Configuración Automática de Algoritmos

Manuel López-Ibáñez

manuel.lopez-ibanez@manchester.ac.uk
http://lopez-ibanez.eu

University of Manchester, UK

MANCHESTER 1824
The University of Manchester
Alliance Manchester Business School

Escuela de Invierno – redHEUR SEIO 22/11/2024

¿Quién soy?

- Full Professor (Catedrático) at AMBS (University of Manchester)

MANCHESTER 1824
The University of Manchester
Alliance Manchester Business School

- "Beatriz Galindo" Senior Distinguished Researcher, 2020–2022

UNIVERSIDAD DE MÁLAGA

- Editor-in-Chief of GECCO 2019
  - ☞ GECCO 2025 es en Málaga!

- (co-)Editor-in-Chief of ACM Transactions on Evolutionary Learning and Optimization (ACM TELO)

2021
Volume 1, Number 4

ACM Transactions on
Evolutionary Learning and Optimization

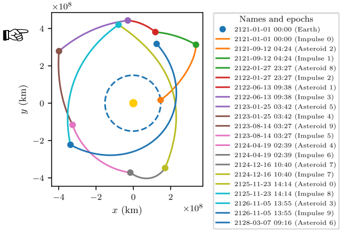
Article 13 (28 pages)	**D. Antipov** **B. Doerr**	Precise Runtime Analysis for Plateau Functions
Article 14 (21 pages)	**M. López-Ibáñez** **J. Branke** **L. Paquete**	Reproducibility in Evolutionary Computation
Article 15 (26 pages)	**F. Khosravi** **A. Rass** **J. Teich**	Efficient Computation of Probabilistic Dominance in Multi-objective Optimization
Article 16 (43 pages)	**B. Doerr** **F. Neumann**	A Survey on Recent Progress in the Theory of Evolutionary Algorithms for Discrete Optimization

ACM Transactions on Evolutionary Learning and Optimization Vol. 1 • No. 4 • 2021 Articles 13–16

Association for Computing Machinery
Advancing Computing as a Science & Profession

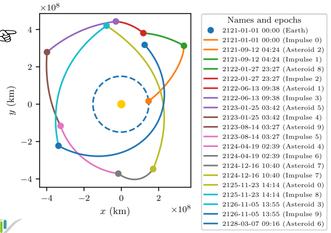
My research is about . . .

- Benchmarking and Empirical Analysis of Optimization Algorithms
  - ☞ Reproducibility in Evolutionary Computation [López-Ibáñez, Branke & Paquete, 2021]
- Multi-objective optimization ☞ EAF package ☞ moocore package
- Interactive optimization (human-in-the-loop)
  - ☞ Machine Decision Makers [López-Ibáñez & Knowles, 2015]
- Automatic configuration, selection and design of algorithms ☞ irace
- Expensive optimization . . . , Asteroid Routing Problem ☞



My research is about . . .

- Benchmarking and Empirical Analysis of Optimization Algorithms
  - ☞ Reproducibility in Evolutionary Computation [López-Ibáñez, Branke & Paquete, 2021]
- Multi-objective optimization ☞ EAF package ☞ moocore package
- Interactive optimization (human-in-the-loop)
  - ☞ Machine Decision Makers [López-Ibáñez & Knowles, 2015]
- Automatic configuration, selection and design of algorithms ☞ irace
- Expensive optimization . . . , Asteroid Routing Problem ☞



- Applications, applications, applications!
  - Optimization in steel manufacturing ArcelorMittal
  - School bus routing for SEND students 365 RESPONSE
  - Supply chain design for Personalised Medicine biopharm
  - Bayesian Optimisation with dynamic constraints IBM
  - Intervowen Optimisation slb

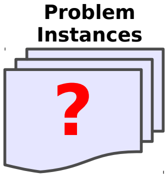
Mario's Pizza Delivery Problem

[Birattari, 2004]

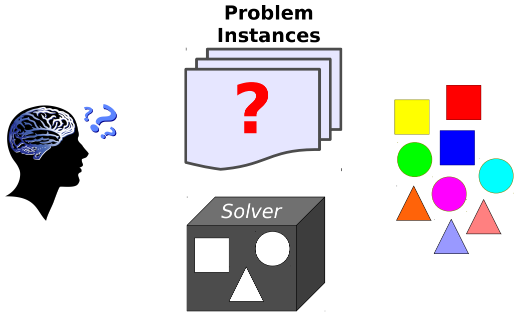
Mario collects phone orders for 30 minutes.
Mario wants to schedule deliveries to get back to the pizzeria as fast as possible.

- Scheduling deliveries is an *optimization problem*
- A different *problem instance* arises every 30 minutes
- Limited time for solving, say one minute (online)
- Limited time to implement an optimization algorithm, say one week (offline)

Traditional design of optimization algorithms

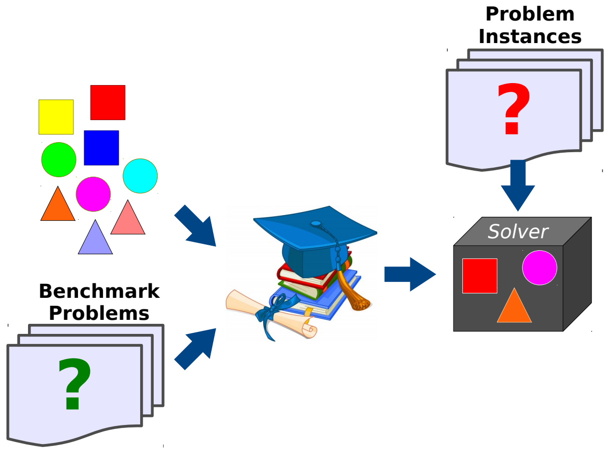


Traditional design of optimization algorithms



Traditional design of optimization algorithms

Human expert + intuition + trial-and-error/statistics



Design choices and parameters everywhere

Modern high-performance optimizers involve a large number of design choices and (hyper)-parameter settings

- Exact solvers
    - Design choices: alternative models, pre-processing, variable selection, value selection, branching rules ...
      $+$ numerical parameters
    - IBM CPLEX: 63 parameters that control the optimization

Design choices and parameters everywhere

Modern high-performance optimizers involve a large number of design choices and (hyper)-parameter settings

- Exact solvers
  - Design choices: alternative models, pre-processing, variable selection, value selection, branching rules . . .
    + numerical parameters
  - IBM CPLEX: 63 parameters that control the optimization
- (Meta)-heuristic solvers
  - Design choices: solution representation, operators, neighborhoods, pre-processing, strategies, . . . + numerical parameters
  - Many are *hidden*

Design choices and parameters everywhere

Modern high-performance ~~optimizers~~ software involve a large number of design choices and (hyper)-parameter settings

Domain	Software	Parameters	
ML	WEKA	768	[Kotthoff et al., 2016]
	Auto-sklearn	110	[Feurer et al., 2015]
Code optimization	GCC	172 flags + 195 numerical	[Pérez Cáceres et al., 2017b]
Databases	Cassandra	23	[Silva-Muñoz et al., 2021]

Design choices and parameters everywhere

Modern high-performance ~~optimizers~~ software involve a large number of design choices and (hyper)-parameter settings

Design choices and parameters everywhere

- *Categorical* parameters

  $$\text{localsearch} \in \{\ \text{tabu search, SA, ILS}\ \}$$

- *Ordinal* parameters

  $$\text{neighborhoods} \in \{\ \text{small, medium, large}\ \}$$

- *Numerical* parameters (integer and real-valued)

  population sizes, acceptance temperature, hidden constants, ...

- *Conditional* parameters are only active for specific values of other parameters:

  `temperature` only enabled if `localsearch == "SA"`

Design choices and parameters everywhere

- *Categorical* parameters

  localsearch $\in$ { tabu search, SA, ILS }

- *Ordinal* parameters

  neighborhoods $\in$ { small, medium, large }

- *Numerical* parameters (integer and real-valued)

  population sizes, acceptance temperature, hidden constants, ...

- *Conditional* parameters are only active for specific values of other parameters:

  `temperature` only enabled if `localsearch == "SA"`

Configuring an algorithm means
setting its categorical, ordinal and numerical parameters

Algorithm configuration and design is hard

Challenges

Algorithm configuration and design is hard

Challenges

- ✘ Many alternative design choices and parameter settings
- ✘ Nonlinear interactions among algorithm components and/or parameters
- ✘ Algorithms are stochastic
- ✘ Problem instances used for design (benchmark instances) are not identical to the ones found in the real-world
- ✘ Performance assessment is difficult (statistical analysis)

Traditional algorithm design and configuration

Traditional approaches

Traditional algorithm design and configuration

Traditional approaches

- Trial–and–error design guided by expertise/intuition
  - ✗ prone to over-generalizations,
  - ✗ limited exploration of design alternatives,
  - ✗ human biases
- Guided by theoretical studies
  - ✗ often based on over-simplifications,
  - ✗ specific assumptions,
  - ✗ few parameters

Traditional algorithm design and configuration

Traditional approaches

- Trial–and–error design guided by expertise/intuition
  - ✗ prone to over-generalizations,
  - ✗ limited exploration of design alternatives,
  - ✗ human biases
- Guided by theoretical studies
  - ✗ often based on over-simplifications,
  - ✗ specific assumptions,
  - ✗ few parameters

Can we make this approach more principled and automatic?

