Algorithm Configuration Survival Guide
Part 1

SIGEVO Summer School

GECCO 2022

Leslie Pérez Cáceres

leslie.perez@pucv.cl

Outline

1. Algorithm configuration: know the terrain

2. Configurators & irace: know your resources

3. Configuration examples: yes, you can!

Algorithm configuration

Know the terrain

Optimization Algorithms

They are great!

Especially when they solve problems…

… efficiently

Unfortunately, this is not easy to achieve

Algorithm design

Optimization algorithms can be powerful and **flexible** tools

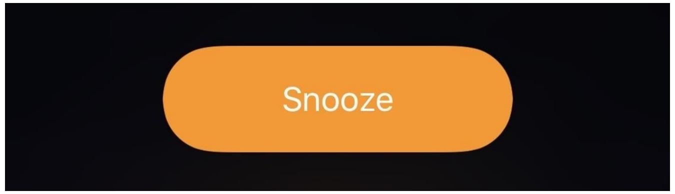
Design choices

Component selection & behavior & interaction

Parameters

Parameters are **design choices** …

… postponed until execution time



Parameters

Flexible algorithms often expose many parameters

Algorithm performance is strongly dependent on parameter settings

Algorithm configuration task

The task of finding parameter settings of a ***target algorithm***
that exhibit ***best empirical performance***
on a given distribution of ***problem instances****.*

Algorithm configuration problem

Given a budget *B*, find a configuration θ^*

$$\boxed{\theta^* = argmin_{\theta \,\epsilon\, \Theta}\ F(\theta, I)}$$

$\Theta \rightarrow$ parameter space

$I \rightarrow$ problem instance set from space P_I

$F(\theta, I) \rightarrow$ configuration objective

Algorithm configuration problem

Configuration objective:

$$F(\theta, I) = \bigwedge_{i=1}^{|I|} \mathcal{Q}(\theta, I_i, b)$$

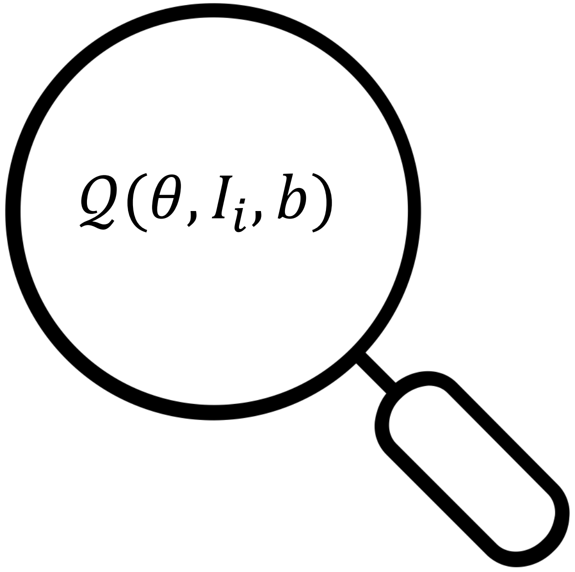
$\mathcal{Q}$ → *performance measure*

b → *termination criterion*

Sources of variability in performance

In most cases, we can only **estimate** performance

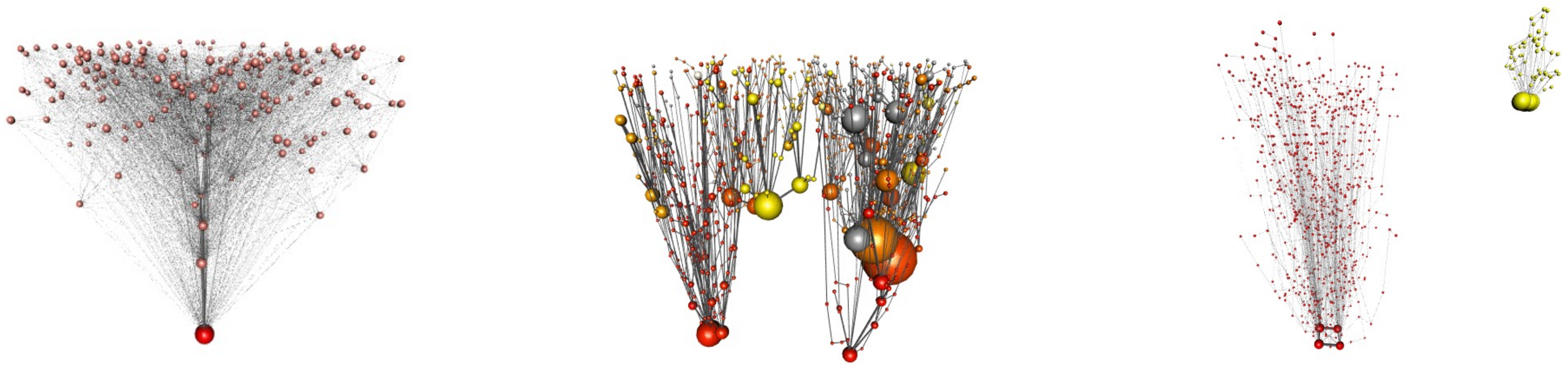
- Parameters values
- Resources: computational budget
- Instances



