Getting the most out of your planner(s): from static to dynamic algorithm configuration

Frank Hutter

University of Freiburg & Bosch Center for AI
fh@cs.uni-freiburg.de

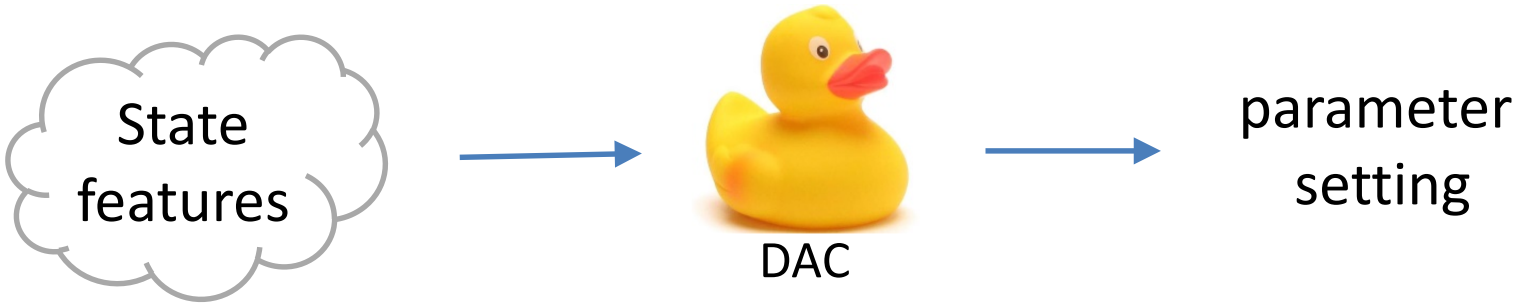
@FrankRHutter
@AutoMLFreiburg

These slides are available at www.automl.org/talks -- all references are hyperlinks

Motivation

- Algorithm configuration (AC) finds good settings of your parameters
  - But it is limited: the parameter setting is fixed

- We propose dynamic algorithm configuration (DAC)
  - This can change parameters based on the instance at hand, search progress, time, etc.



Outline

- Part 1: an overview of previous meta-algorithmic approaches
  - Algorithm Configuration
  - Algorithm Portfolios


Holger
Hoos


Kevin
Leyton-Brown

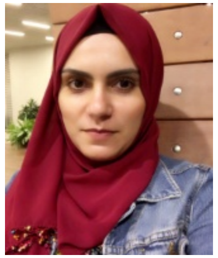
- Part 2: Dynamic Algorithm Configuration

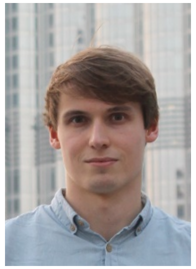

André
Biedenkapp


Marius
Lindauer


David
Speck


Robert
Mattmüller

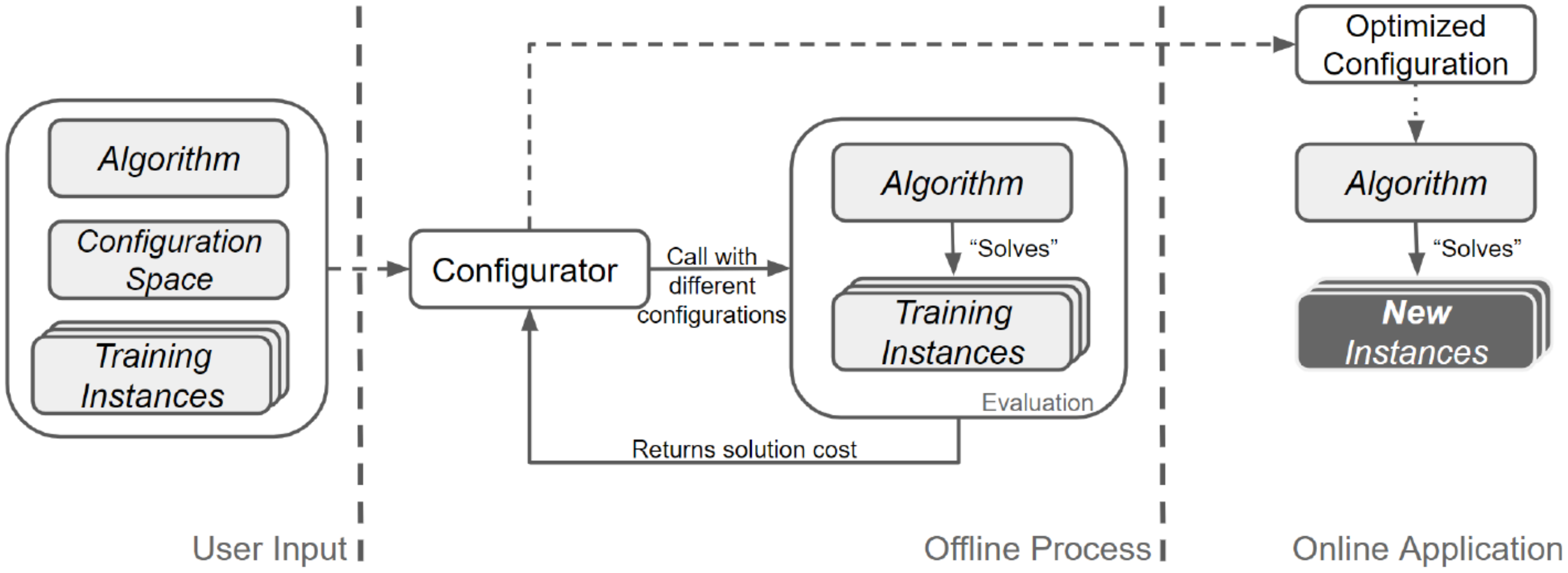

Noor
Awad


Steven
Adriaensen


Gresa
Shala


Theresa
Eimer

The Algorithm Configuration (AC) Problem



Definition: Algorithm Configuration (AC)

Given:

- a parameterized algorithm A with configuration space $\boldsymbol{\Theta}$
- a distribution $\mathcal{D}$ over problem instances with domain $\mathcal{I}$
- a cost metric $c : \boldsymbol{\Theta} \times \mathcal{I} \rightarrow \mathbb{R}$ assessing the cost of a config. $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ on a instance $i \in \mathcal{I}$

Find: $\boldsymbol{\theta}^* \in \arg\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \mathbb{E}_{i \sim \mathcal{D}} [c(\boldsymbol{\theta}, i)]$

What Can be Parameters in Planning?

Examples

- Heuristics
  - Which heuristics to use
  - Subparameters of each heuristic
  - How to combine the heuristics
- Search strategy
  - Global / local search
  - Randomization
  - How to combine them
- Problem encoding
  - Domain model
  - Problem model

In general

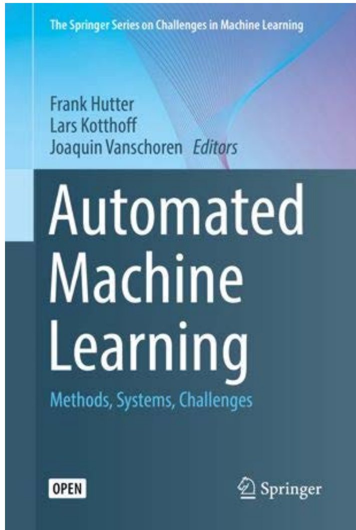
- Any design decision for which you have more than 1 alternative
- Parameter types
  - Boolean, categorical, integer, continuous
  - Conditional: only active dependent on setting of other parameters
- Often, parameters give rise to a high-dimensional structured space
  - E.g., LPG: 62 parameters, 6.5×10^{17} configurations

