	98.66		7.34e-2		4.22e-2	
		clustF_h2	97.59		7.26e-2		4.49e-2	
		noClust_h1_gbMean	98.66		7.34e-2		4.22e-2	
		noClust_h1_gbMin	98.66		7.34e-2		4.22e-2	
		noClust_h2	97.46		7.91e-2		4.40e-2	
	agg	clustF_h1_gbMean	98.93		8.35e-2		5.71e-2	
		clustF_h1_gbMin	97.59		7.46e-2		4.71e-2	
		clustF_h2	97.59		9.16e-2		6.87e-2	
		noClust_h1_gbMean	97.59		7.45e-2		4.71e-2	
		noClust_h1_gbMin	97.59		7.45e-2		4.71e-2	
		noClust_h2	97.33		8.93e-2		6.23e-2	
OS	no	none	90.48	100	1.15e-1	4.63e-1	8.18e-2	8.12e-2
	kind	clustF_h1_gbMean	91.67		1.54e-1		1.05e-1	
		clustF_h1_gbMin	90.87		1.41e-1		1.01e-1	
		clustF_h2	88.89		1.43e-1		1.16e-1	
		noClust_h1_gbMean	90.87		1.41e-1		1.01e-1	
		noClust_h1_gbMin	90.87		1.41e-1		1.01e-1	
		noClust_h2	88.89		1.45e-1		1.08e-1	
	agg	clustF_h1_gbMean	90.87		2.03e-1		8.81e-2	
		clustF_h1_gbMin	88.89		1.85e-1		7.97e-2	
		clustF_h2	86.51		2.23e-1		1.28e-1	
		noClust_h1_gbMean	88.89		1.85e-1		7.97e-2	
		noClust_h1_gbMin	88.89		1.85e-1		7.97e-2	
		noClust_h2	88.49		2.16e-1		8.60e-2	

Outline

1. Motivation
2. The Algorithm Configuration Problem
3. Our ML+MO take on the ACP
4. Implementation: the ML model counts – a lot!
5. Implementation: the nitty-gritty details count – a lot!
6. A glimpse to computational results
7. Conclusions

Conclusions and (a lot of) future work

- General concept and initial results promising
- Many other things to test, among which
  - results with more ML paradigms (Gabriele's talk)
  - different solvers (Octeract?) & classes of problems (MINLP)
  - different/wider sets of instances (MI[N]LPLib, . . .)
  - different performance metrics
- Most interesting ideas (to me):
  - different performance metrics during learning
  - specialised approaches for different CSSPs
  - extension to multilevel nested decomposition (SMS++)
- Most difficult challenges to overcome:
  - automatically finding the right set of features (SMS++?)
  - what if Feature Transformation (SVD) rather than FS?
  - designing a system/community to share the enormous training cost
- Lots of fun to be had


B. Adenso-Díaz and M. Laguna.
Fine-tuning of algorithms using Fractional Experimental Design and Local Search.
Operations Research, 54(1):99–114, 2006.


C. Ansótegui, M. Sellmann, and K. Tierney.
A gender-based genetic algorithm for the automatic configuration of algorithms.
In *Proceedings of the 15th International Conference on Principles and Practice of Constraint Programming*, CP'09, pages 142–157, Berlin, Heidelberg, 2009. Springer-Verlag.


C. Audet, D. Kien, and D. Orban.
Algorithmic parameter optimization of the dfo method with the opal framework.
Software Automatic Tuning: From Concepts to State-of-the-Art Results, pages 255–274, 2010.


C. Audet and D. Orban.

Finding optimal algorithmic parameters using Derivative-Free Optimization.
SIAM Journal on Optimization, 17(3):642–664, 2006.


R. Battiti and M. Brunato.
Reactive Search: machine learning for memory-based heuristics.
Technical report, University of Trento, 2005.


N. Belkhir, J. Dréo, P. Savéant, and M. Schoenauer.
Per instance algorithm configuration of cma-es with limited budget.
In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '17, pages 681–688, New York, NY, USA, 2017. ACM.


T. Berthold and G. Hendel.
Learning to scale mixed-integer programs.
2020.


M. G. V. Boas, H. G. Santos, R. de S. O. Martins, and L. H. C. Merschmann.

Data mining approach for feature based parameter tuning for mixed-integer programming solvers.
In P. Koumoutsakos, M. Lees, V. V. Krzhizhanovskaya, J. J. Dongarra, and P. M. A. Sloot, editors, *International Conference on Computational Science, ICCS 2017, 12-14 Jun 2017, Zurich, Switzerland*, volume 108 of *Procedia Computer Science*, pages 715–724. Elsevier, 2017.

M. G. Vilas Boas, H. Gambini Santos, L. H. de Campos Merschmann, and G. Vanden Berghe.
Optimal decision trees for the algorithm selection problem: Integer programming based approaches.
CoRR, abs/1907.02211, 2019.