About instances

Instances can define very different landscapes

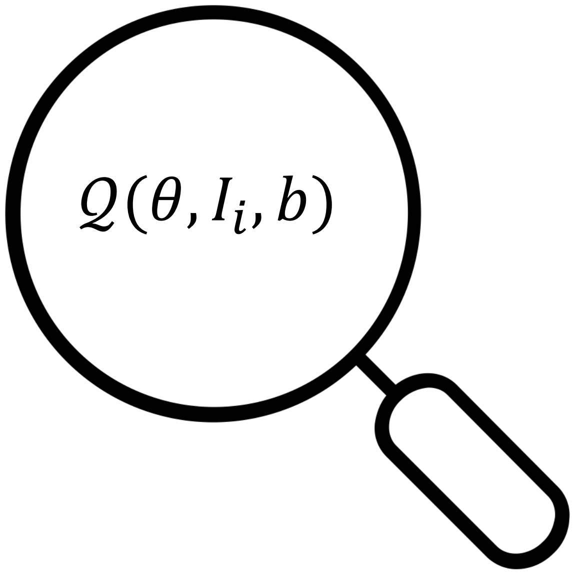
http://lonmaps.com/gallery/



Exploration / exploitation **balance** for good performance

Sources of variability in performance

In most cases, we can only **estimate** performance

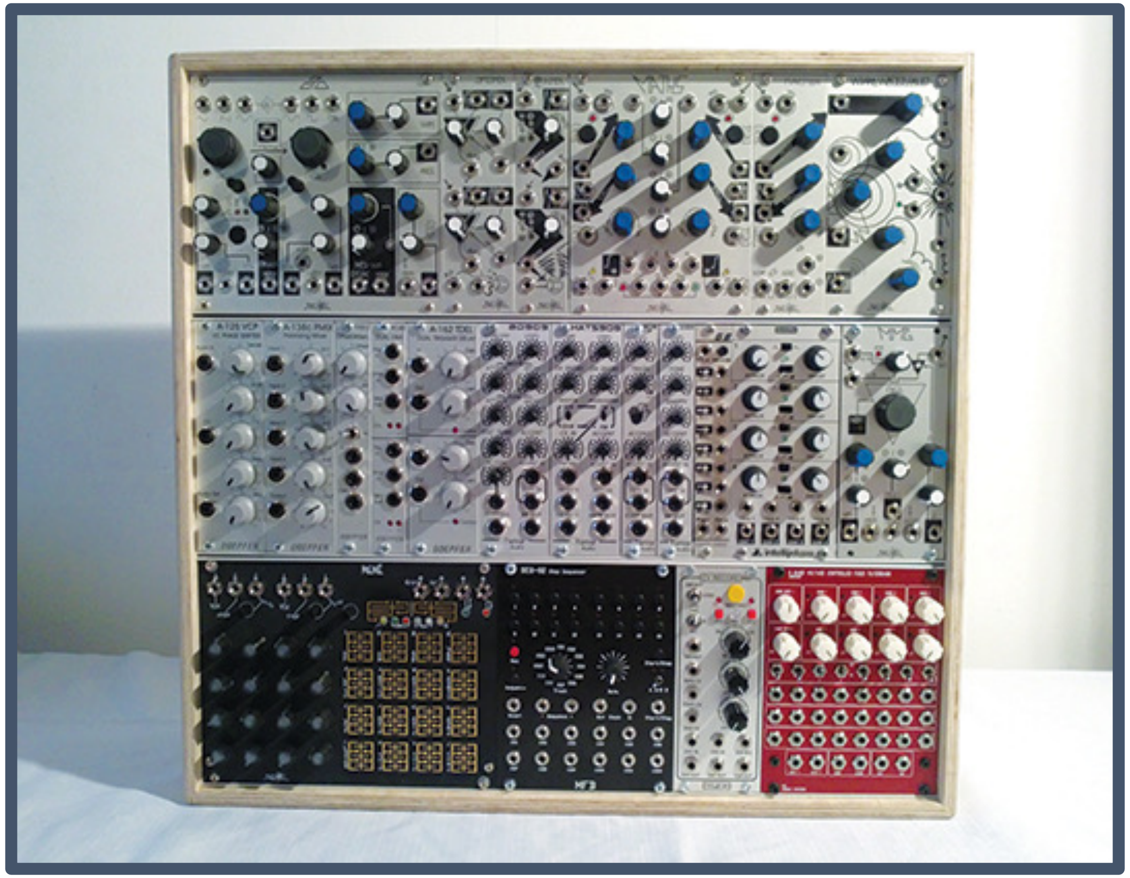


- Parameters values
- Resources: computation budget
- Instances

- Platform: platforms, executables, memory, processing loads….

- Stochasticity

Find the best configuration



How to approach the problem?

- Offline tuning: set an adequate parameter settings **before execution**
- Online tuning: set and adjust parameter settings **during execution**

Manual algorithm configuration

Based on knowledge

- Rule of thumbs
- Default settings
- Literature studies

- Requires expertise
- Makes a lot assumptions
- Based on simplifications
- No exploration

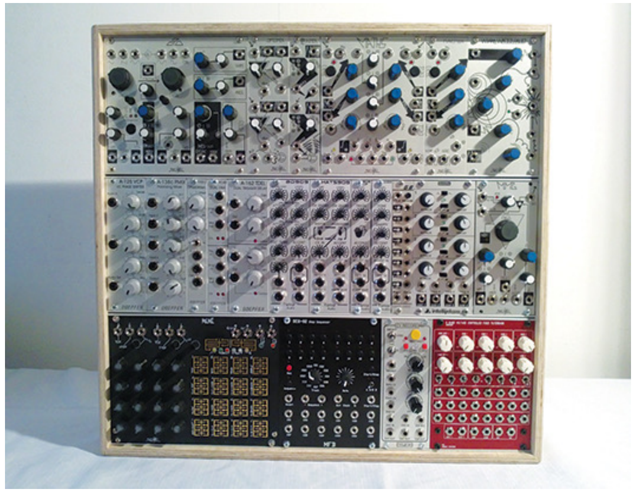
Trial and error

- Systematic or not …

- Time consuming
- Tedious process
- Prone to bias
- Limited exploration

Manual algorithm configuration

Pros and cons

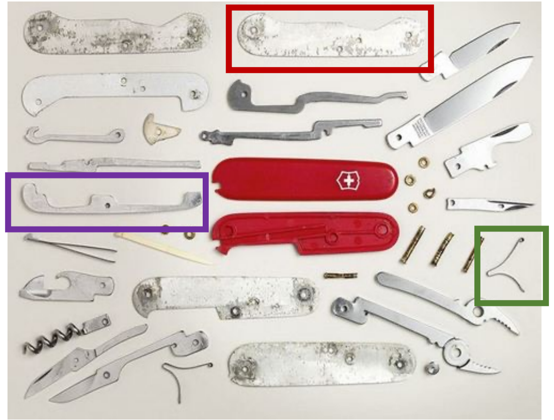


- Allows to obtain knowledge
- Good when limited resources

- Does not promote flexibility
- Avoids to focus on creativity

Automatic Algorithm Configuration

*Automatically searching for **high-performing** parameter settings **before the execution** of an algorithm.*



Apply specialized tools: configurators

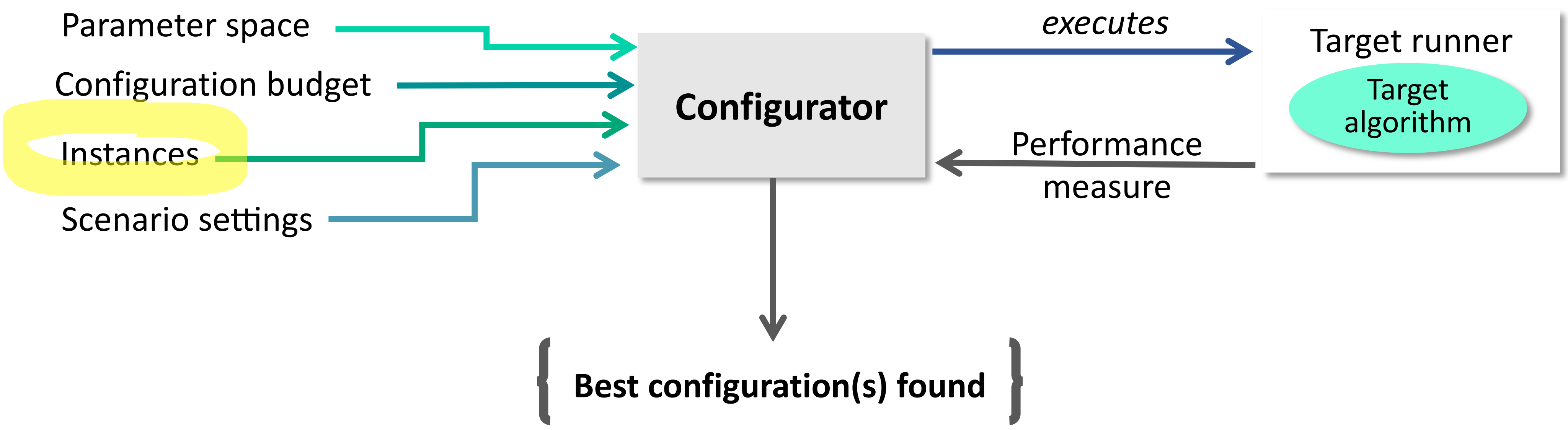
- Effectively use available computational resources