AC is a Useful Abstraction: Improvements in Many Areas

Domain	Algorithm	#params	#configurations	Speedup factor	Reference
SAT	Spear	26	8.3×10^{17}	$4.50\times$–$500\times$	[Hutter et al, FMCAD 2007]
MIP	CPLEX	76	1.9×10^{47}	$2.0\times$–$52\times$	[Hutter et al, CPAIOR 2010]
MPE	GLS+	5	1680	>360	[Hutter et al, AAAI 2007]
Time-tabling	UBC-TT	18	1.0×10^{13}	$\geq 28\times$	[Fawcett et al, TR 2009]
AI Planning	FastDownward	45	3.0×10^{13}	$1.0\times$–$23\times$	[Fawcett et al, ICAPS-PAL 2011]
AI Planning	LPG	62	6.5×10^{17}	$3.0\times$–$118\times$	[Vallati et al, SOCS 2013]
AI Planning	Domain configuration	109	∞	$1.0\times$–$339\times$	[Vallati et al, IJCAI 2015]
AI Planning	Problem configuration	26	∞	$1.0\times$–$39\times$	[Vallati & Serina, ICAPS 2018]

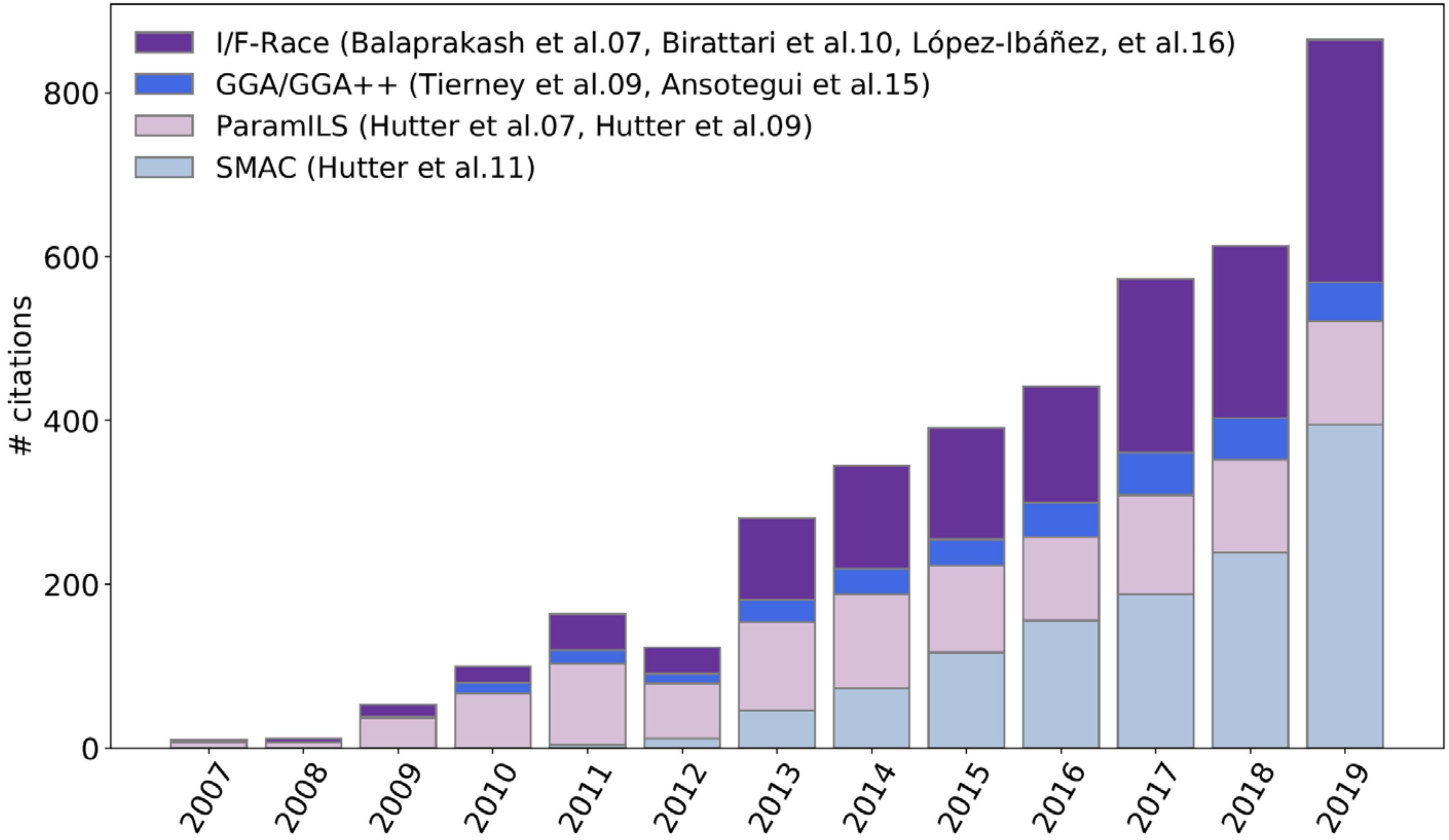
AC is also a key enabling technology in automated machine learning (AutoML), e.g.:

- Auto-WEKA [Thornton et al, KDD 2013]
- Auto-sklearn [Feurer et al, NeurIPS 2015]
- Auto-PyTorch [Zimmer et al, arXiv 2020]



AC is a Useful Abstraction: Increasingly Popular

AC is increasingly popular (citation numbers from Google scholar)



- Iterated F-Race
  - Sampling based
- GGA/GGA++
  - Genetic algorithm
- ParamILS
  - Local search
- SMAC
  - Bayesian optimization
- All these algorithms are available through a unified interface in AClib

Empirical Evaluation of AC Methods in AClib

[Hutter et al, 2020]

	Scenario	Default	SMAC	ParamILS	GGA++	GGA	IRACE
Mixed integer programming	CPLEX on Regions200	10.98	**3.45**	3.66	10.98	10.98	7.33
	CPLEX on COR-LAT	22.71	**7.44**	23.89	22.71	22.71	22.27
	CPLEX on RCW2	72.66	**64.29**	**71.38**	72.66	72.66	72.66
AI planning	LPG on Depots	35.01	**0.82**	4.52	−−	−−	35.01
	LPG on Satelitte	18.68	**6.30**	6.54	−−	−−	18.68
	LPG on Zenotravel	26.7	**1.75**	3.23	−−	−−	26.70
Boolean satisfiability solving (SAT)	Cadical on Circuit Fuzz	397.11	**303.12**	**302.24**	537.54	408.73	445.95
	Lingeling on Circuit Fuzz	319.73	**258.33**	**281.37**	574.83	430.55	−−
	Clasp on Queens	713.5	**6.33**	28.41	−−	−−	58.80
	Clasp on 3CNF-v350	332.33	**43.68**	50.88	−−	−−	47.81
	ProbSAT on 5SAT500	3000	**1.94**	**1.96**	5.32	4.15	**2.06**
TSP	LKH on TSP-Rue-1000-3000	340.31	**253.20**	325.02	647.80	340.31	335.03
Answer set programming (ASP)	Clasp on Ricochet	83.78	**56.93**	84.38	−−	−−	93.64
	Clasp on Riposte	5.65	**0.81**	3.03	−−	−−	5.65
	Clasp on Weighted Sequence	979.13	**96.63**	575.53	−−	−−	857.08

Sequential Model-based AC (SMAC): high-level overview

[Hutter et al, LION 2011]

Algorithm 1: SMAC (high-level overview)