The algorithm configuration problem

1. Find the best algorithm configuration given a set of *training problem instances*
2. Repeatedly use this algorithm configuration to solve *unseen problem instances*

The algorithm configuration problem

1. Find the best algorithm configuration
   given a set of *training problem instances*
2. Repeatedly use this algorithm configuration to solve
   unseen problem instances

A problem with many names:

offline parameter *tuning*,
automatic algorithm configuration,
hyper-parameter optimization,
hyper-heuristics, genetic programming,
meta-optimisation, programming by optimisation [Hoos, 2012], . . .

Offline configuration vs. Online control

Offline tuning / Algorithm configuration

- Learn best configuration before *solving* the real problem instance
- Configuration done on training problem instances
- Performance measured over test ($\neq$ training) instances

Offline configuration vs. Online control

Offline tuning / Algorithm configuration

- Learn best configuration before *solving* the real problem instance
- Configuration done on training problem instances
- Performance measured over test ($\neq$ training) instances

Online tuning / Parameter control / Reactive search

- Learn best configuration *while* solving each instance
- No training phase but more expensive *while* solving
- Very popular in continuous optimization
- Ultimate goal: parameter-free algorithms

Offline configuration vs. Online control

Offline tuning / Algorithm configuration

- Learn best configuration before *solving* the real problem instance
- Configuration done on training problem instances
- Performance measured over test ($\neq$ training) instances

Online tuning / Parameter control / Reactive search

- Learn best configuration *while* solving each instance
- No training phase but more expensive *while* solving
- Very popular in continuous optimization
- Ultimate goal: parameter-free algorithms

All online methods have parameters that are configured offline

AC is a mixed-integer stochastic black-box optimization problem

Mixed-decision variables

- discrete (categorical, ordinal and integer) and real-valued
- conditional parameters, box-constraints and other constraints

Stochasticity

- of the target algorithm
- of the problem instances

Black-box

evaluation requires running a configuration on an instance

Typical tuning goals

- maximize solution quality within given time
- minimize run-time to decision / optimal solution

AC is a mixed-integer stochastic black-box optimization problem

Mixed-decision variables

- discrete (categorical, ordinal and integer) and real-valued
- conditional parameters, box-constraints and other constraints

Stochasticity

- of the target algorithm
- of the problem instances

Black-box

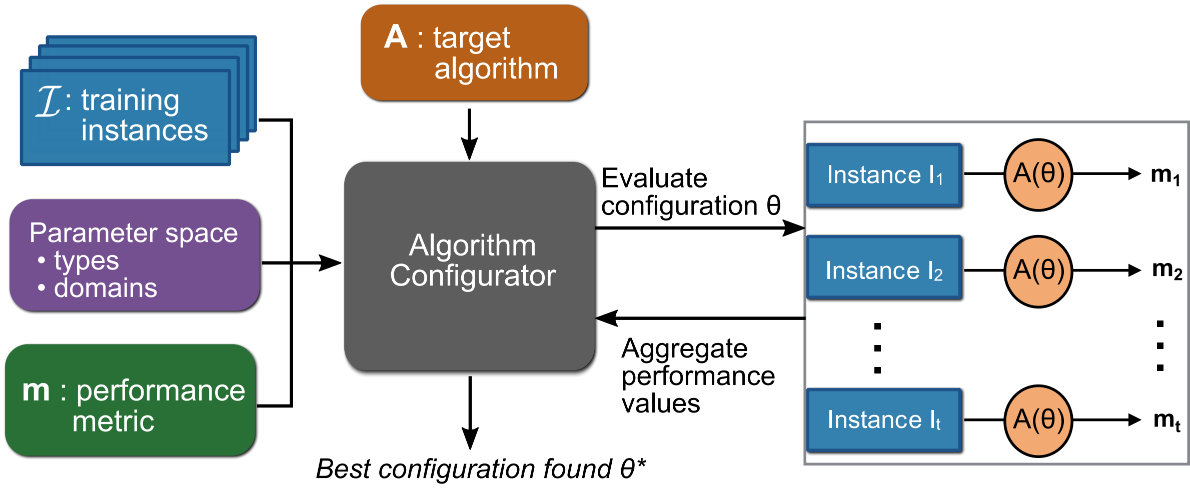
evaluation requires running a configuration on an instance

Typical tuning goals

- maximize solution quality within given time
- minimize run-time to decision / optimal solution

AC requires specialized methods !

(Offline) Automatic Algorithm Configuration



The racing approach [Maron & Moore, 1997; Birattari et al., 2002]

Racing is a method for the *selection of the best*
among a given set of algorithm configurations

- ✓ Reduce effort evaluating low performance configurations
- ✓ Focus effort on selecting the best configurations

The racing approach

[Maron & Moore, 1997; Birattari et al., 2002]

Θ_0
Instances

Racing is a method for the *selection of the best* among a given set of algorithm configurations

The racing approach

[Maron & Moore, 1997; Birattari et al., 2002]

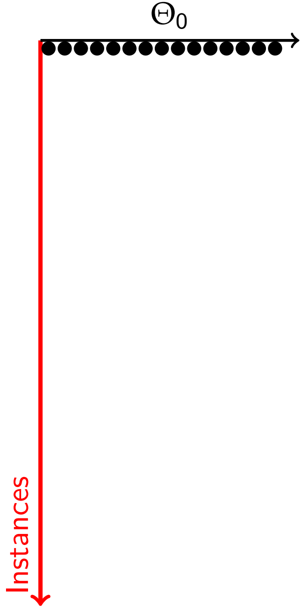
Θ_0
Instances

Racing is a method for the *selection of the best* among a given set of algorithm configurations

- start with a set of initial candidates

The racing approach

[Maron & Moore, 1997; Birattari et al., 2002]



Racing is a method for the *selection of the best* among a given set of algorithm configurations

- start with a set of initial candidates
- consider a *stream* of instances

The racing approach

[Maron & Moore, 1997; Birattari et al., 2002]

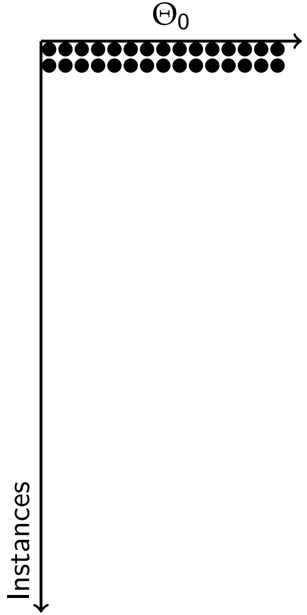
Θ_0
Instances

Racing is a method for the *selection of the best* among a given set of algorithm configurations

- start with a set of initial candidates
- consider a *stream* of instances
- sequentially evaluate candidates

The racing approach

[Maron & Moore, 1997; Birattari et al., 2002]



Racing is a method for the *selection of the best* among a given set of algorithm configurations

- start with a set of initial candidates
- consider a *stream* of instances
- sequentially evaluate candidates

The racing approach

[Maron & Moore, 1997; Birattari et al., 2002]

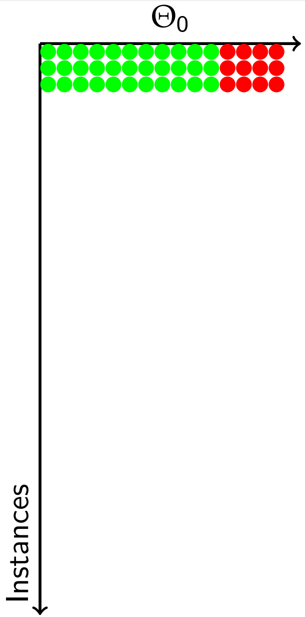
Θ_0
Instances

Racing is a method for the *selection of the best* among a given set of algorithm configurations

- start with a set of initial candidates
- consider a *stream* of instances
- sequentially evaluate candidates

The racing approach

[Maron & Moore, 1997; Birattari et al., 2002]



Racing is a method for the *selection of the best* among a given set of algorithm configurations

- start with a set of initial candidates
- consider a *stream* of instances
- sequentially evaluate candidates
- discard inferior candidates

  as sufficient evidence is gathered against them

The racing approach

[Maron & Moore, 1997; Birattari et al., 2002]

Θ_0
Instances

Racing is a method for the *selection of the best* among a given set of algorithm configurations

- start with a set of initial candidates
- consider a *stream* of instances
- sequentially evaluate candidates
- discard inferior candidates

  as sufficient evidence is gathered against them

The racing approach

[Maron & Moore, 1997; Birattari et al., 2002]

Θ_0
Instances

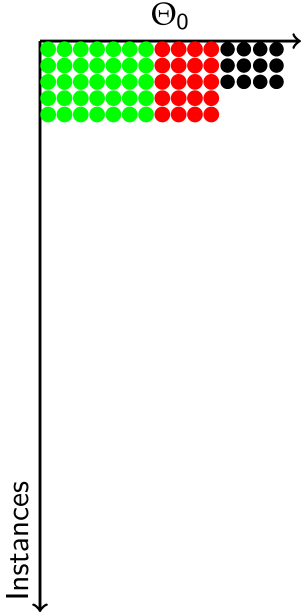
Racing is a method for the *selection of the best* among a given set of algorithm configurations

- start with a set of initial candidates
- consider a *stream* of instances
- sequentially evaluate candidates
- discard inferior candidates

  as sufficient evidence is gathered against them

The racing approach

[Maron & Moore, 1997; Birattari et al., 2002]



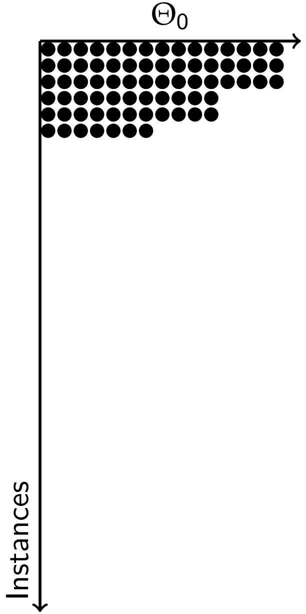
Racing is a method for the *selection of the best* among a given set of algorithm configurations