P. Bonami, A. Lodi, and G. G. Zarpellon.
Learning a classification of mixed-integer quadratic programming problems.
In W. J. van Hoeve (eds), editor, *Integration of Constraint Programming, Artificial Intelligence, and Operations Research.*

CPAIOR 2018, volume 10848 of *Lecture Notes in Control and Information Sciences*, pages 595–604. Springer, Cham, 2018.


A. Borghetti, C. D'Ambrosio, A. Lodi, and S. Martello.
An MILP approach for short-term hydro scheduling and unit commitment with head-dependent reservoir.
IEEE Transactions on Power Systems, 23(3):1115–1124, 2008.


A. Collins, L. Tierney, and J. Beel.
Per-instance algorithm selection for recommender systems via instance clustering.
CoRR, abs/2012.15151, 2020.


D. Cox.
The regression analysis of binary sequences.
Journal of the Royal Statistical Society, Series B, 20(2):215–242, 1958.


L. Pérez Cáceres and T. Stützle.
Exploring variable neighborhood search for automatic algorithm configuration.
Electronic Notes in Discrete Mathematics, 58:167–174, 2017.

C. Ansótegui *et al.*
Model-based genetic algorithms for algorithm configuration.
In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 733—739. AAAI Press, 2015.

J. Pérez *et al.*
A statistical approach for algorithm selection.
In C. C. Ribeiro, editor, *Experimental and Efficient Algorithms*, pages 417–431, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

M. Feurer *et al.*
Efficient and robust automated machine learning.
In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, volume 2 of *NIPS'15*, pages 2755—2763, Cambridge, MA, USA, 2015. MIT Press.

M. Fischetti and J. Jo.
Deep neural networks and mixed integer linear optimization.
Constraints, 23:296–309, 2018.

I. Goodfellow, Y. Bengio, and A. Courville.

Deep Learning.
MIT Press, 2016.


F. Hutter and Y. Hamadi.
Parameter adjustment based on performance prediction: Towards an instance-aware problem solver.
Technical report, In: Technical Report: MSR-TR-2005125, Microsoft Research, 2005.


F. Hutter, H. H. Hoos, and K. Leyton-Brown.
Sequential Model-based Optimization for general algorithm configuration.
In *Proceedings of the 5th International Conference on Learning and Intelligent Optimization*, LION'05, pages 507–523. Springer-Verlag, 2011.


F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stützle.
ParamILS: An automatic algorithm configuration framework.
Journal of Artificial Intelligence Research, 36(1):267–306, 2009.


G. Iommazzo, C. D'Ambrosio, A. Frangioni, and L. Liberti.

A learning-based mathematical programming formulation for the automatic configuration of optimization solvers.
In G. Nicosia *et. al*, editor, *Machine Learning, Optimization, and Data Science*, volume 12565 of *Information Systems and Applications, incl. Internet/Web, and HCI*, pages 700–712. Springer, 2020.


G. Iommazzo, C. D'Ambrosio, A. Frangioni, and L. Liberti.
Learning to configure mathematical programming solvers by mathematical programming.
In I. S. Kotsireas and P. M. Pardalos, editors, *Learning and Intelligent Optimization - 14th International Conference, LION 14, Athens, Greece, May 24-28, 2020, Revised Selected Papers*, volume 12096 of *Lecture Notes in Computer Science*, pages 377–389. Springer, 2020.


S. Kadioglu, Y. Malitsky, M. Sellmann, and K. Tierney.
ISAC: Instance Specific Algorithm Configuration.
In *Proceedings of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence*, pages 751–756, Amsterdam, The Netherlands, 2010. IOS Press.


M. Lombardi, M. Milano, and A. Bartolini.

Empirical decision model learning.
Artificial Intelligence, 244:343–367, 2017.


V. Nannen and A. E. Eiben.
Relevance estimation and value calibration of evolutionary algorithm parameters.
In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 975–980. Morgan Kaufmann Publishers Inc., 2007.


M. López-Ibá nez *et al.*
The irace package: iterated racing for automatic algorithm configuration.
Operations Research Perspectives, 3:43–58, 2016.


R. Pavón, F. Díaz R. and Laza, and V. Luzón.
Automatic parameter tuning with a Bayesian case-based reasoning system. a case of study.
Expert Syst. Appl., 36(2):3407–3420, 2009.


J. R. Rice.

The algorithm selection problem.
Advances in Computers, 15:65–118, 1976.


L. Xu, H. H. Hoos, and K. Leyton-Brown.
Hydra: Automatically configuring algorithms for portfolio-based selection.
In *Proceedings of the National Conference on Artificial Intelligence*, volume 1, 2010.


L. Xu, F. Hutter, H. H. Hoos, and K. Leyton-Brown.
Hydra-MIP: Automated algorithm configuration and selection for mixed integer programming.
In *Proceedings of the RCRA workshop on Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion at the International Joint Conference on Artificial Intelligence (IJCAI)*, 2012.