Initialize by executing some runs and collecting their performance data

repeat

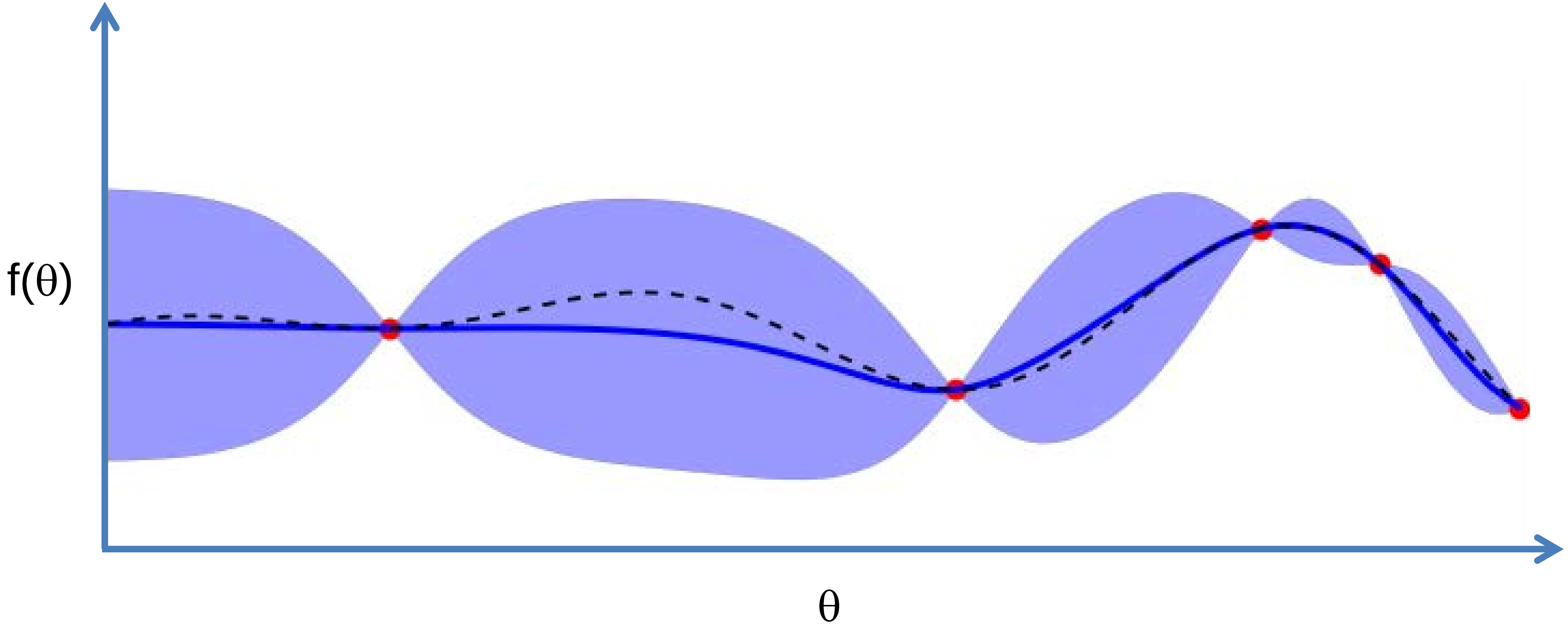
  Learn a model $\hat{m}$ from performance data so far: $\hat{m} : \mathbf{\Theta} \times \mathcal{I} \rightarrow \mathbb{R}$

  Use model $\hat{m}$ to select promising configurations $\mathbf{\Theta}_{new}$ $\rightsquigarrow$ **Bayesian optimization**

  Compare $\mathbf{\Theta}_{new}$ against best configuration so far by executing new algorithm runs

until *time budget exhausted*

Bayesian Optimization Visualized



Sequential Model-based AC (SMAC): high-level overview

[Hutter et al, LION 2011]

Algorithm 1: SMAC (high-level overview)

Initialize by executing some runs and collecting their performance data

repeat

Learn a model $\hat{m}$ from performance data so far: $\hat{m} : \mathbf{\Theta} \times \mathcal{I} \rightarrow \mathbb{R}$

Use model $\hat{m}$ to select promising configurations $\mathbf{\Theta}_{new}$

$\rightsquigarrow$ **Bayesian optimization with random forests**

Compare $\mathbf{\Theta}_{new}$ against best configuration so far by executing new algorithm runs

$\rightsquigarrow$ **How many instances to evaluate for $\theta \in \Theta_{new}$?**

until *time budget exhausted*

Saving time: aggressive racing

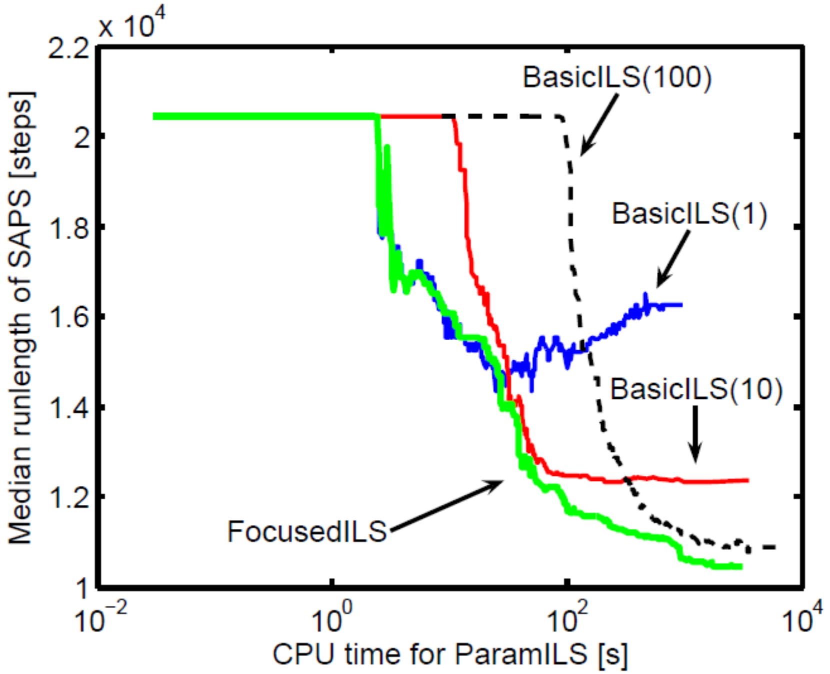
[Hutter et al, JAIR 2009]

Using a fixed number of N instances is suboptimal
- Large N: too slow
- Small N: too noisy, overfitting

Adaptive choice of N (in FocusedILS & SMAC)
- Start with N=1, reject aggressively
- Increase only for good configurations

Theorem

Let Θ be finite. Then, when using aggressive racing, the probability that ParamILS and SMAC find the true optimal parameter configuration approaches 1.

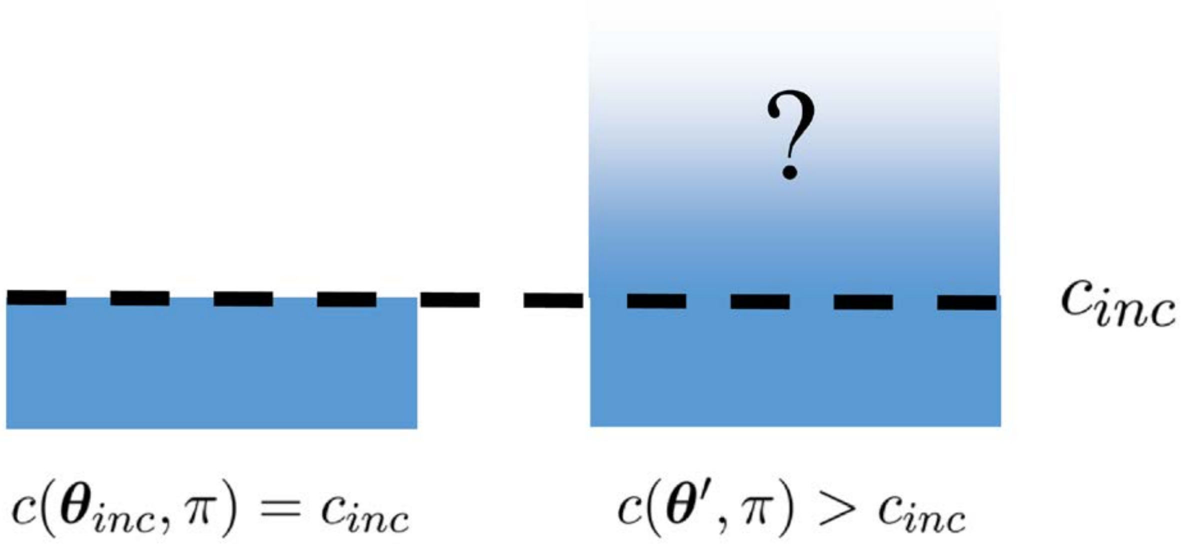


ParamILS on a single QWH instance
(test performance with 1000 new seeds)

Saving More Time: Adaptive Capping

[Hutter et al, JAIR 2009]

- Poor configurations often take a very long time (e.g., 1h vs. 1s)
- We can cap their evaluation when we know them to be worse than the incumbent



Theorem

Let Θ be finite. Then, when using aggressive racing and adaptive capping, the probability that ParamILS and SMAC find the true optimal parameter configuration approaches 1.