- start with a set of initial candidates
- consider a *stream* of instances
- sequentially evaluate candidates
- discard inferior candidates

  as sufficient evidence is gathered against them

The racing approach

[Maron & Moore, 1997; Birattari et al., 2002]



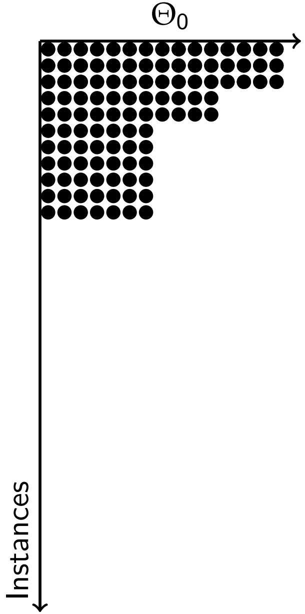
Racing is a method for the *selection of the best* among a given set of algorithm configurations

- start with a set of initial candidates
- consider a *stream* of instances
- sequentially evaluate candidates
- discard inferior candidates

  as sufficient evidence is gathered against them

The racing approach

[Maron & Moore, 1997; Birattari et al., 2002]

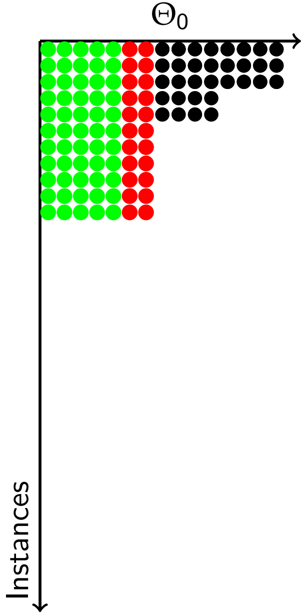


Racing is a method for the *selection of the best* among a given set of algorithm configurations

- start with a set of initial candidates
- consider a *stream* of instances
- sequentially evaluate candidates
- discard inferior candidates
  as sufficient evidence is gathered against them

The racing approach

[Maron & Moore, 1997; Birattari et al., 2002]



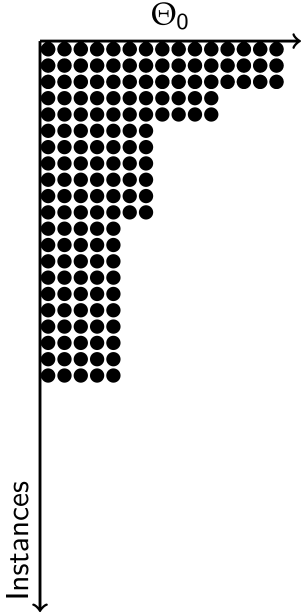
Racing is a method for the *selection of the best* among a given set of algorithm configurations

- start with a set of initial candidates
- consider a *stream* of instances
- sequentially evaluate candidates
- discard inferior candidates

  as sufficient evidence is gathered against them

The racing approach

[Maron & Moore, 1997; Birattari et al., 2002]



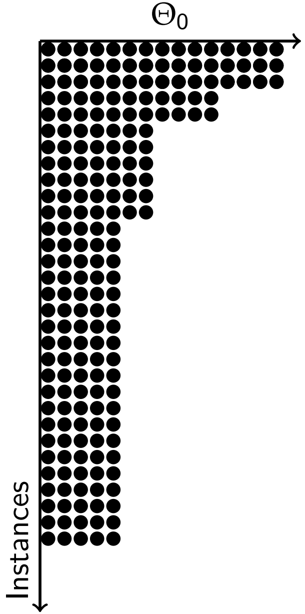
Racing is a method for the *selection of the best* among a given set of algorithm configurations

- start with a set of initial candidates
- consider a *stream* of instances
- sequentially evaluate candidates
- discard inferior candidates

  as sufficient evidence is gathered against them

The racing approach

[Maron & Moore, 1997; Birattari et al., 2002]



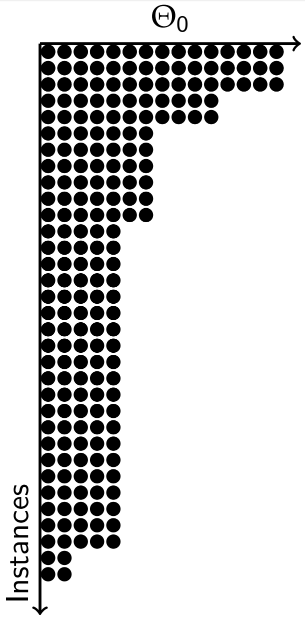
Racing is a method for the *selection of the best* among a given set of algorithm configurations

- start with a set of initial candidates
- consider a *stream* of instances
- sequentially evaluate candidates
- discard inferior candidates

  as sufficient evidence is gathered against them

The racing approach

[Maron & Moore, 1997; Birattari et al., 2002]



Racing is a method for the *selection of the best* among a given set of algorithm configurations

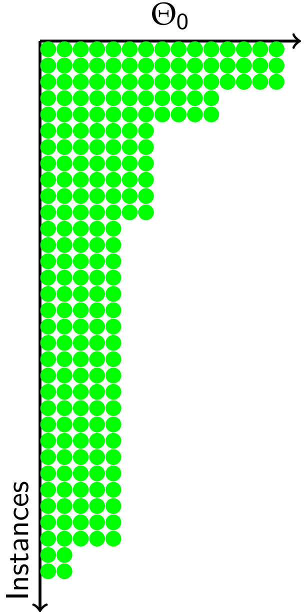
- start with a set of initial candidates
- consider a *stream* of instances
- sequentially evaluate candidates
- discard inferior candidates

  as sufficient evidence is gathered against them
- . . . repeat until a winner is selected

  or until computation time expires

The racing approach

[Maron & Moore, 1997; Birattari et al., 2002]



Racing is a method for the *selection of the best* among a given set of algorithm configurations

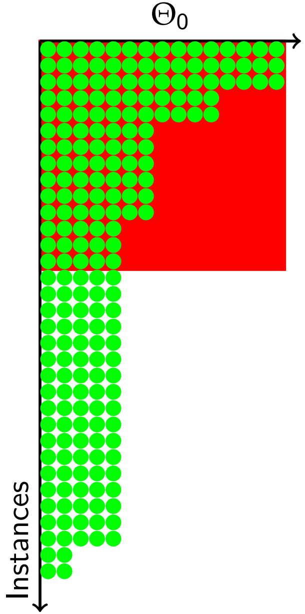
- start with a set of initial candidates
- consider a *stream* of instances
- sequentially evaluate candidates
- discard inferior candidates

  as sufficient evidence is gathered against them
- . . . repeat until a winner is selected

  or until computation time expires

The racing approach

[Maron & Moore, 1997; Birattari et al., 2002]



Racing is a method for the *selection of the best* among a given set of algorithm configurations

- start with a set of initial candidates
- consider a *stream* of instances
- sequentially evaluate candidates
- discard inferior candidates

  as sufficient evidence is gathered against them
- . . . repeat until a winner is selected

  or until computation time expires

The racing approach

[Maron & Moore, 1997; Birattari et al., 2002]

How to discard?

The racing approach

[Maron & Moore, 1997; Birattari et al., 2002]

How to discard?

Statistical tests!

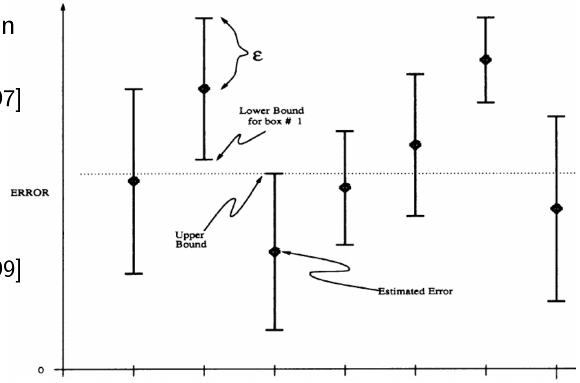
- Paired t-test with/*without* p-value correction (against the best)

  [Maron & Moore, 1997]

- *F-Race:* Friedman two-way analysis of variance by ranks
  + Friedman post-hoc test

  [Conover, 1999]

- *Bayesian:* Bayesian nonparametric statistics [Benavoli et al., 2015]



Taken from Maron & Moore [1997]

Sampling configurations

Racing (F-race, t-race, . . .) is a method for the *selection of the best* among a given set of algorithm configurations

Sampling configurations

Racing (F-race, t-race, . . .) is a method for the *selection of the best* among a given set of algorithm configurations

How to define this set of configurations?

Sampling configurations

Racing (F-race, t-race, . . .) is a method for the *selection of the best* among a given set of algorithm configurations

How to define this set of configurations?