2

Configurators & irace

Know your resources

Algorithm Configurators

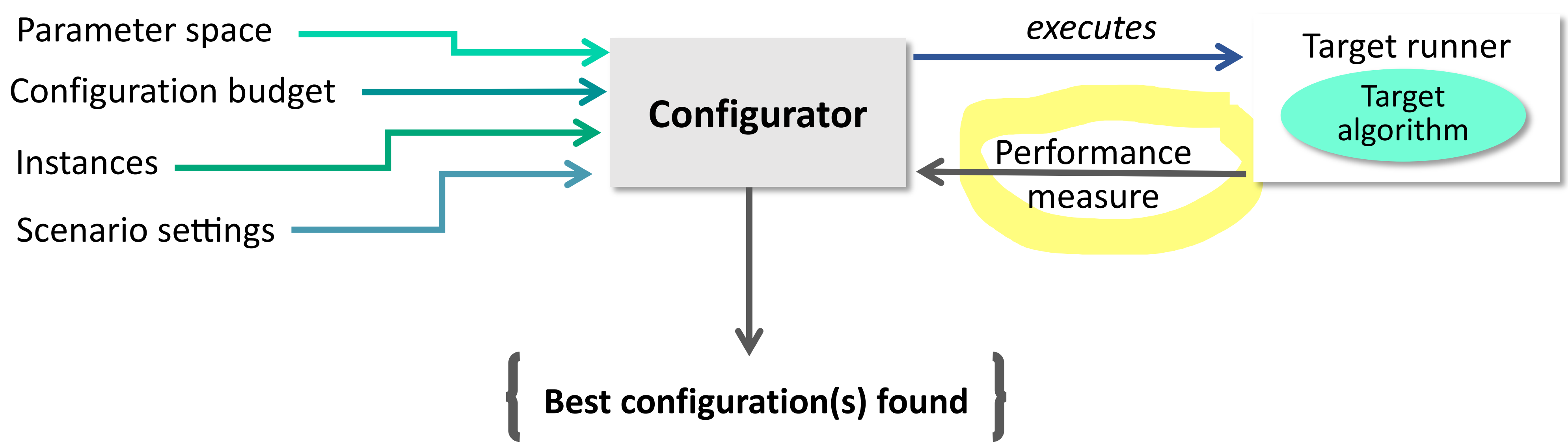


About the instances

Instances must be representative of the ones the algorithm willl encounter in production

- Training set: to perform the search for good configurations
- Test set: to evaluate configurations (assess overtuning)

Algorithm Configurators



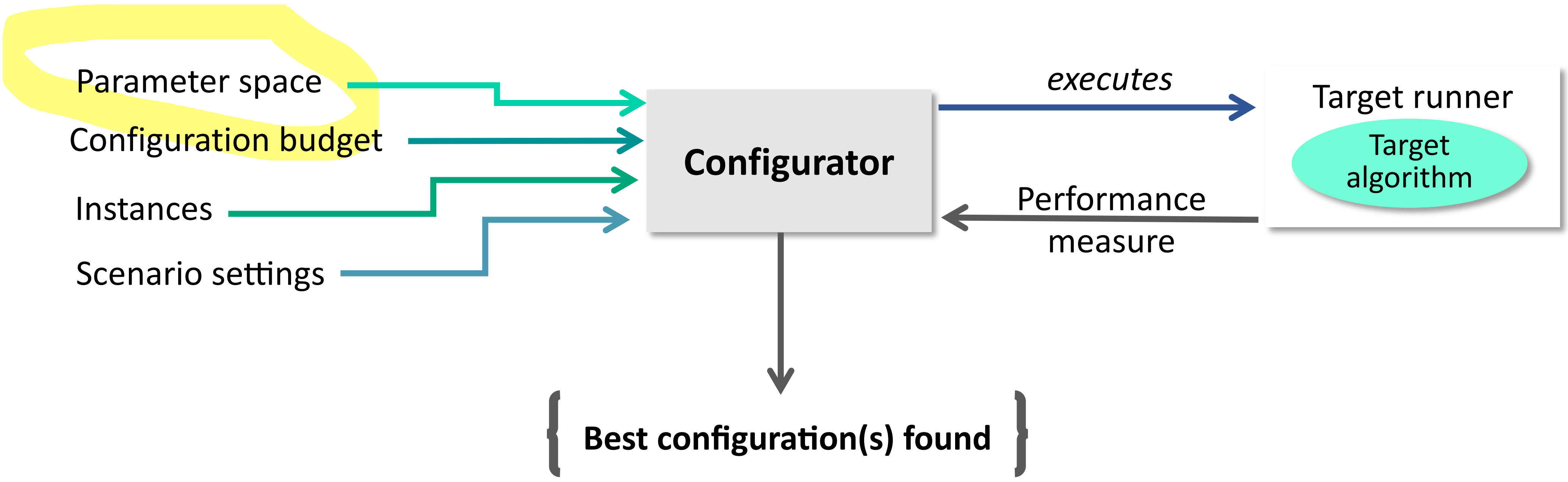
About performance measure

Configuration objective can be:

- Quality-based (solution quality, gap, etc.)
- Resource-based (time to optima, time to x quality, …)

The performance measure can be **penalized** for configuration!