Sequential Model-based AC (SMAC): high-level overview

[Hutter et al, LION 2011]

Algorithm 1: SMAC

Initialize by executing some runs and collecting their performance data

repeat

Learn a model $\hat{m}$ from performance data so far: $\hat{m} : \mathbf{\Theta} \times \mathcal{I} \rightarrow \mathbb{R}$

Use model $\hat{m}$ to select promising configurations $\mathbf{\Theta}_{new}$

$\rightsquigarrow$ **Bayesian optimization with random forests**

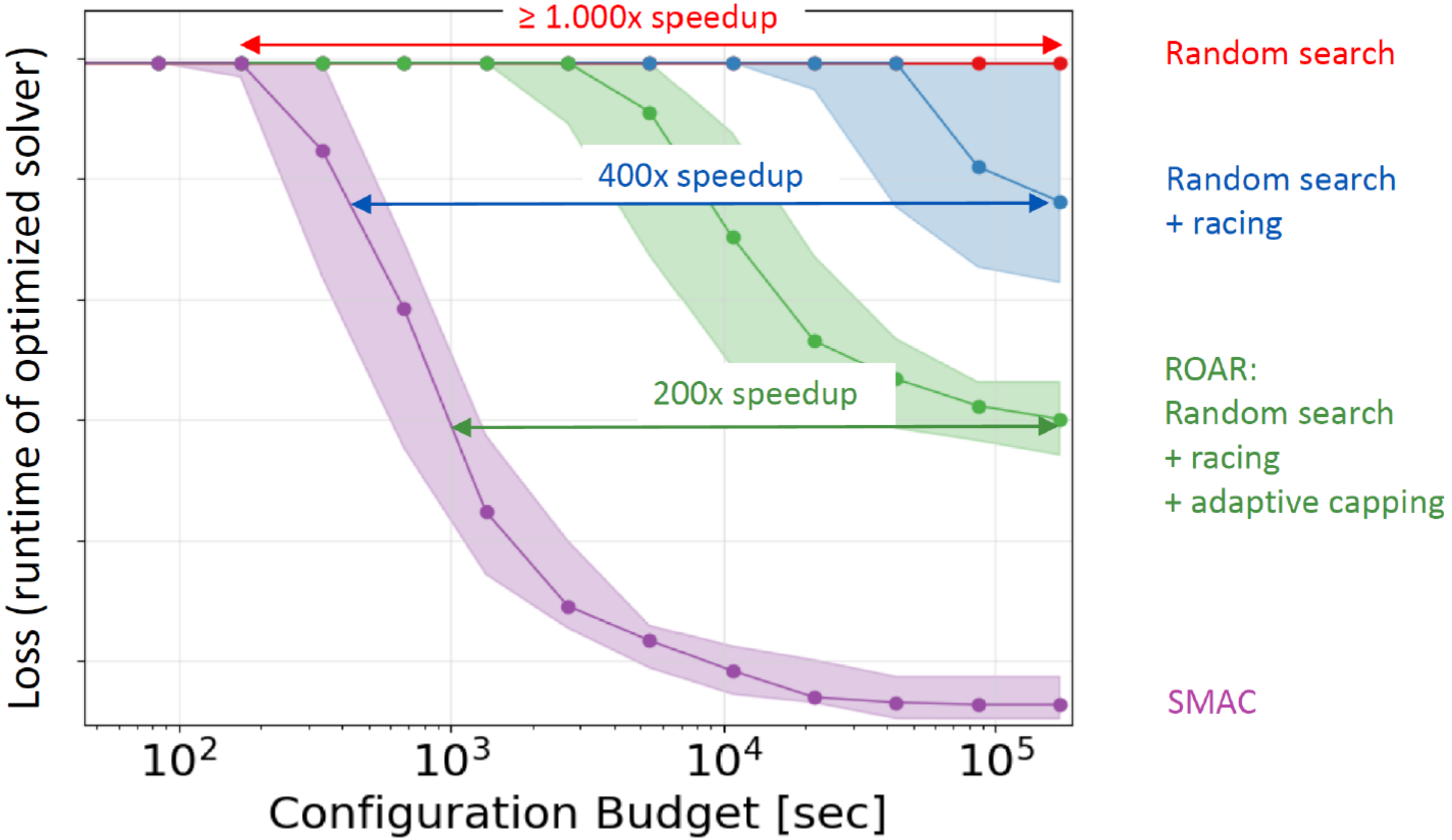
Compare $\mathbf{\Theta}_{new}$ against best configuration so far by executing new algorithm runs

$\rightsquigarrow$ **Aggressive racing and adaptive capping**

until *time budget exhausted*

All of these components matter for performance

[Hutter et al, 2020]

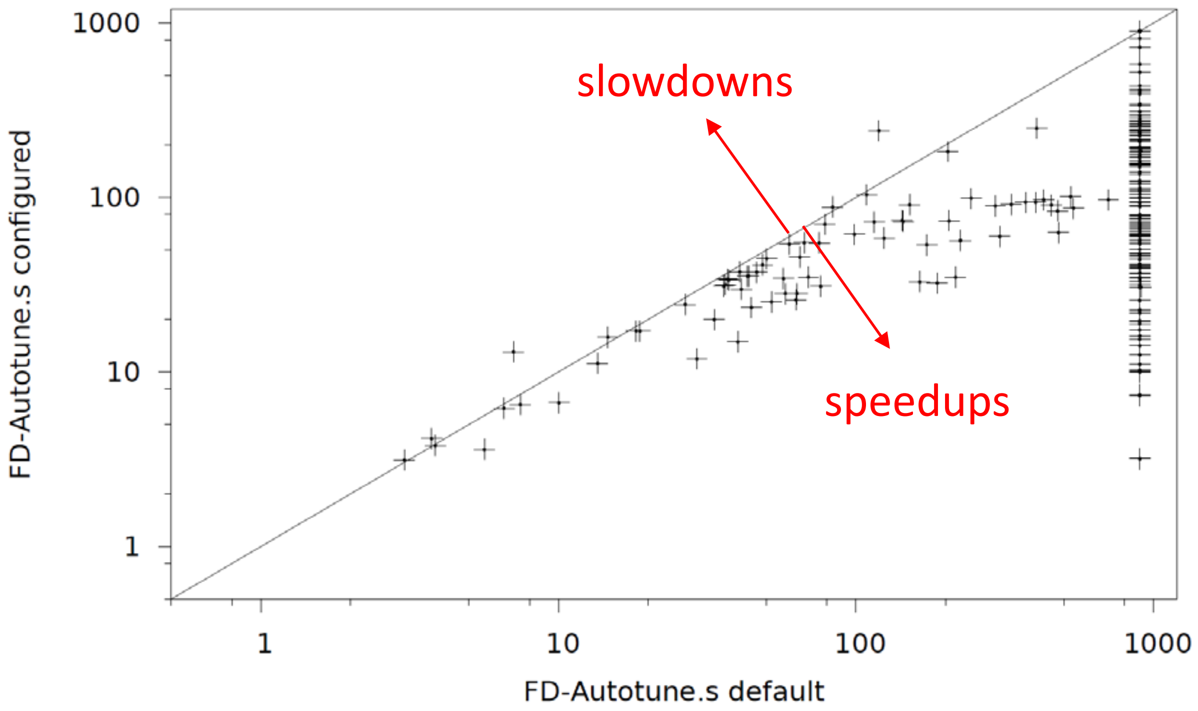


Example: optimizing CPLEX on combinatorial auctions (Regions-100)

AC Application #1: FD-AutoTune

[Fawcett et al, ICAPS-PAL 2011]