- Full factorial
- Random sampling
- Iterative update of a probabilistic sampling model

  ($\approx$ Estimation of Distribution Algorithm)

$\Rightarrow$ *Iterated F-Race (I/F-Race)* [Balaprakash et al., 2007]

Iterated Racing

1. **Sampling** new configurations according to a probability distribution
2. **Selecting** the best configurations from the newly sampled ones by means of racing
3. **Updating** the probability distribution in order to bias the sampling towards the best configurations

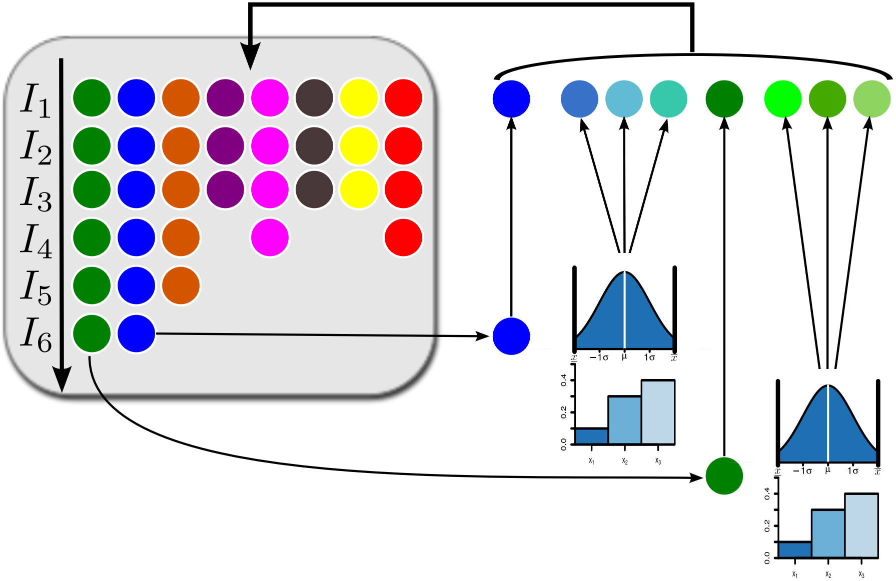
I/F-race: Balaprakash, Birattari & Stützle [2007], Birattari, Yuan, Balaprakash & Stützle [2010]

irace (v1): López-Ibáñez, Dubois-Lacoste, Stützle & Birattari [2011]

elitist irace (v2): López-Ibáñez, Dubois-Lacoste, Pérez Cáceres, Stützle & Birattari [2016]

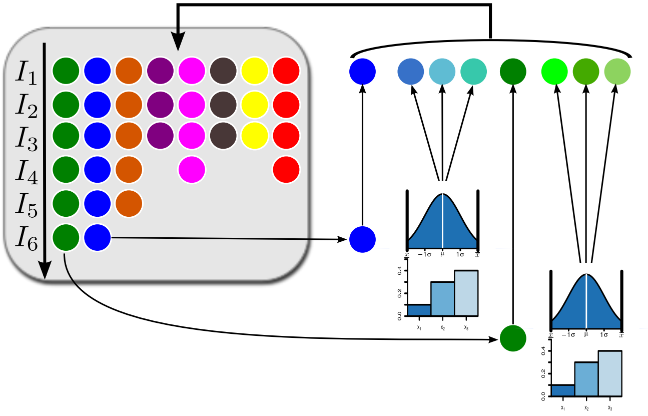
elitist irace + adaptive capping (v3): Pérez Cáceres, López-Ibáñez, Hoos & Stützle [2017a]

Iterated Racing



Elitist Iterated Racing

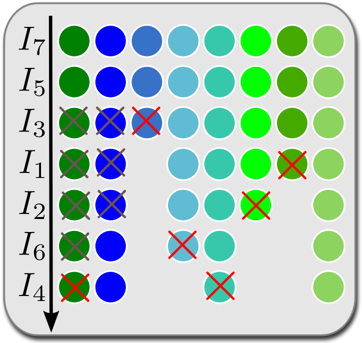
✘ Each new iteration (race) forgets the results of the previous one
$\Rightarrow$ Iterated F-race may "*lose*" the best-so-far configuration



Elitist Iterated Racing

✘ Each new iteration (race) forgets the results of the previous one
⇒ Iterated F-race may "*lose*" the best-so-far configuration

✔ Protect the best configurations (*elites*) from being discarded unless all their results are considered

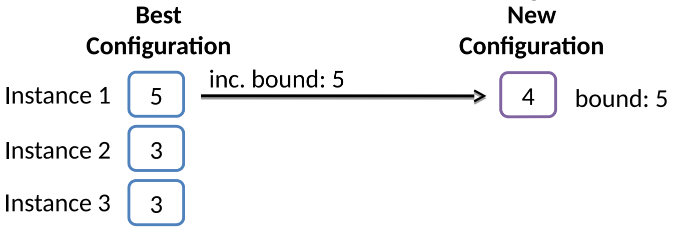


Adaptive capping (irace 3.0) [Pérez Cáceres, López-Ibáñez, Hoos & Stützle, 2017a]

- Extension of irace to better handle run-time minimization
- Configuration θ_1 evaluated on I_1 *dominates* θ_2 evaluated on I_2 if

$$I_2 \subset I_1 \quad \textit{and} \quad \sum_{i \in I_2} m(\theta_1, i) \leq \sum_{i \in I_2} m(\theta_2, i)$$

- Adaptive bound: $\kappa_i^{\text{new}} = \sum_{k=1}^{i} m(\theta_{\text{best}}, k) - \sum_{k=1}^{i-1} m(\theta_{\text{new}}, k)$
- Dominance elimination and adaptive capping:

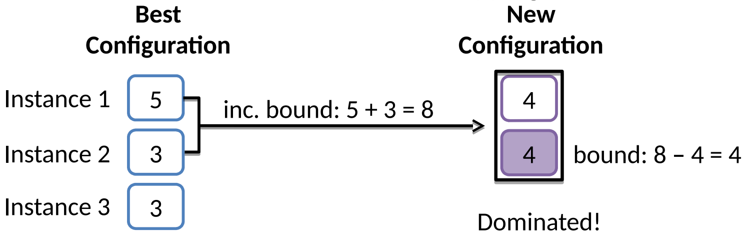


Adaptive capping (irace 3.0) [Pérez Cáceres, López-Ibáñez, Hoos & Stützle, 2017a]

- Extension of irace to better handle run-time minimization
- Configuration θ_1 evaluated on I_1 *dominates* θ_2 evaluated on I_2 if

$$I_2 \subset I_1 \quad \textit{and} \quad \sum_{i \in I_2} m(\theta_1, i) \leq \sum_{i \in I_2} m(\theta_2, i)$$

- Adaptive bound: $\kappa_i^{\text{new}} = \sum_{k=1}^{i} m(\theta_{\text{best}}, k) - \sum_{k=1}^{i-1} m(\theta_{\text{new}}, k)$
- Dominance elimination and adaptive capping:

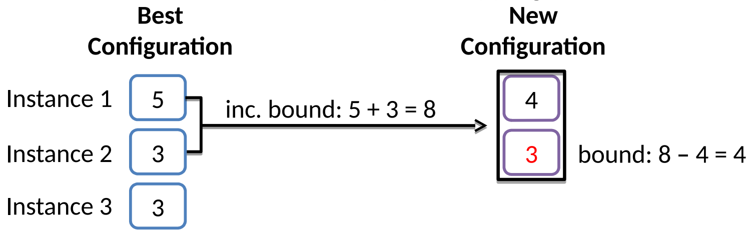


Adaptive capping (irace 3.0) [Pérez Cáceres, López-Ibáñez, Hoos & Stützle, 2017a]

- Extension of irace to better handle run-time minimization
- Configuration θ_1 evaluated on I_1 *dominates* θ_2 evaluated on I_2 if

$$I_2 \subset I_1 \quad \textit{and} \quad \sum_{i \in I_2} m(\theta_1, i) \leq \sum_{i \in I_2} m(\theta_2, i)$$

- Adaptive bound: $\kappa_i^{\text{new}} = \sum_{k=1}^{i} m(\theta_{\text{best}}, k) - \sum_{k=1}^{i-1} m(\theta_{\text{new}}, k)$
- Dominance elimination and adaptive capping:

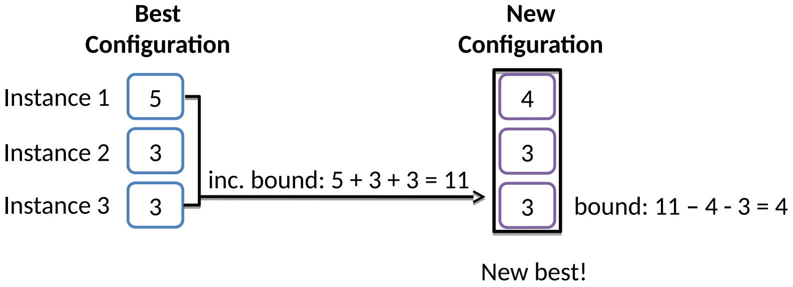


Adaptive capping (irace 3.0) [Pérez Cáceres, López-Ibáñez, Hoos & Stützle, 2017a]

- Extension of irace to better handle run-time minimization
- Configuration θ_1 evaluated on I_1 *dominates* θ_2 evaluated on I_2 if

$$I_2 \subset I_1 \quad \textit{and} \quad \sum_{i \in I_2} m(\theta_1, i) \leq \sum_{i \in I_2} m(\theta_2, i)$$

- Adaptive bound: $\kappa_i^{\text{new}} = \sum_{k=1}^{i} m(\theta_{\text{best}}, k) - \sum_{k=1}^{i-1} m(\theta_{\text{new}}, k)$
- Dominance elimination and adaptive capping:



An overview of applications of irace

- Parameter tuning
  - Exact MIP solvers (CPLEX, SCIP [López-Ibáñez & Stützle, 2014])
  - single-objective optimization metaheuristics
  - multi-objective optimization metaheuristics [López-Ibáñez & Stützle, 2012; Bezerra et al., 2016]
  - semi-interactive tuning [Diaz & López-Ibáñez, 2021]
  - anytime optimization (improve time-quality trade-offs) [López-Ibáñez & Stützle, 2014]
  - command-line flags of GCC compiler [Pérez Cáceres et al., 2017b]
- Automatic algorithm design
  - From a design grammar [Mascia et al., 2014; Martín-Santamaría et al., 2024]
- Machine learning [Lang et al., 2014; Miranda et al., 2014]
  - **mlr** and **mlr3tuning** R packages [Bischl et al., 2013, 2016]
- Design of control software for robots [Francesca et al., 2015]
- Theoretical research [Friedrich et al., 2018; Dang & Doerr, 2019; Hall et al., 2019]

2 098 citations in Google Scholar, 201 000 downloads

The irace Package

Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Thomas Stützle, and Mauro Birattari.
The irace package: Iterated Racing for Automatic Algorithm Configuration.
Operations Research Perspectives, 3:43–58, 2016. doi: 10.1016/j.orp.2016.09.002
https://mlopez-ibanez.github.io/irace/

The irace Package

Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Thomas Stützle, and Mauro Birattari.
The irace package: Iterated Racing for Automatic Algorithm Configuration.
Operations Research Perspectives, 3:43–58, 2016. doi: 10.1016/j.orp.2016.09.002
https://mlopez-ibanez.github.io/irace/

- Implementation of Iterated Racing in R

  Goal 1: Flexible

  Goal 2: Easy to use

The irace Package

Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Thomas Stützle, and Mauro Birattari.
The irace package: Iterated Racing for Automatic Algorithm Configuration.
Operations Research Perspectives, 3:43–58, 2016. doi: 10.1016/j.orp.2016.09.002
https://mlopez-ibanez.github.io/irace/

- Implementation of Iterated Racing in R

  Goal 1: Flexible

  Goal 2: Easy to use

- R package available at CRAN (GNU/Linux, Windows, OSX)

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

The irace Package

Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Thomas Stützle, and Mauro Birattari.
The irace package: Iterated Racing for Automatic Algorithm Configuration.
Operations Research Perspectives, 3:43–58, 2016. doi: 10.1016/j.orp.2016.09.002
https://mlopez-ibanez.github.io/irace/

- Implementation of Iterated Racing in R

  Goal 1: Flexible

  Goal 2: Easy to use

- R package available at CRAN (GNU/Linux, Windows, OSX)

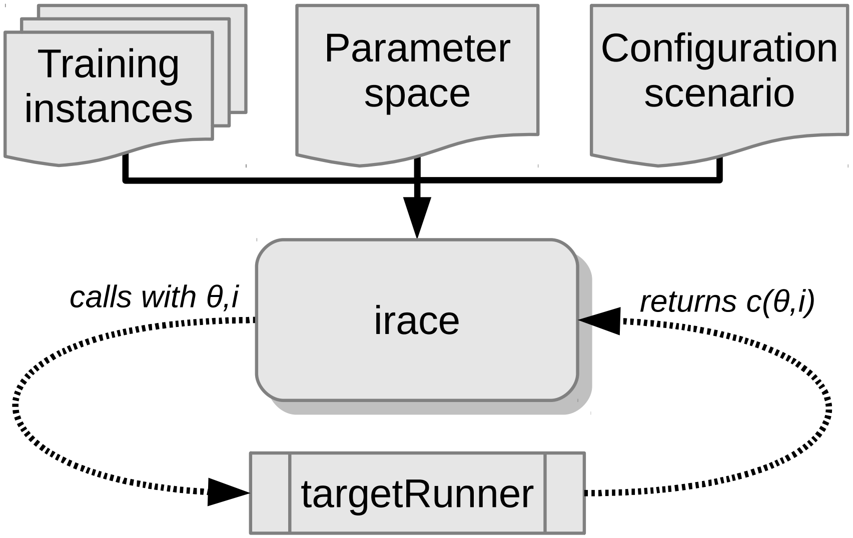
  http://cran.r-project.org/package=irace

- Use it through the command-line: (see `irace --help`)

```
irace --max-experiments 1000 --param-file parameters.txt
```

✔ No knowledge of R needed

The irace Package



The irace Package: Instances

- TSP instances

```
$ dir Instances/
3000-01.tsp 3000-02.tsp 3000-03.tsp ...
```

- Continuous functions

```
$ cat instances.txt
function=1 dimension=100
function=2 dimension=100
...
```

- Parameters for an instance generator

```
$ cat instances.txt
I1 --size 100 --num-clusters 10 --sym yes --seed 1
I2 --size 100 --num-clusters 5 --sym no --seed 1
...
```

- Script / R function that generates instances

  ☞ if you need this, tell us!

The irace Package: Parameter space

- Categorical (`c`), ordinal (`o`), integer (`i`) and real (`r`)
- Subordinate parameters (`| condition`)
- Logarithmic scale (`,log`)   *(irace 3.0)*

```
$ cat parameters.txt
```

```
# Name        Label/switch       Type     Domain            Condition
LS            "--localsearch "     c      (SA, TS, II)
rate          "--rate="            o      (low, med, high)
population    "--pop "           i,log    (1, 100)
temp          "--temp "            r      (0.5, 1)          | LS == "SA"
```

- For real parameters, number of decimal places is controlled by option *digits* (`--digits`)

The irace Package: Options

- *maxExperiments* (*maxTime*): maximum number of runs (or overall time) of the target algorithm (tuning budget)
- *testType*: either F-test or t-test

The irace Package: target-runner

- A script/program that calls the software to be tuned:

```
./target-runner configID instanceID seed instance configuration
```

e.g. :

```
./target-runner 2 1 1234567 3000-01.tsp --localsearch SA ...
```

- An R function

Flexibility: If there is something you cannot tune, let us know!

The irace Package: Other features

1. Initial configurations (e.g., default configuration)
2. Parallel evaluation:
   multiple CPUs, MPI, batch job clusters (SGE, PBS, Torque, Slurm)
3. Forbidden configurations (+ rejection):
   ```
   popsize < 5 & LS == "SA"
   ```
4. Recovery file: allows resuming an interrupted irace run
5. Test instances
6. Repair configurations before being evaluated
7. Adaptive capping (for runtime minimization)

The irace Package

Last version 3.5 (23/10/2022) ☞ 4.0 very soon !

A detailed user-guide / tutorial:

https://cran.r-project.org/web/packages/irace/vignettes/irace-package.pdf

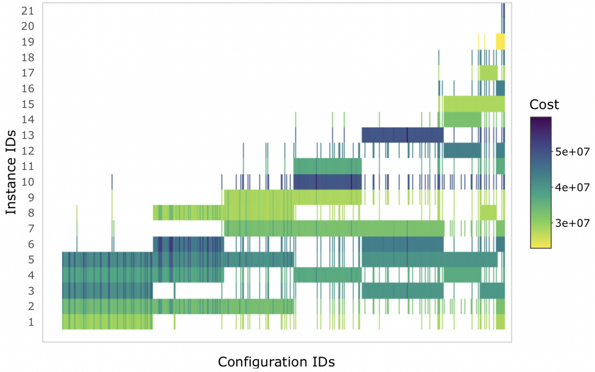
GitHub: https://github.com/MLopez-Ibanez/irace

Google group

https://groups.google.com/d/forum/irace-package

iraceplot: Opening the black-box

https://auto-optimization.github.io/iraceplot/



- Interactive HTML post-configuration report
- Summary statistics per instance / per configuration / per iteration
- Interactive visualizations
- Ablation report

Example #1

Automatically Improving the Anytime Behavior of Optimization Algorithms with irace

Manuel López-Ibáñez and Thomas Stützle.
Automatically improving the anytime behaviour of optimisation algorithms.
European Journal of Operational Research, 2014. doi: 10.1016/j.ejor.2013.10.043.

Automatically Improving the Anytime Behavior

Anytime Algorithm [Dean & Boddy, 1988]

- May be interrupted at any moment and returns a solution
- Keeps improving its solution until interrupted
- Eventually finds the optimal solution

Automatically Improving the Anytime Behavior

Anytime Algorithm [Dean & Boddy, 1988]

- May be interrupted at any moment and returns a solution
- Keeps improving its solution until interrupted
- Eventually finds the optimal solution

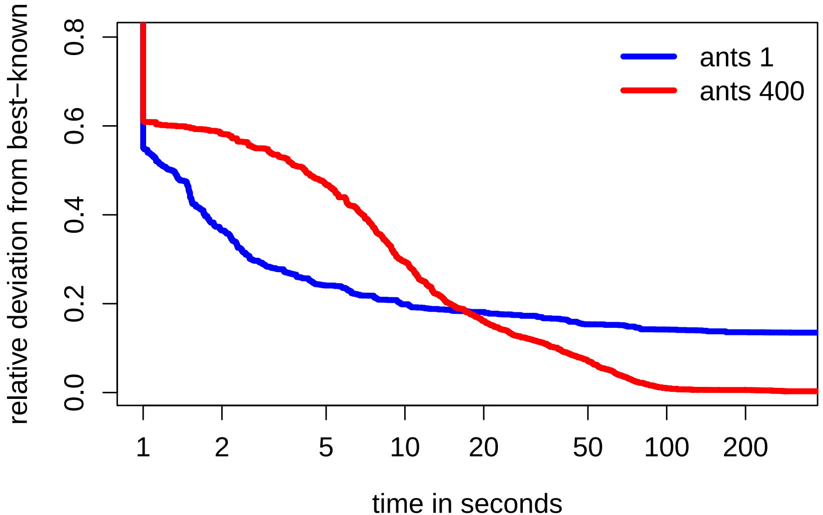
Good Anytime Behavior [Zilberstein, 1996]

Algorithms with good *"anytime" behavior* produce as high quality result as possible at any moment of their execution.