Algorithm Configurators



About parameter spaces

Parameter type and domain:

- Categorical
- Ordered
- Numerical:
  - real
  - integer

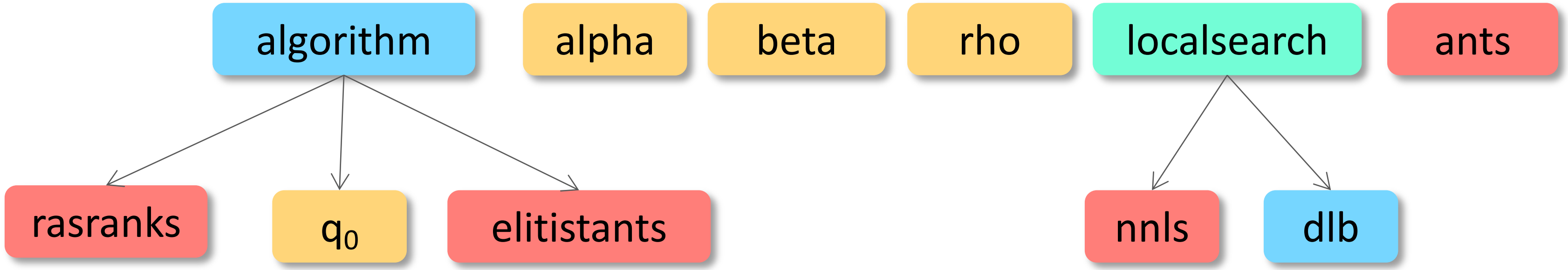
algorithm	Categorical → {AS, MMAS, ACS}
localsearch	Ordered → {none, small, medium, large}
alpha	Real → {0.0, 1.0}
ants	Integer → {10, 50}

About parameter spaces

Conditionality

- Parameters can be activated based on the value of others

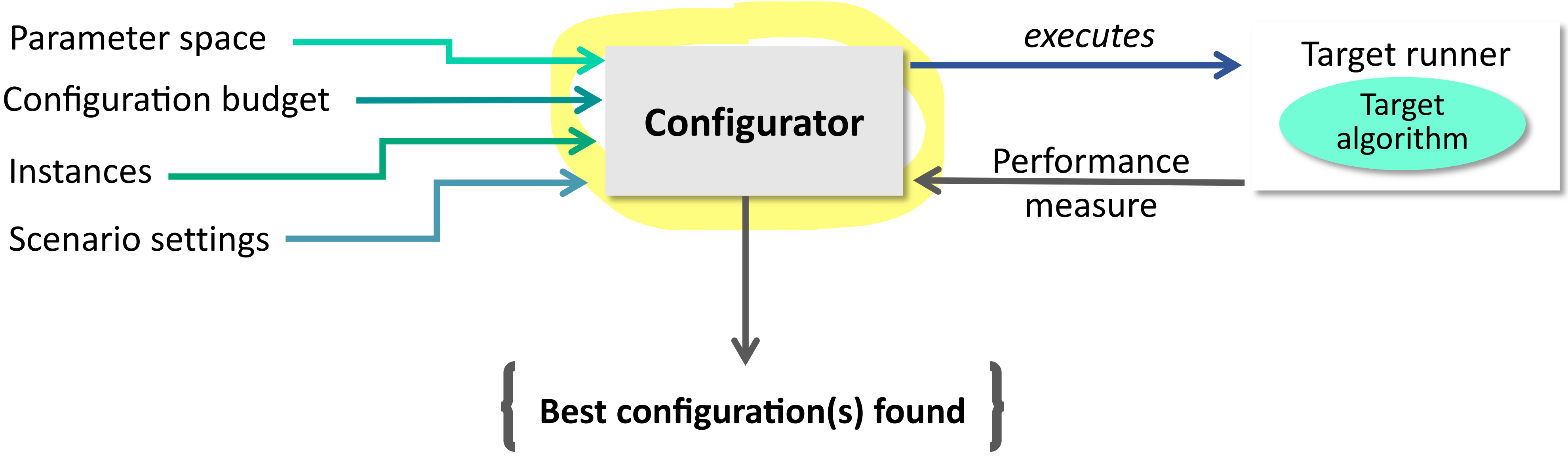
q_0 *is active only if* algorithm == *"acs"*



Algorithm configurators

- Experimental design: CALIBRA [1]
- Numerical optimization: MADS [2], CMAES [3], BOBYQA [3]
- Heuristic optimization: metaGA [4], REVAC [5], ParamILS [6], GGA [7], linear GP[8]
- Model-based: SPOT [9][10], SMAC [11], GGA++ [12]
- Sequential statistical testing: F-Race [13], Iterated F-Race [14], irace [15]

Algorithm Configurators



The irace package

The irace package: Iterated Racing for Automatic Algorithm Configuration. Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Thomas Stützle and Mauro Birattari. *Operations Research Perspectives* volume 3, pages 43–58 (2016)

Webpage:

http://iridia.ulb.ac.be/irace/

R package:

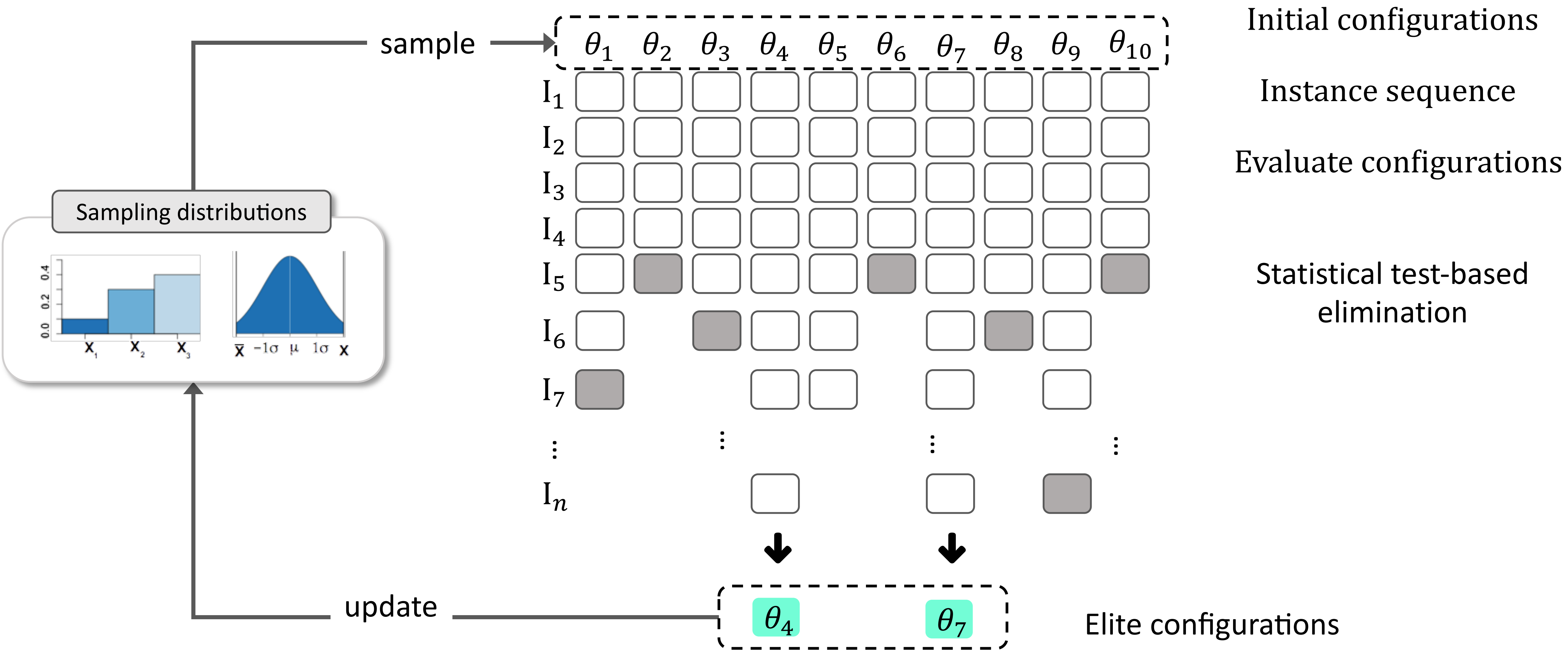
http://cran.r-project.org/package=irace

The irace package

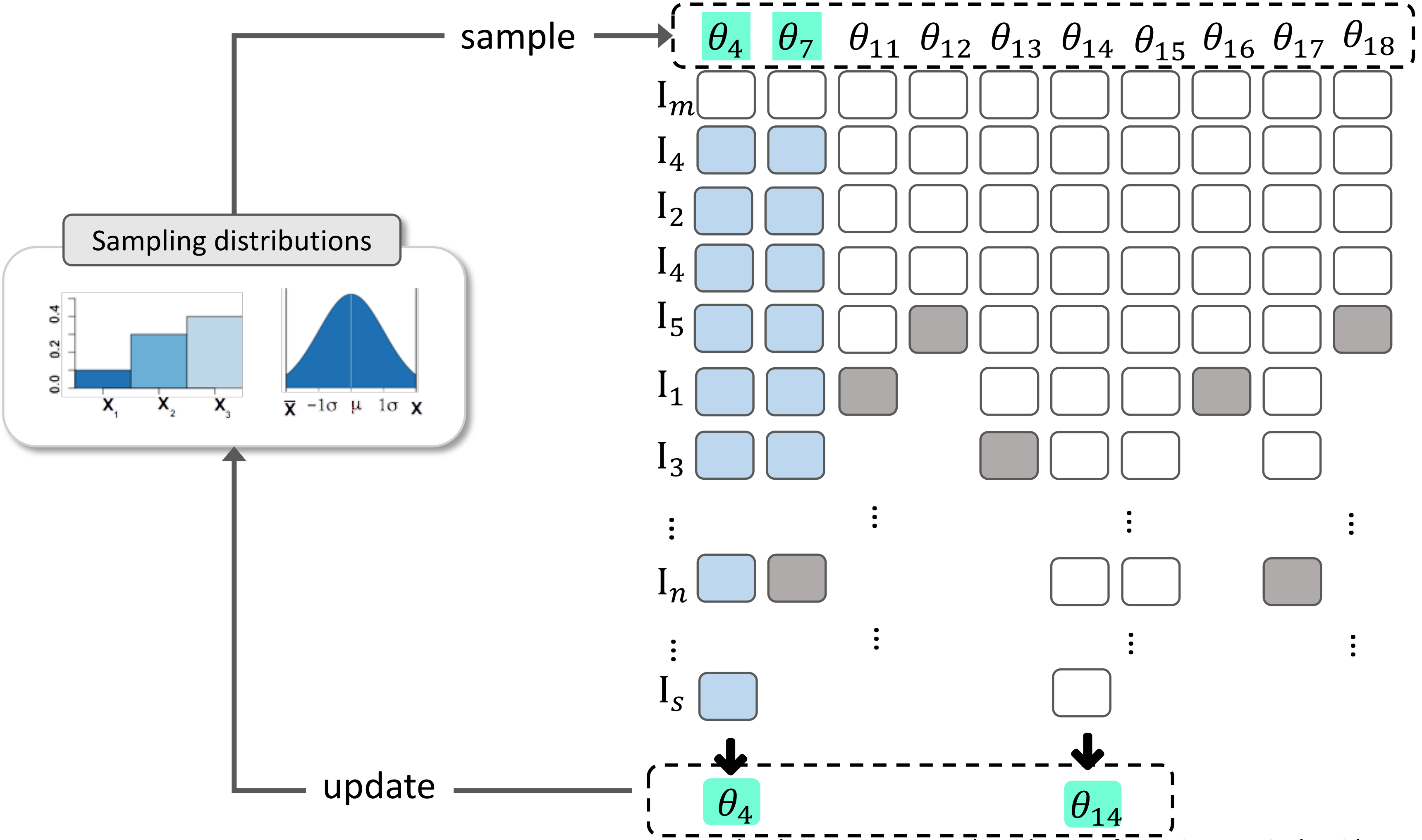
Iterated racing + Estimation of distribution

- State of the art configurator
- Implemented in R
  - Multiplatform
  - Flexible
- Parallel evaluation (MPI, multicores, grid engine)
- Several scenario options

The irace package



The irace package



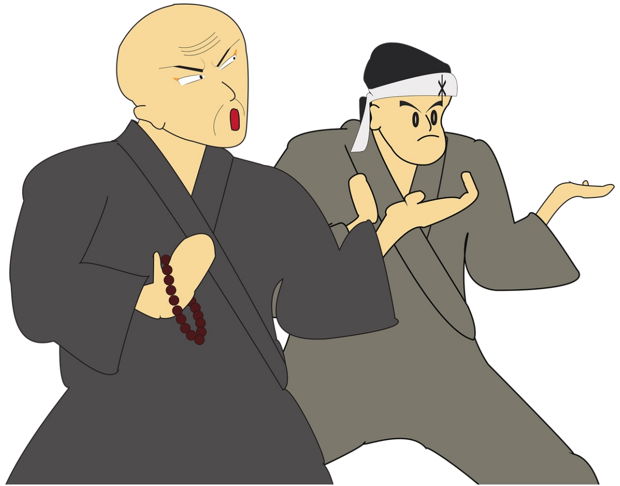
The irace package: some features

- Different parameters **types**
  - categorical, ordinal, real, integer, log real, log integer
- Configuration **repair**: fix configurations when generated
- Automatic **rejection**: remove undesirable configurations
- Adaptive **capping**: for runtime minimization objectives
- Deterministic / stochastic mode