- Parameter space for Fast Downward
  - Choice of heuristics & subparameters
    - h_{lm} (×12)
    - h_{lmcut} (×2)
    - h_{add} (×3)
    - h_{cg} (×3)
    - h_{cea} (×3)
    - h_{ff} (×3)
    - h_{goal_count} (×3)
    - h_{mas} (×4)
    - h_{hm} (×2)
    - h_{blind}
    - h_{max}
  - Search
    - 8 additional parameters
  - In total: 45 params, 2.99×10^{13} configs
- Domain-wise configuration with FocusedILS



Result: over 10x speedup on average

Per domain: 1x – 23x speedup

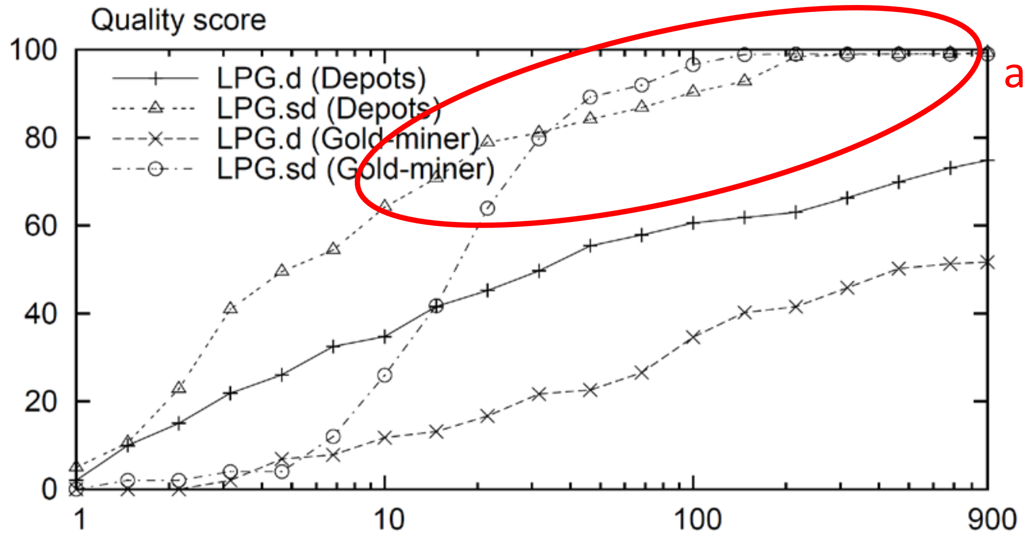
AC Application #2: Configuration of LPG

[Vallati et al, SOCS 2013]

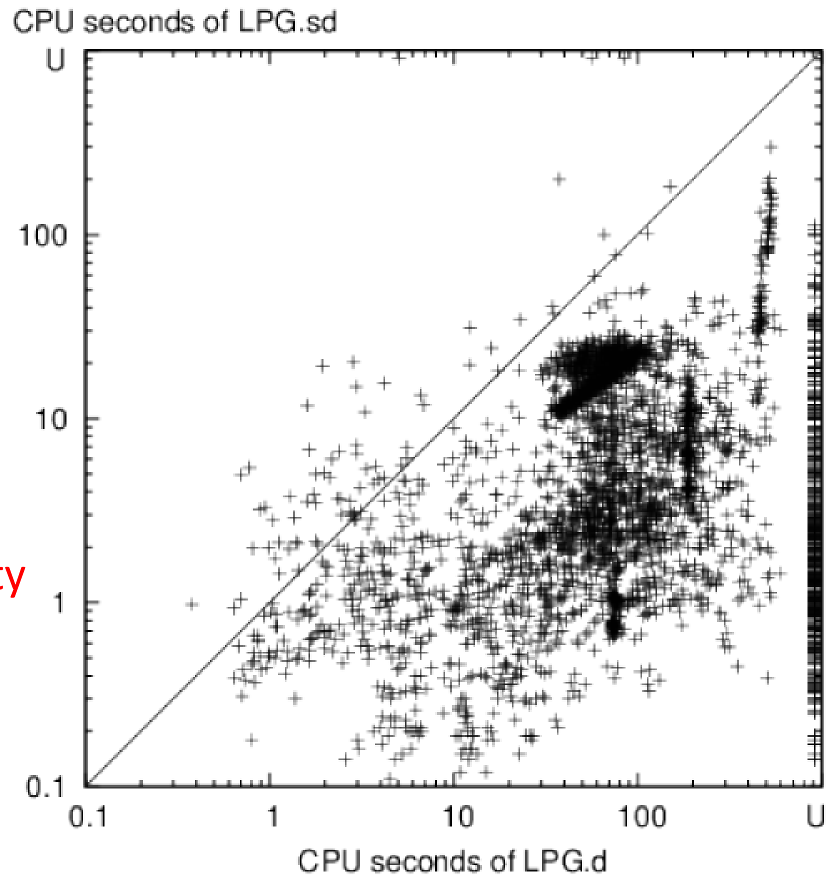
- Parameter space for LPG (local search on linear action graph)
  - Preprocessing (×6)
  - Search strategy (×15)
  - Flaw selection strategy (×8)
  - Search neighbourhood (×6)
  - Heuristic funcion (×17)
  - Reachability information (×7)
  - Search randomization (×3)

  In total: 62 params, 6.5×10^{17} configs

- Domain-wise configuration with FocusedILS



Configuration can also improve quality



Result: over 10x speedup on average

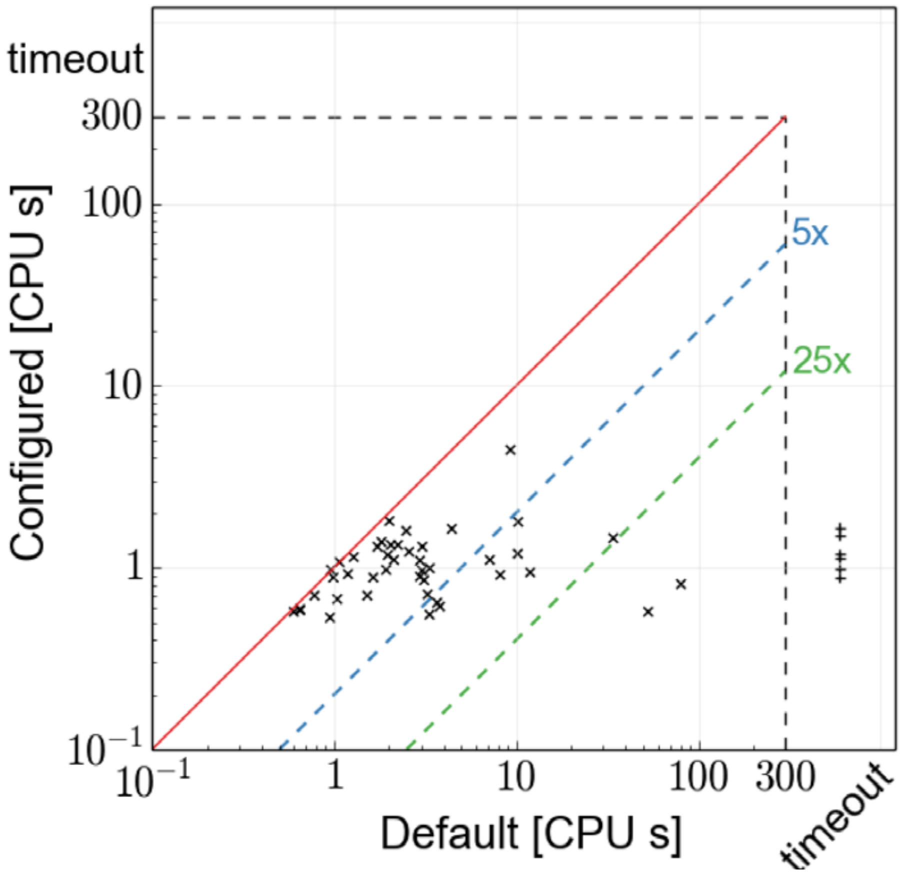
Per domain: 3x – 118x speedup

AC Application #3: Domain Model Configuration

[Vallati et al, IJCAI 2015]