Automatically Improving the Anytime Behavior

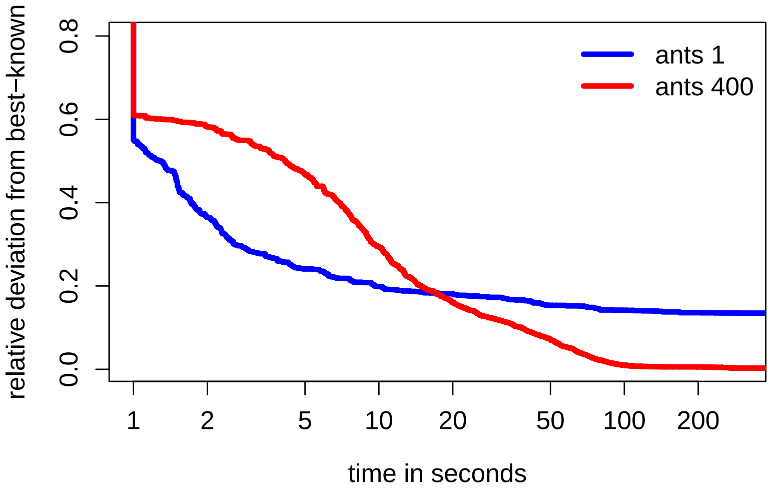
Max-Min Ant System w/o LS

Solution-quality vs. time (SQT) curve / Performance profile



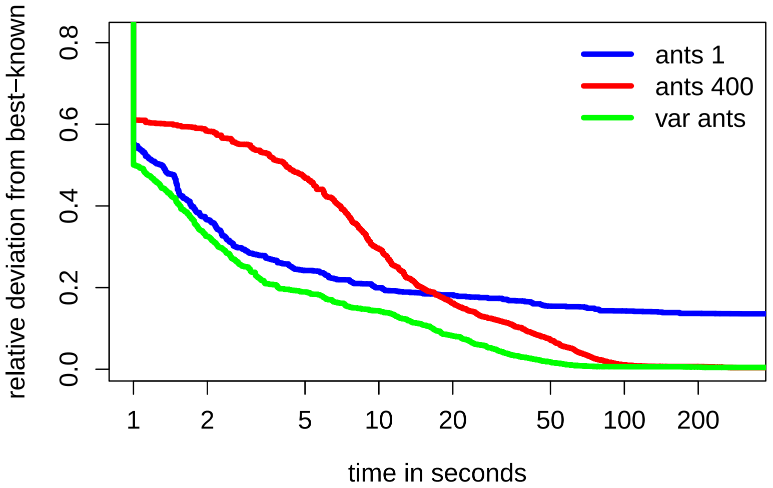
Automatically Improving the Anytime Behavior

Algorithms with good *“anytime” behaviour* produce as high quality result as possible at any moment of their execution [Zilberstein, 1996]



Automatically Improving the Anytime Behavior

Algorithms with good *"anytime" behaviour* produce as high quality result as possible at any moment of their execution [Zilberstein, 1996]



Improving Anytime Behaviour

How to improve the anytime behaviour of MMAS?

☞ Online parameter variation:

- Start with 1 ant, add 1 ant every iteration until 400 ants
- Start with $\beta = 10$, switch to $\beta = 2$ after 100 iterations
- ...

Improving Anytime Behaviour

How to improve the anytime behaviour of MMAS?

☞ Online parameter variation:

- Start with 1 ant, add 1 ant every iteration until 400 ants
- Start with $\beta = 10$, switch to $\beta = 2$ after 100 iterations
- ...

✗ More parameters!

✗ How to compare SQT curves?

Automatically Improving the Anytime Behavior

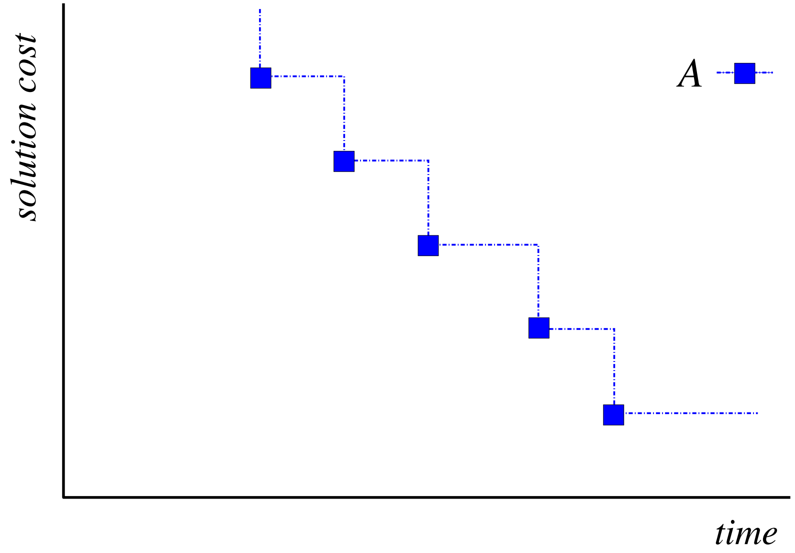
Online parameter control

- ✗ Which parameters to adapt? How? $\Rightarrow$ More parameters!
- ✓ Use irace (offline) to select the best parameter control strategies

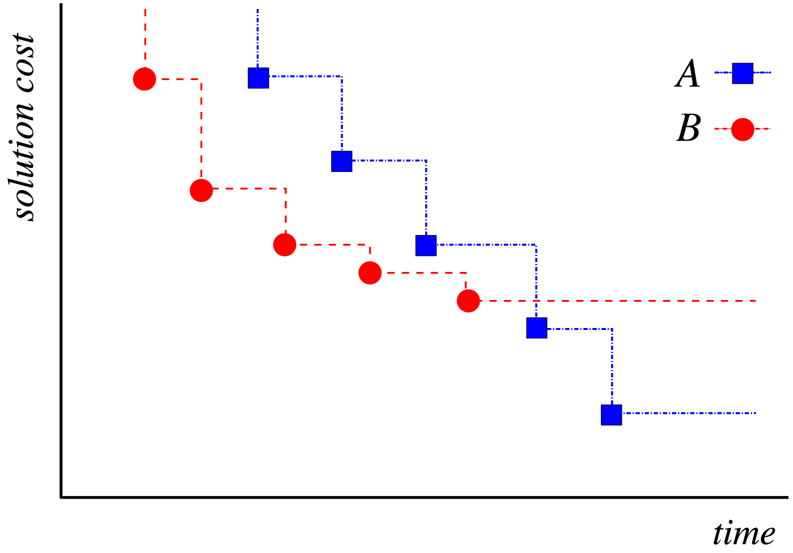
Improve Anytime Behavior

- ✓ More robust to different termination criteria
- ✗ How can irace compare SQT curves?

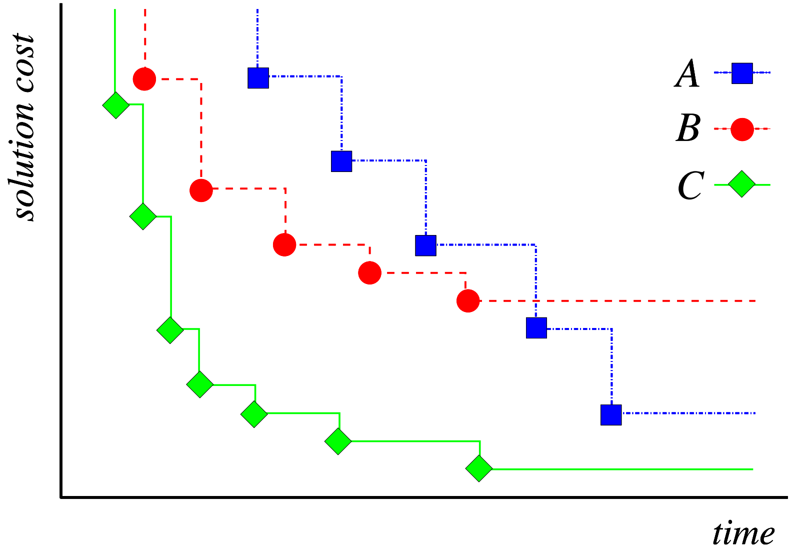
Automatically Improving the Anytime Behavior



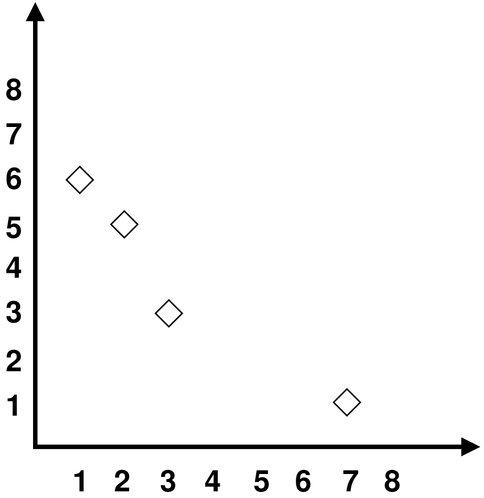
Automatically Improving the Anytime Behavior