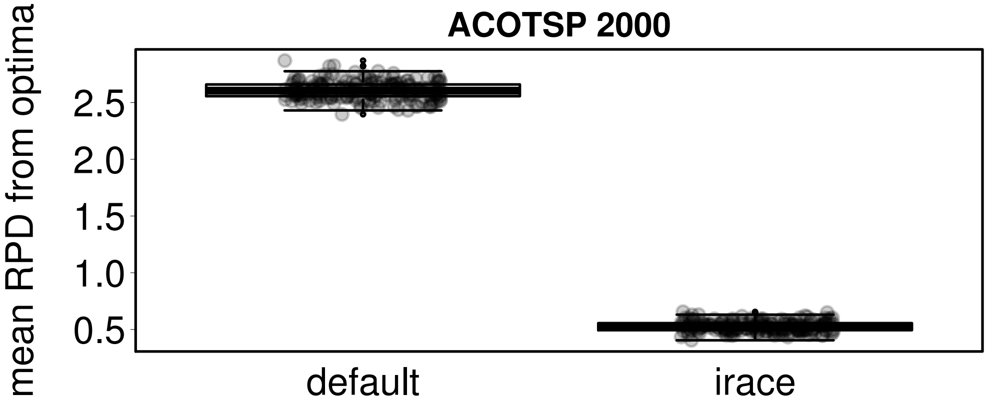
3

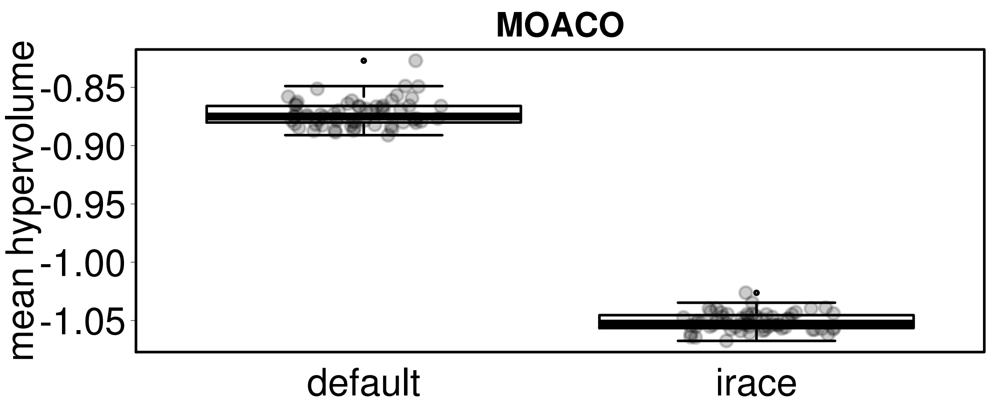
Configuration examples

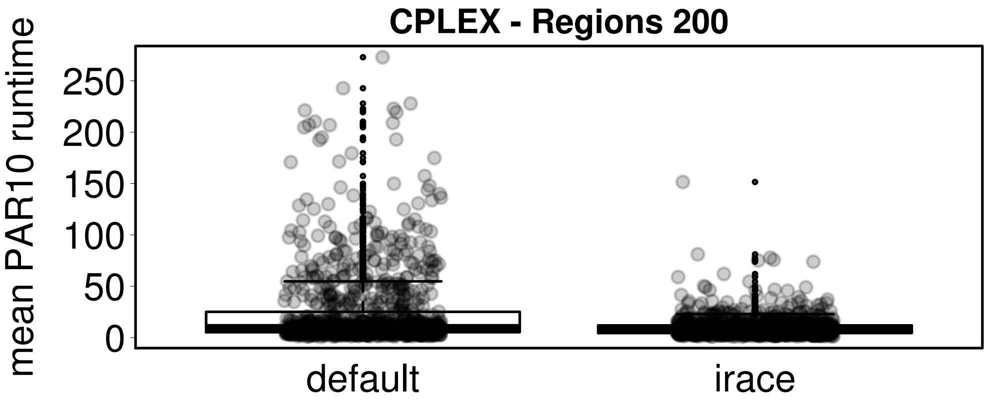
Yes, you can!



Default configuration tests







During the design process …

Irace can help to get know our algorithm

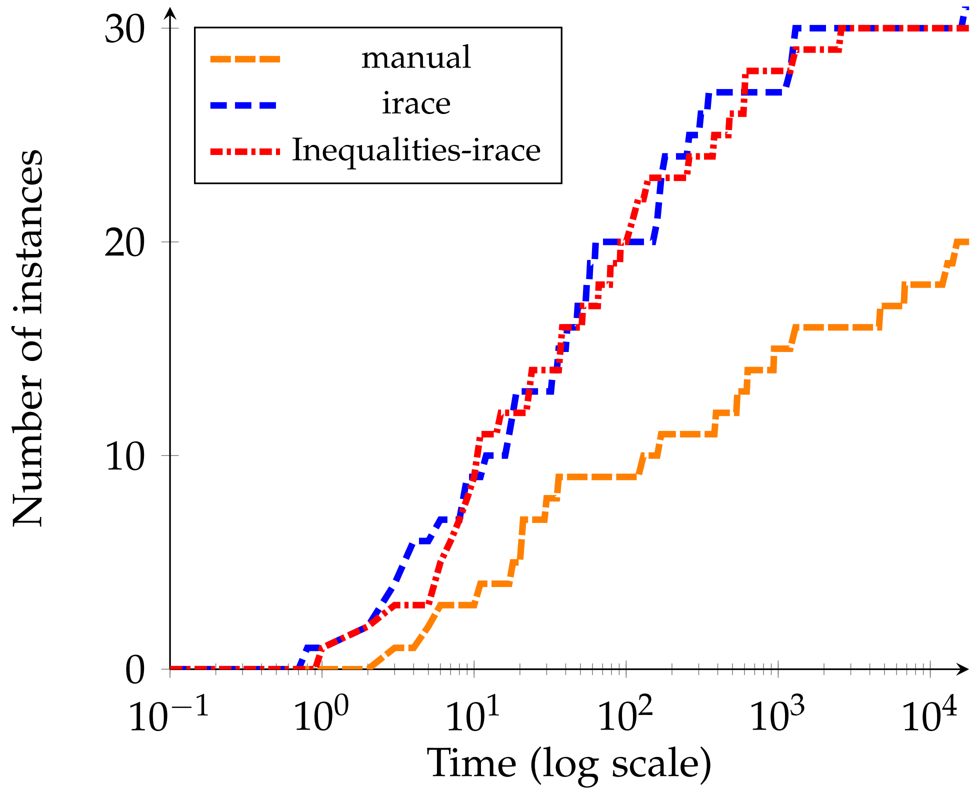
Example:

Column generation algorithm and branch-and-cut-and-price algorithm

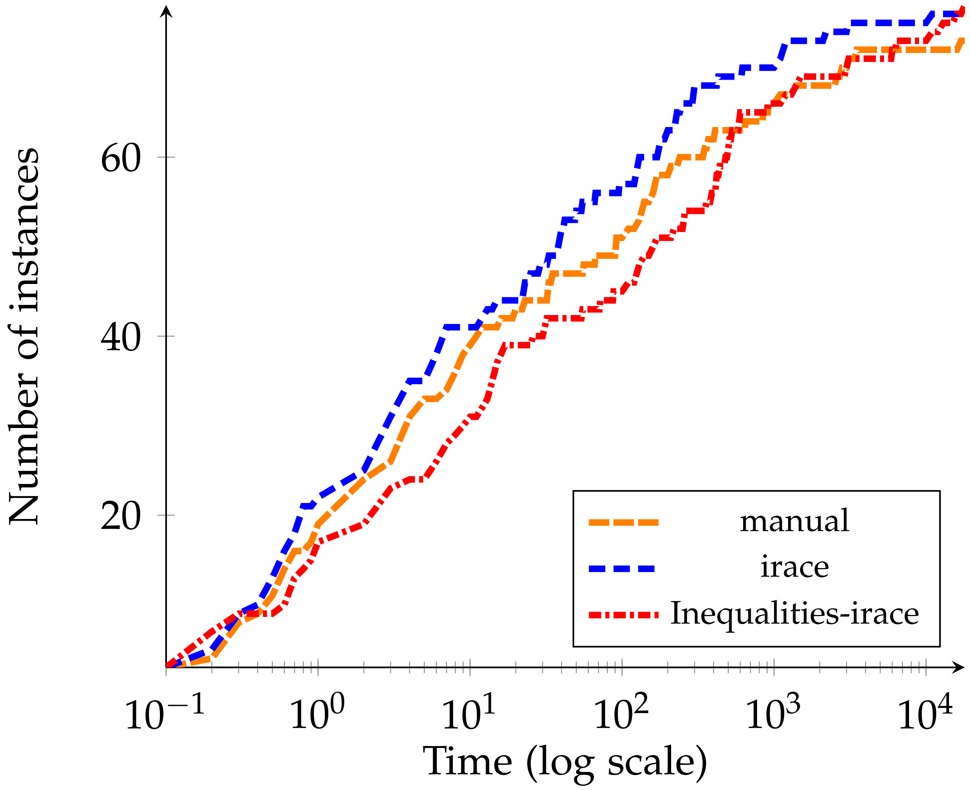
- Reduced the mean execution time of the column generation algorithm
- Improved performance of the branch-and-cut-and-price algorithm

During the design process …

Model #1



Model #2



Work performed with Alessia Violin for her Ph.D. Thesis.

When resources change …

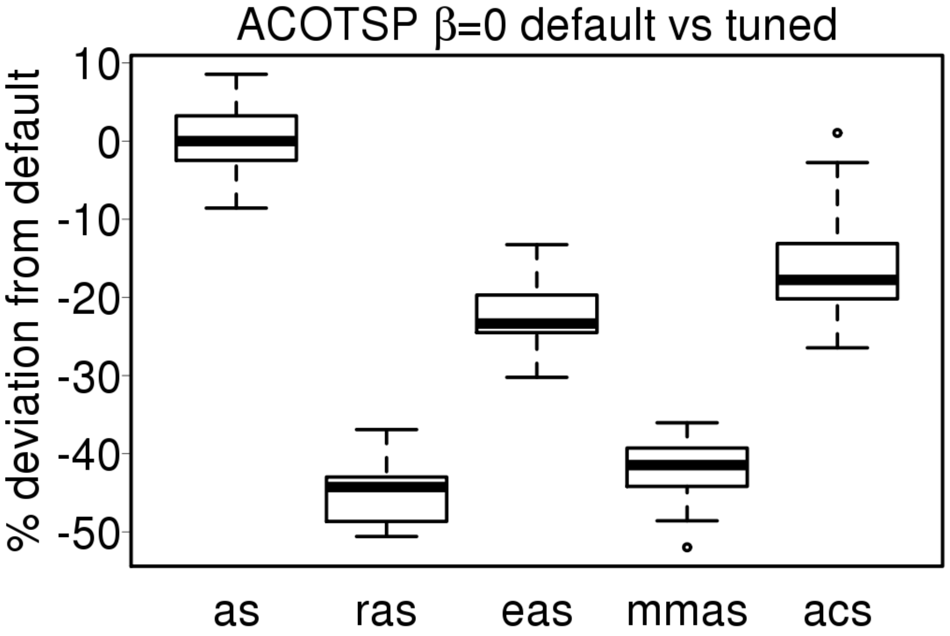
What happens with the performance of default settings?

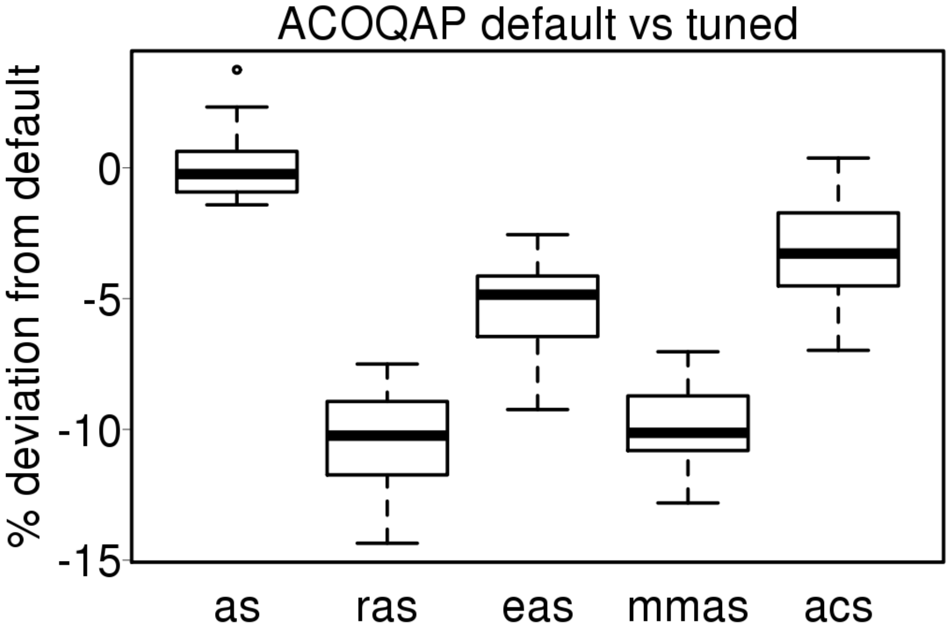
Algorithm: ACOTSP / ACOQAP

- Strongly restricted execution budget

Are the current ACO settings still adequate?

When resources change …





When the platform change …

What happens with the performance of "default settings"?

Algorithm: GCC

- Different platforms

Are GCC optimization settings preferrable?