- Parameter space for any planner, for how to rewrite the PDDL file
  – Order of domain predicates
  – Order of operators
  – Within each operator:
    - Order of preconditions
    - Order of postconditions
  – Up to 109 continuous parameters configured with SMAC

- Analysis can provide useful information to effectively engineer domain models
  – fANOVA parameter importance suggests:
    - First list operators that are used most/early
    - First list preconditions unlikely to be satisfied



Yahsp on Depots

Per domain: 1x – 339x speedup

AC Application #4: Problem Model Configuration

[Vallati & Serina, ICAPS 2018]

- For any planner, how to rewrite the problem model file

Original: (on-table A), (on-table B), (on C A), (clear C), (clear B), (handempty)	**Configured:** (on C A), (on-table B), (on-table A), (clear B), (clear C), (handempty)

- Need a domain-specific heuristic that applies for all problems in the domain
  - Construct a parameterized heuristic using features of facts in the planning encoding graph (PEG)
  - Configure the heuristic‘s 26 parameters by SMAC

- Per domain: 1x – 39x speedup

- Analysis can provide useful information to effectively engineer problem models
  - fANOVA parameter importance suggests:
    - Initial and goal states‘ ordering should be aligned
    - First list propositional facts that often occur in preconditions & often occur positively
    - First list propositional facts that are most connected in the PEG

Outline

- Part 1: an overview of previous meta-algorithmic approaches
  - Algorithm Configuration
  - Algorithm Portfolios
- Part 2: Dynamic Algorithm Configuration

Algorithm Portfolios in Planning

- No single algorithm or parameter setting works best everywhere
  $\rightarrow$ Exploit the complementary strengths of different planners

- Algorithm schedules
  – Very popular in planning
  – First work on schedules already goes back two decades! [Howe et al, ECP 1999]
  – Fast Downward Stonesoup [Helmert et al, ICAPS-WS 2011] has been very successful in the IPC

- Algorithm selection
  – Has been less popular in planning
  – IBaCoP [Cenamor et al, IPC 2012, ICAPS-PAL 2013 & JAIR 2016]
    - Per-instance selection of top algorithms (to be combined in a schedule)

Instance Features for Planning Tasks

[Fawcett et al, ICAPS 2014]

- How can we characterize the fingerprint of a planning instance?
- 311 features from several categories
  - PDDL features by Roberts et al [ICAPS 2008]
  - FDR features (from translation to finite domain representation)
  - Causal and domain transition graph features by Cenamor et al [ICAPS-PAL 2013]
  - LPG preprocessing
  - Torchlight search sampling
  - FD probing from running FastDownward for 1s
  - SAT representation
  - Success & timing
- Better results with more features (based on random forests [Hutter et al, AIJ 2014])

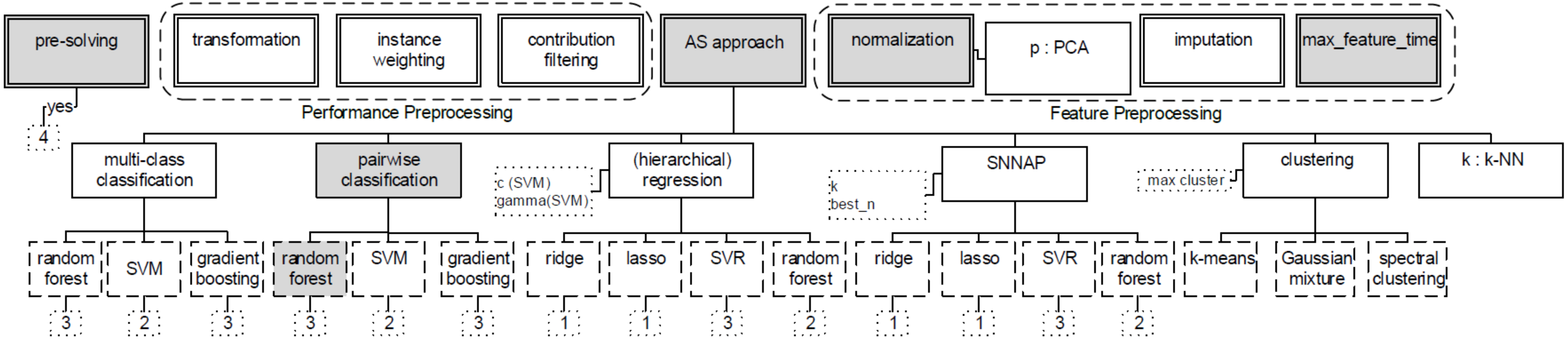
Algorithm Selection with Deep Learning

- Delfi [Katz et al, IPC 2018; Sievers et al, AAAI 2020]
  - Algorithm selection with CNNs based on an image encoding of abstract structure graph
- Simple graph features [Ferber, ICAPS WS 2020; Ferber & Seipp, ICAPS WS 2020]
  - Simple ML techniques perform similarly with simple statistics of the graph
- Graph convolutional neural networks (GCNs) [Ma et al, AAAI 2020]
  - Perform better than CNNs on graph encoding

AutoFolio: Configuring an Algorithm Selector

[Lindauer et al, JAIR 2015]

- Outside of planning, many more algorithm selection methods exist
- We spanned a design space over them: 54 parameters



- Used SMAC to find instantiation with best cross-validation performance
- Won ICON challenge on algorithm selection, categories #solved & PAR10

Combining AC + Portfolios

- AC and portfolios have opposite strengths
  - AC finds great configurations for homogeneous instance distributions
  - Portfolios take these as inputs to address heterogeneous distributions

- Combining AC & algorithm selection (per-instance AC)
  - ISAC [Kadioglu et al, ECAI 2010]
    - Cluster instances, use AC for each cluster
  - Hydra [Xu et al, AAAI 2010; IJCAI-RCRA 2011]
    - Use AC to search for the configuration maximally improving an algorithm selector

- Combining AC & algorithm schedules
  - Seipp et al [ICAPS 2012]
    - Use AC for several planning domains; combine the result into a schedule with uniform time shares
  - Cedalion [Seipp et al, AAAI 2015]
    - Similar to Hydra, but for schedules: search for configuration + time slot to add

Outline

- Part 1: an overview of previous meta-algorithmic approaches
  - Algorithm Configuration
  - Algorithm Portfolios

Part 2: Dynamic Algorithm Configuration

Which Planning Parameters Could be Adapted?

- Heuristics
  - Which heuristics to use
  - Subparameters of each heuristic
  - How to combine the heuristics
- Search strategy
  - Global / Local search
  - Randomization
  - How to combine them
- Problem encoding
  - Domain model
  - Problem model
- LPG‘s local search parameters
- Further promising parameters
  - Merge strategies for merge & shrink
  - In general, when to do X
    - E.g., when to derive a new heuristic, when to discard an old one

Others Have Also Thought About Reacting to State

- Early pioneering work by Lagoudakis & Littmann
  - Lagoudakis, Littmann & Parr [2001]: State-specific selection of sorting algorithm
  - Lagoudakis & Littmann [2004a]: State-specific selection of branching rules in DPLL for SAT
  - Lagoudakis & Littmann [2004b]: Reinforcement Learning for Algorithm Selection

- Very recent related work in AI planning by Gomoluch et al
  - Policy gradient for learning to switch between search methods [Gomoluch et al, ICAPS 2019]
    - Tabular state space (4 states) and action space (5 search methods)
  - Blackbox optimization of neural search policy [Gomoluch et al, ICAPS 2020]
    - Adaptive parameterization of mix between global & local best first search and random moves

Dynamic Algorithm Configuration (DAC)

Definition: Algorithm Configuration (AC)

Given:

- a parameterized algorithm A with configuration space $\boldsymbol{\Theta}$
- a distribution $\mathcal{D}$ over problem instances with domain $\mathcal{I}$
- a cost metric $c : \boldsymbol{\Theta} \times \mathcal{I} \to \mathbb{R}$ assessing the cost of a config. $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ on a instance $i \in \mathcal{I}$

Find: $\boldsymbol{\theta}^* \in \arg\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \mathbb{E}_{i \sim \mathcal{D}} [c(\boldsymbol{\theta}, i)]$