Automatically Improving the Anytime Behavior

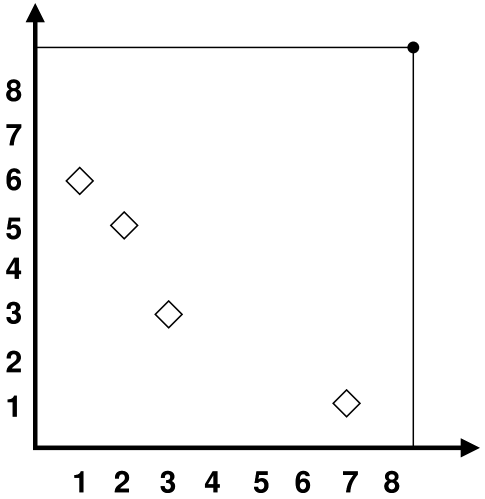


Automatically Improving the Anytime Behavior



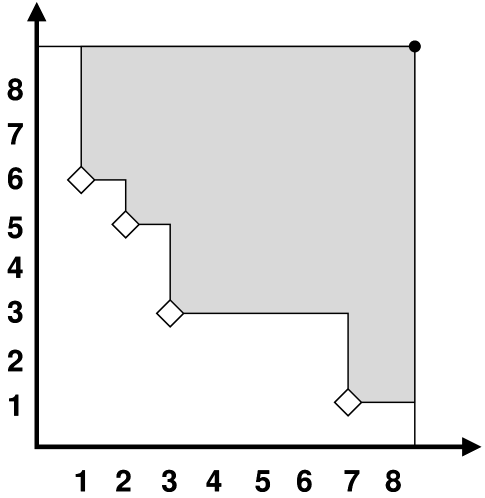
Hypervolume measure ≈ Anytime behaviour

Automatically Improving the Anytime Behavior



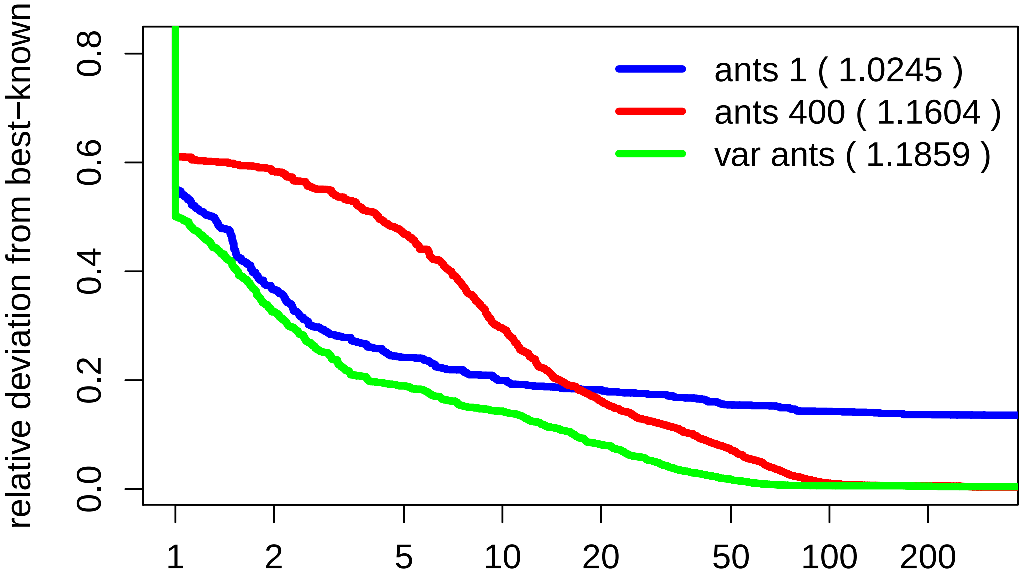
Hypervolume measure ≈ Anytime behaviour

Automatically Improving the Anytime Behavior



Hypervolume measure ≈ Anytime behaviour

Automatically Improving the Anytime Behavior



Automatically Improving the Anytime Behavior

`irace` + hypervolume = automatically improving the anytime behavior of optimization algorithms

1. Run configuration until large stopping time
2. Compute hypervolume of SQT curve
3. Evaluate anytime behavior according to hypervolume

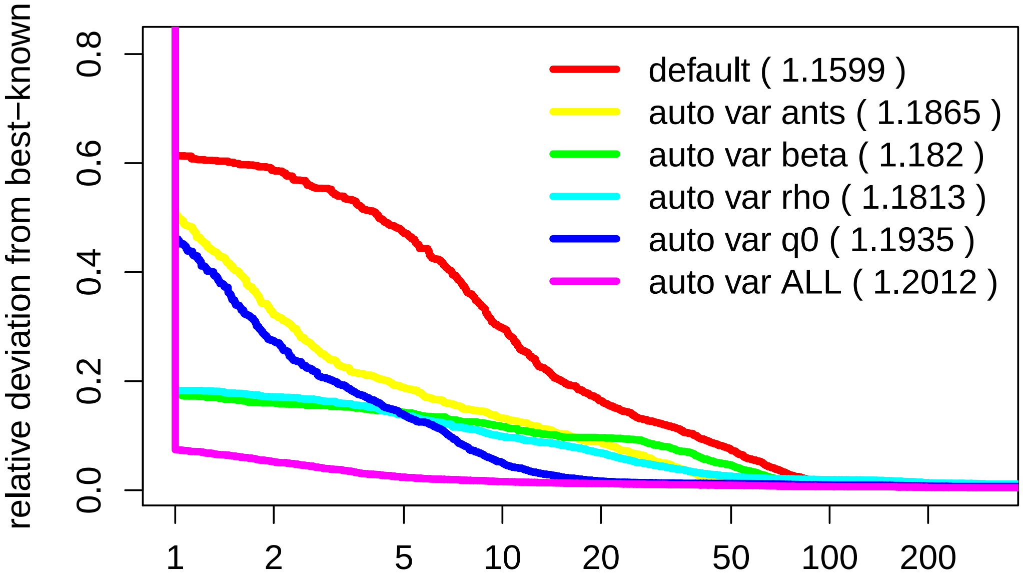
Automatically Improving the Anytime Behavior

`irace` + hypervolume = automatically improving the anytime behavior of optimization algorithms

1. Run configuration until large stopping time
2. Compute hypervolume of SQT curve
3. Evaluate anytime behavior according to hypervolume

- Hypervolume (multi-objective) optimization
  - ✔ Objectively defined comparison
  - ✔ Well-known performance measure
- Automatic configuration using irace
  - ✔ Most effort done by the computer
  - ✔ Best configurations selected by the computer: *Unbiased*

Scenario #1: Experimental comparison

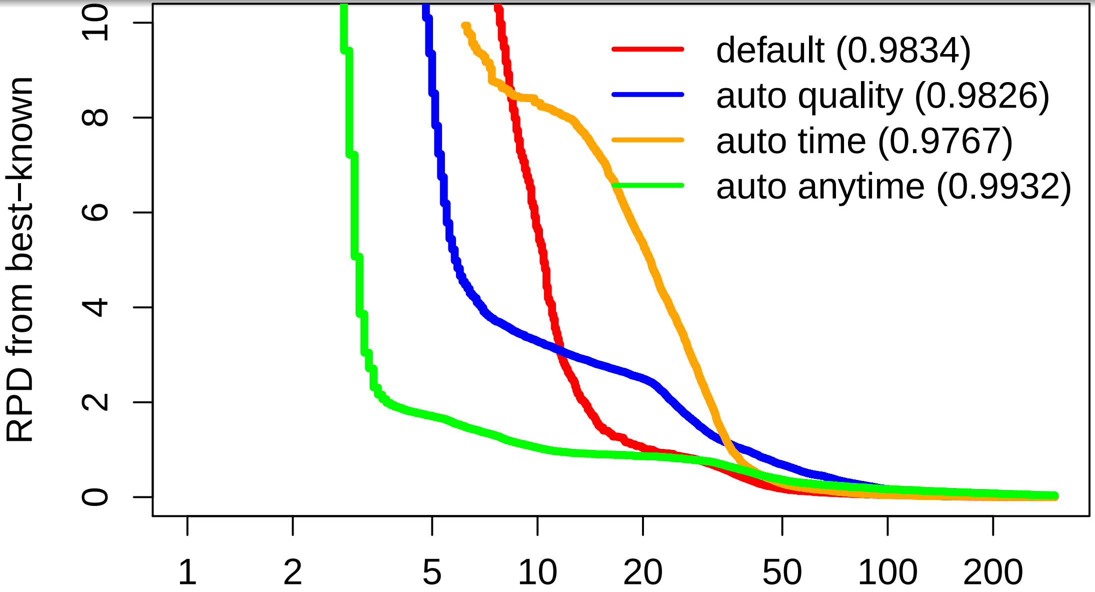


Scenario #2: SCIP [López-Ibáñez & Stützle, 2014]

SCIP: an open-source mixed integer programming (MIP) solver [Achterberg, 2009]

- 200 parameters controlling search, heuristics, thresholds, ...
- Benchmark set: Winner determination problem for combinatorial auctions [Leyton-Brown et al., 2000]
  1 000 training + 1 000 testing instances
- Single run timeout: 300 seconds
- irace budget (*maxExperiments*): 5 000 runs