When the platform changes …

We used irace to configure the **optimization options** of GCC

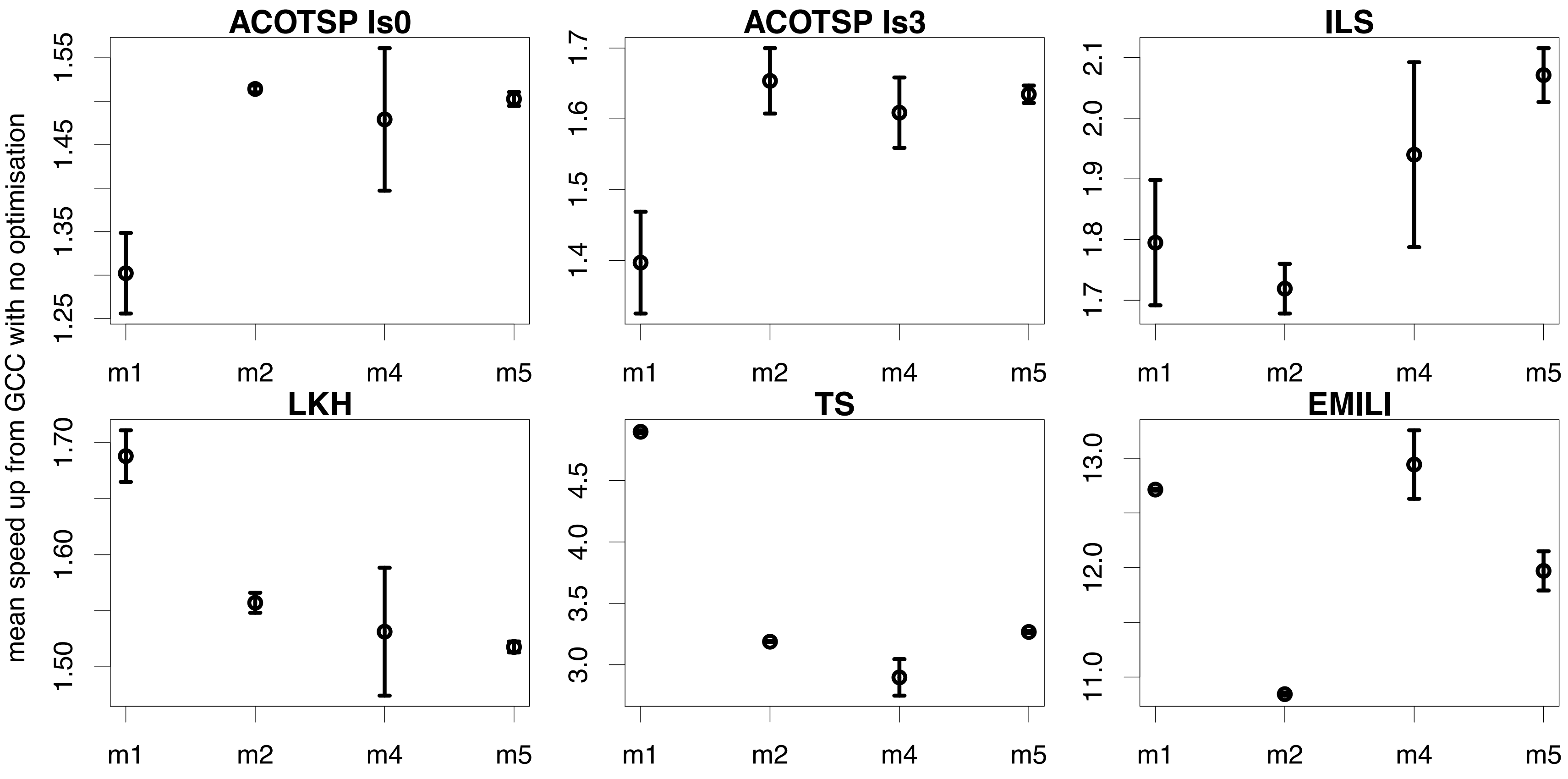
- Two parameter spaces
  - 172 categorical parameters
  - 367 mixed parameters

- Experiments: compiling **six optimization algorithms (C, C++)**

- In 4 different machines

- Compared to –O3 settings

When the platform changes …



Speed up from -O3
in
different machines

Take home message

- Parameter settings have often a large effect over performance
  - **Configure** when evaluating / comparing algorithms

- Any type of configuration is better than no configuration

- Configuration is useful not only for performance
  - But also for getting **knowledge** about an algorithm

- A configurator can provide assistance in the task of algorithm design

Take home message: perspectives

- Has the potential to enable **automated algorithm design**
  - Parameter tuning
  - Component selection
  - Algorithmic structure

Why?

1. Free designers of the tedious task of fully configuring and algorithm
2. Focus on the creation of new/better algorithmic components and structures

In the next episode of the survival guide …

We try irace!

Thanks for your attention !

Questions, comments?

References

[1] B. Adenso-Díaz and M. Laguna. Fine-tuning of algorithms using fractional experimental design and local search. *Operations Research*, 54(1):99–114, 2006.

[2] C. Audet and D. Orban. Finding optimal algorithmic parameters using derivative-free optimization. *SIAM Journal on Optimization*, 17(3):642–664, 2006.

[3] Z. Yuan, M. A. Montes de Oca, T. Stützle, and M. Birattari. Continuous optimization algorithms for tuning real and integer algorithm parameters of swarm intelligence algorithms. *Swarm Intelligence*, 6(1):49–75, 2012.

[4] J. J. Grefenstette. Optimization of control parameters for genetic algorithms. IEEE Transactions on Systems, Man, and Cybernetics, 16(1):122–128, 1986.

[5] V. Nannen and A. E. Eiben. Relevance estimation and value calibration of evolutionary algorithm parameters. In M. M. Veloso, editor, *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 975–980. AAAI Press, Menlo Park, CA, 2007.

[6] F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stützle. ParamILS: an automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, 36:267–306, October 2009.

[7] C. Ansótegui, M. Sellmann, and K. Tierney. A gender-based genetic algorithm for the automatic configuration of algorithms. In I. P. Gent, editor, *Principles and Practice of Constraint Programming, CP 2009*, volume 5732 of *LNCS*, pages 142–157. Springer, 2009.

References

[8] M. Oltean. Evolving evolutionary algorithms using linear genetic programming. Evolutionary Computation, 13(3):387–410, 2005.

doi: 10.1162/1063656054794815.

[9] T. Bartz-Beielstein, C. Lasarczyk, and M. Preuss. Sequential parameter optimization. In *IEEE CEC*, pages 773–780, Piscataway, NJ, September 2005. IEEE Press.

[10] T. Bartz-Beielstein, C. Lasarczyk, and M. Preuss. The sequential parameter optimization toolbox. In Bartz-Beielstein et al. (2010a), pages 337–360.

[11] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In C. A. Coello Coello, editor, *Learning and Intelligent Optimization, 5th International Conference, LION 5*, volume 6683 of *LNCS*, pages 507–523. Springer, 2011.

[12] C. Ansótegui, Y. Malitsky, H. Samulowitz, M. Sellmann, and K. Tierney. Model-based genetic algorithms for algorithm configuration. In Q. Yang and M. Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI-15)*, pages 733–739. IJCAI/AAAI Press, Menlo Park, CA, 2015.

[13] M. Birattari, T. Stützle, L. Paquete, and K. Varrentrapp. A racing algorithm for configuring metaheuristics. In W. B. Langdon et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2002*, pages 11–18. Morgan Kaufmann Publishers, San Francisco, CA, 2002.

[14] P. Balaprakash, M. Birattari, and T. Stützle. Improvement strategies for the F-race algorithm: Sampling design and iterative refinement. In T. Bartz- Beielstein, M. J. Blesa, C. Blum, B. Naujoks, A. Roli, G. Rudolph, and M. Sampels, editors, *Hybrid Metaheuristics*, volume 4771 of *LNCS*, pages 108–122. Springer, 2007.

[15] M. López-Ibáñez, J. Dubois-Lacoste, L. Pérez Cáceres, T. Stützle, and M. Birattari. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016a