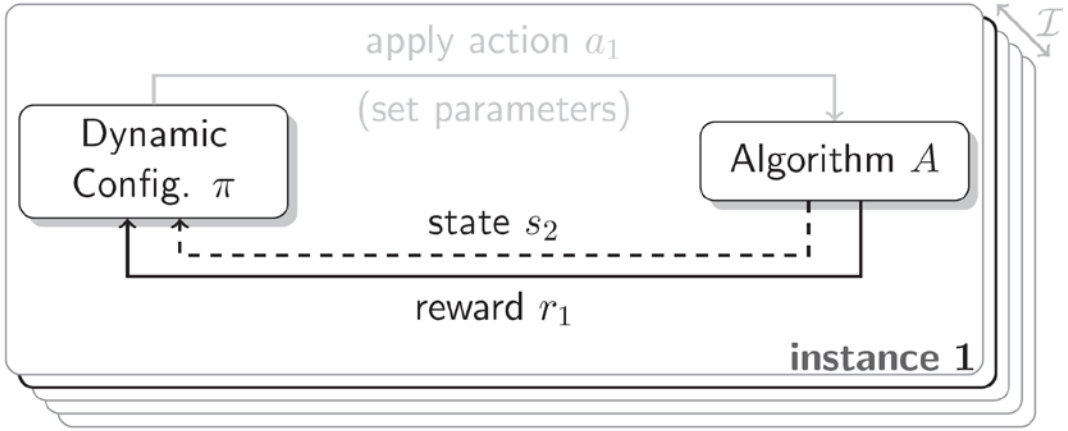
Definition: Dynamic AC (DAC)

Given:

- a parameterized algorithm A with configuration space $\boldsymbol{\Theta}$
- a distribution $\mathcal{D}$ over problem instances with domain $\mathcal{I}$
- A space of dynamic configuration policies $\pi \in \Pi$ with $\pi : \mathcal{S} \times \mathcal{I} \to \boldsymbol{\Theta}$ that adaptively choose a configuration $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ for each instance $i \in \mathcal{I}$ and state $s \in \mathcal{S}$ of A
- a cost metric $c : \Pi \times \mathcal{I} \to \mathbb{R}$ assessing the cost of a policy $\pi \in \Pi$ on a instance $i \in \mathcal{I}$

Find: $\pi^* \in \arg\min_{\pi \in \Pi} \mathbb{E}_{i \sim \mathcal{D}} [c(\pi, i)]$

DAC as a Contextual Markov Decision Process



DAC as a contextual MDP

DAC can be formalized as a contextual MDP $\mathcal{M}_\mathcal{I} = \{\mathcal{M}\}_{i \sim \mathcal{I}}$, where each $\mathcal{M}_i$ is an MDP:

- State Space $\mathcal{S}$
- Action Space $\boldsymbol{\Theta}$
- Transition Function T_i
- Reward Function R_i

Definition: Dynamic AC (DAC)

Given:

- a parameterized algorithm A with configuration space $\boldsymbol{\Theta}$
- a distribution $\mathcal{D}$ over problem instances with domain $\mathcal{I}$
- A space of dynamic configuration policies $\pi \in \Pi$ with $\pi : \mathcal{S} \times \mathcal{I} \rightarrow \boldsymbol{\Theta}$ that adaptively choose a configuration $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ for each instance $i \in \mathcal{I}$ and state $s \in \mathcal{S}$ of A
- a cost metric $c : \Pi \times \mathcal{I} \rightarrow \mathbb{R}$ assessing the cost of a policy $\pi \in \Pi$ on a instance $i \in \mathcal{I}$

Find: $\pi^* \in \arg\min_{\pi \in \Pi} \mathbb{E}_{i \sim \mathcal{D}} [c(\pi, i)]$

DAC Strictly Generalizes All Formulations We‘ve Seen

Meta-algorithmic problem	Formally
AC	$\pi : \emptyset \to \mathbf{\Theta}$
Selection	$\pi : \mathcal{I} \to \mathcal{A}$
Schedules	$\pi : \mathbb{R}^+ \to \mathcal{A}$
Selection + schedules	$\pi : \mathcal{I} \times \mathbb{R}^+ \to \mathcal{A}$
AC + selection	$\pi : \mathcal{I} \to \mathbf{\Theta}$
AC + schedules	$\pi : \mathbb{R}^+ \to \mathbf{\Theta}$
DAC	$\pi : \mathcal{I} \times \mathbb{R}^+ \times \mathcal{S} \to \mathbf{\Theta}$

Notation reminder:

- $\mathcal{I}$: Instances
- $\mathbf{\Theta}$: configuration space
- $\mathcal{A}$: set of algorithms
- $\mathbb{R}^+$: positive real numbers (time steps)
- $\mathcal{S}$: state space
- π: policy

Proposition

The optimal DAC policy is at least as good as the optimal solution of any of the above.

Theorem

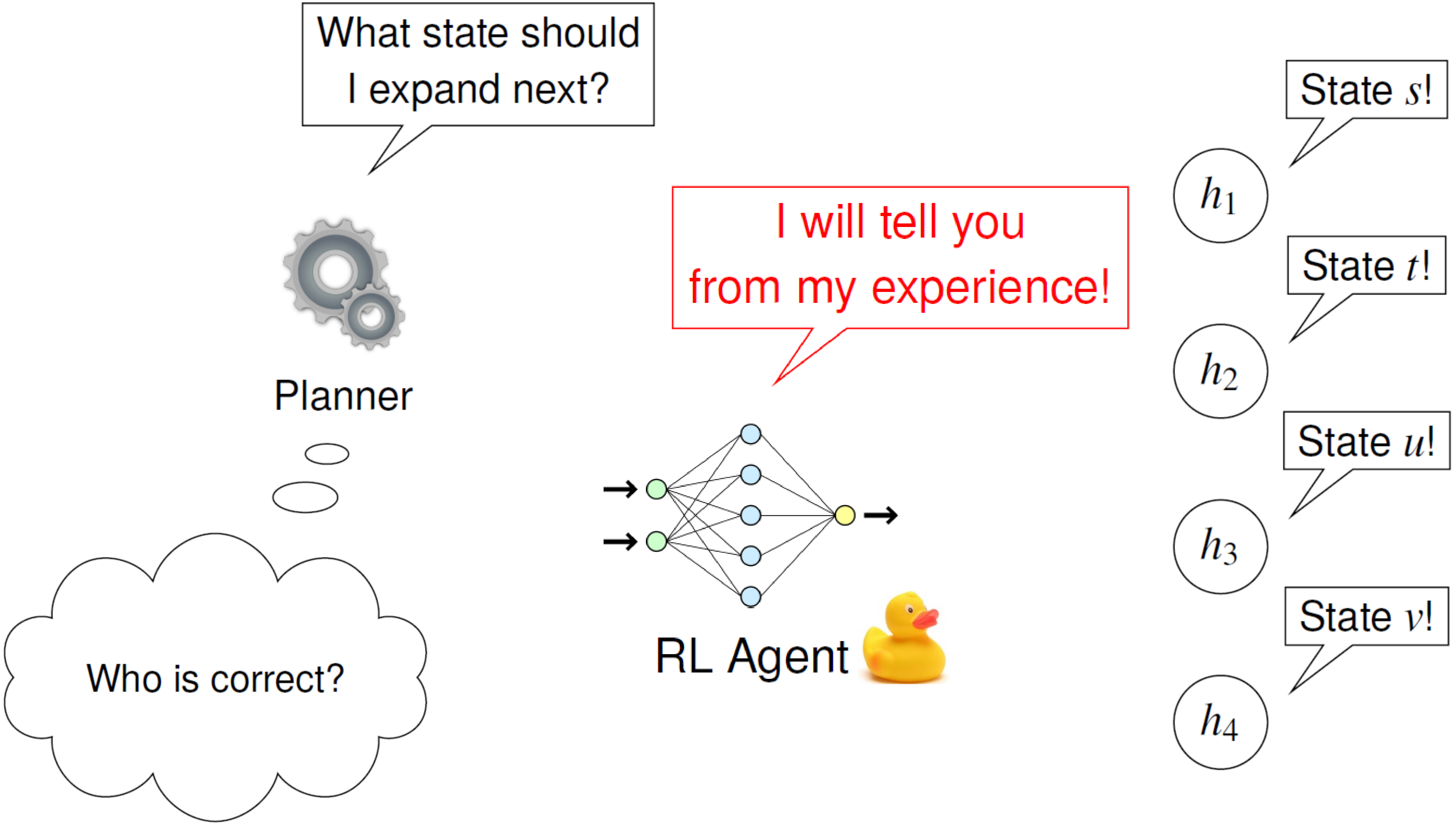
The optimal DAC policy can be exponentially better than the optimal selector or schedule.

Evidence that DAC Might Be a Useful Abstraction

- Experiments on whitebox/toy benchmarks [Biedenkapp et al, ECAI 2020]
  - Strong generalization across instances
  - Moderate scaling with number of parameters
  - Found optimal solution in a task that required using both instance & state features
- DAC for controlling the step size in CMA-ES [Shala et al, PPSN 2020]
  - Guided policy search [Levine & Kuhn, 2013], learning from an existing heuristic
- DAC for selecting heuristics in AI planning [Speck et al, ICAPS-PRL 2020]

DAC for Selecting Heuristics in AI planning

[Speck et al, ICAPS-PRL 2020]



Basics

[Speck et al, ICAPS-PRL 2020]

- Satisficing planning
  - Search for a good plan
  - Inadmissible heuristics are difficult to combine
- Greedy search with multiple heuristics [Helmert, JAIR 2006]
  - One separate open list for each heuristic
  - Each heuristic is evaluated at each step
  - Alternation strategy can be better than any single heuristic [Röger & Helmert, ICAPS 2010]
  - Can we do better than alternation?

Theorem

For each algorithm schedule π_{sched} and each algorithm selector π_{sel}, there exists a family of planning instances i_n, a collection of heuristics H and a dynamic control policy π_{dac}, so that greedy best-first search with H and π_{dac} expands exponentially less states in i_n than greedy best-first search with H and π_{sched} or π_{sel} until a plan is found.

Reinforcement Learning Setup

[Speck et al, ICAPS-PRL 2020]

- Action Space
  - 4 different heuristic functions: h_{ff} , h_{cg} , h_{cea} , h_{add}
- State space
  - Time step t
  - Simple features over the states in the open list of each considered heuristic:
    - max_h, min_h, μ_h, σ^2_h, $\#_h$
    - Actually taking the difference of each feature between t-1 and t
- Reward
  - Simply -1 for each expansion step until solution is found
- RL strategy
  - ε-greedy deep Q-learning with a double DQN [van Hasselt et al, 2015]
  - Simple feed-forward network with 2 hidden layers of 75 units each

Experiments: Coverage on Unseen Test Instances

[Speck et al, ICAPS-PRL 2020]

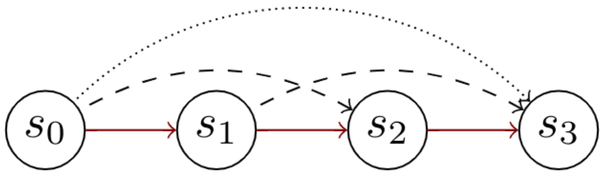
- Experimental setup: domain-wise training on 6 domains
  - 100 train & 100 test instances each
- Baselines
  - All single heuristics & oracle per-instance selector
  - Random and alternating heuristic

Algorithm	CONTROL POLICY			SINGLE HEURISTIC				BEST AS (ORACLE)
Domain (# Inst.)	RL	RND	ALT	h_{ff}	h_{cg}	h_{cea}	h_{add}	SINGLE h
BARMAN (100)	**84.4**	83.8	83.3	66.0	17.0	18.0	18.0	67.0
BLOCKSWORLD (100)	**92.9**	83.6	83.7	75.0	60.0	92.0	92.0	93.0
CHILDSNACK (100)	**88.0**	86.2	86.7	75.0	86.0	86.0	86.0	86.0
ROVERS (100)	95.2	**96.0**	**96.0**	84.0	72.0	68.0	68.0	91.0
SOKOBAN (100)	87.7	87.1	87.0	88.0	**90.0**	60.0	89.0	92.0
VISITALL (100)	56.9	51.0	51.5	37.0	**60.0**	**60.0**	**60.0**	60.0
SUM (600)	**505.1**	487.7	488.2	425.0	385.0	384.0	413.0	489.0

Further Work Under Way on DAC

Exploiting that actions often need to be repeated many times

- Learn when to act
- TempoRL [Biedenkapp et al, 2020]



Active selection of instances that are helpful in learning

- Self-paced reinforcement learning
- Making use of changes in the value function [Eimer et al, 2020]

Creating a library of DAC benchmarks

- OpenAI gym format
- We would love to include your DAC problems

Opportunities for Planning Experts #1: Action Space

Choosing the right problem

- Where is DAC likely to help most?
- Which parameters are crucial to adapt?

Constrain DAC to simple strategies

- To aid interpretability

Combinations of AC & DAC

- Configuring many static parameters and some dynamic ones

Opportunities for Planning Experts #2: Feature Space

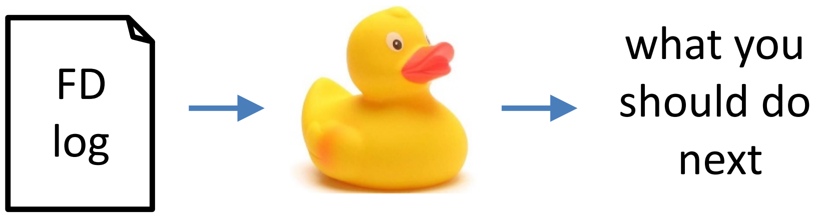
Instance features

- We have good instance features, but we don‘t actually use them yet
- These might directly allow for domain-independent planning by DAC

State features

- We only had a very first shot at these
- Better state features will improve domain-specific & domain-independent planning

Let‘s parse Fast Downward‘s log file?



Opportunities for Planning Experts #3: Insights

Use these data-driven tools to gain scientific understanding

1. Use AC/DAC to improve planner‘s performance

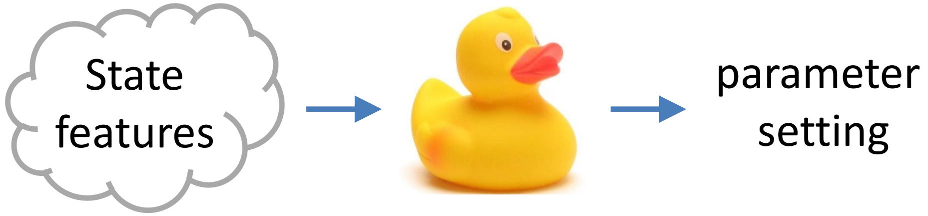
2. Use meta-algorithmic tools to understand why performance improved
   - For AC, we have automated parameter importace analysis methods
     - Forward selection [Hutter et al, LION 2013]
     - Ablation analysis [Fawcett & Hoos, 2016; Biedenkapp et al, AAAI 2017]
     - Functional ANOVA [Hutter et al, ICML 2014] → [Vallati et al, IJCAI 2015] and [Vallati & Serina, ICAPS 2018]
     - CAVE framework to automatically generate reports [Biedenkapp et al, LION 2018]
   - For DAC, we still need to come up with such methods
     - E.g., can strong yet complex policies be approximated with a simpler one? (→ Ferber & Seipp, ICAPS WS 2020])

3. Use the gained insights to develop new & better algorithms

Conclusion

- Algorithm configuration (AC) is a reliable workhorse
  - Often leads to speedups of orders of magnitudes

- Dynamic algorithm configuration (DAC) is the new kid on the block
  - Strict generalization of AC, selection & schedules
  - Also much harder (RL setting)
  - First success stories, but still at an early state



- Please join us in making DAC a great thing for the community
  - Try DAC, break DAC, improve DAC ☺
  - We‘re building a team of postdocs on DAC in Freiburg