Scenario #2: SCIP



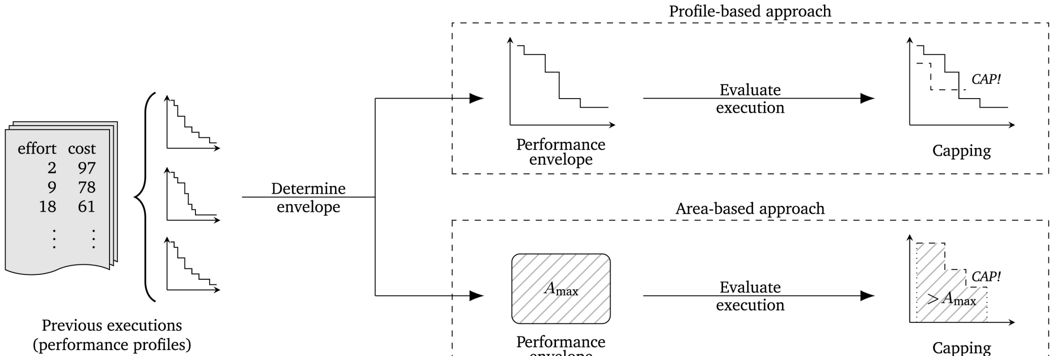
Example #2

Capping methods for the automatic configuration of optimization algorithms

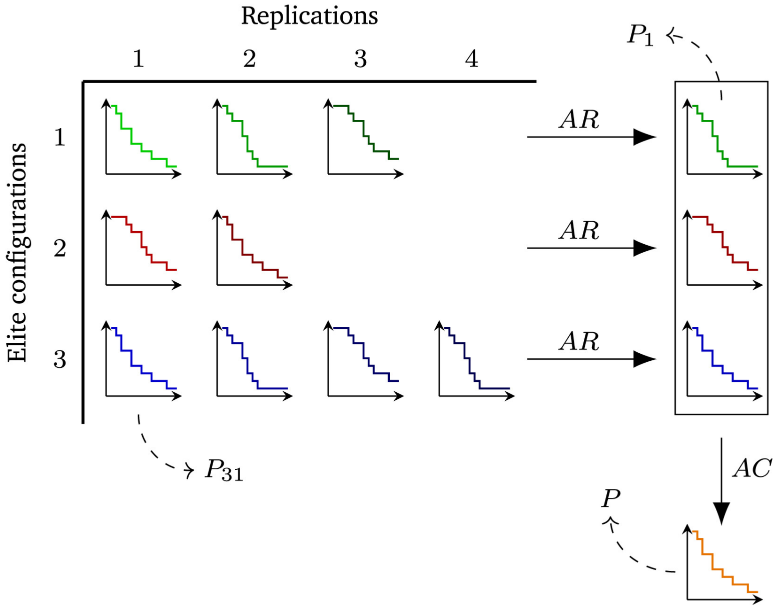
Marcelo De Souza, Marcus Ritt, and Manuel López-Ibáñez. **Capping methods for the automatic configuration of optimization algorithms**. *Computers & Operations Research*,2022. doi: 10.016/j.cor.2021.105615.

Capping methods for automatic algorithm configuration [De Souza et al., 2022]

- Adaptive capping only useful for decision algorithms (*time-to-target*)
- In many scenarios, we optimize cost over time until maximum termination time
  - ✘ Running bad configurations until maximum time is *wasteful*
  - ✔ Terminate (*cap*) bad configurations as soon as possible

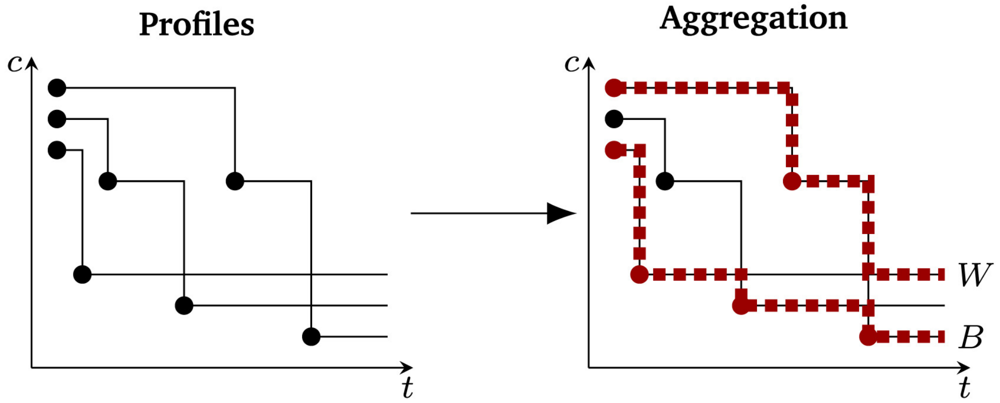


Capping methods for automatic algorithm configuration [De Souza et al., 2022]



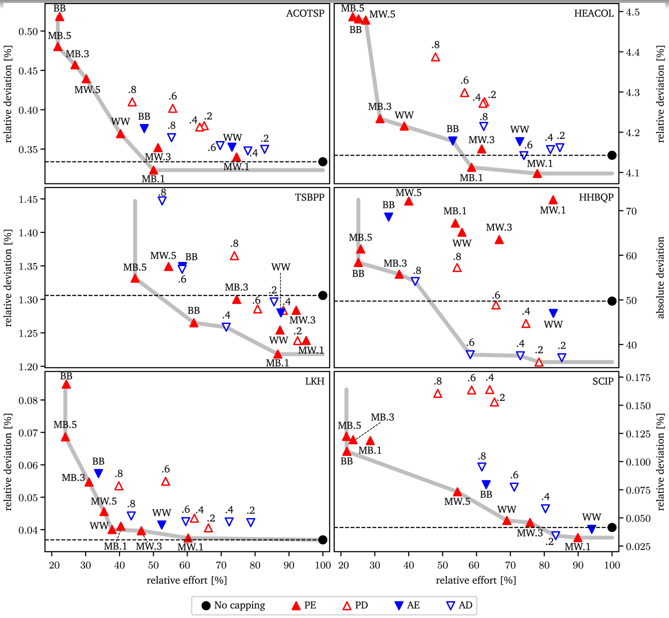
Capping methods for automatic algorithm configuration [De Souza et al., 2022]

Best and Worst Profile-based Aggregation Methods:



Capping methods for automatic algorithm configuration [De Souza et al., 2022]

P = profile-based
A = area-based
E = elitist
D = adaptive
B = best
W = worst
M = model



Capping methods for automatic algorithm configuration [De Souza et al., 2022]

Category	Capping method	Relative effort [%]	Quality loss [%]					
			ACOTSP	HEACOL	TSBPP	HHBQP	LKH	SCIP
Conservative	AD.4	76.1	0.02	0.02	−0.05	−12.24	0.00	0.02
Aggressive	PEMB.1	53.0	−0.01	−0.03	−0.09	17.47	0.00	0.08

- AEBB for scenarios where the configuration budget is time
- Python add-on for irace: https://github.com/souzamarcelo/capopt

Why automatic algorithm configuration and design?

1. More scientific, more principled
2. The end of the up-the-wall game
3. Computing power is exponentially cheaper
4. AC tools are becoming better
5. More interesting, fun and useful

Reason #1: More scientific, more principled

- ✓ Reproducible results
- ✓ Fairer comparisons (best-effort)
- ✓ Avoid / reduce human biases
- ✓ Codify good practices

Reason #1: More scientific, more principled

- ✓ Reproducible results
- ✓ Fairer comparisons (best-effort)
- ✓ Avoid / reduce human biases
- ✓ Codify good practices

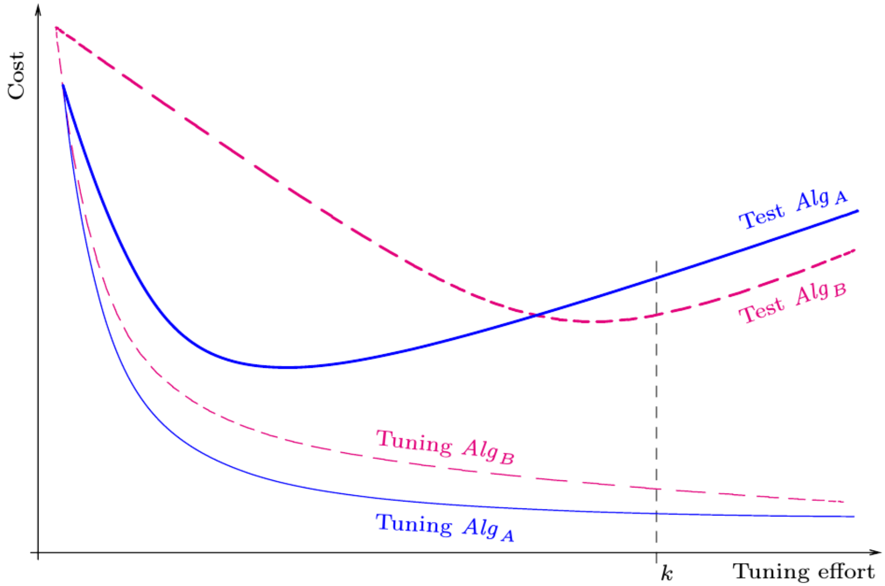
“*For procedures that require parameter tuning, the available data must be partitioned into a training and a test set. Tuning should be performed in the training set only.*”

[Journal of Heuristics: Policies on Heuristic Search Research]

“*The performance of swarm intelligence algorithms* [. . .] *is often strongly dependent on the value of the algorithm parameters. Such values should be set using either sound statistical procedures* [. . .] *or automatic parameter tuning procedures.*”

[Swarm Intelligence Journal (Springer)]

Over-tuning



(Taken from Birattari [2009])

Reason #2: The End of the Game

> *The Journal of Heuristics does not endorse the up-the-wall game.*
>
> [Journal of Heuristics: Policies on Heuristic Search Research]

> *True innovation in metaheuristics research therefore does not come from yet another method that performs better than its competitors, certainly if it is not well understood why exactly this method performs well.* *[Sörensen, 2015]*

- Finding a state-of-the-art algorithm is "easy":

  problem modeling + algorithmic components + computing power

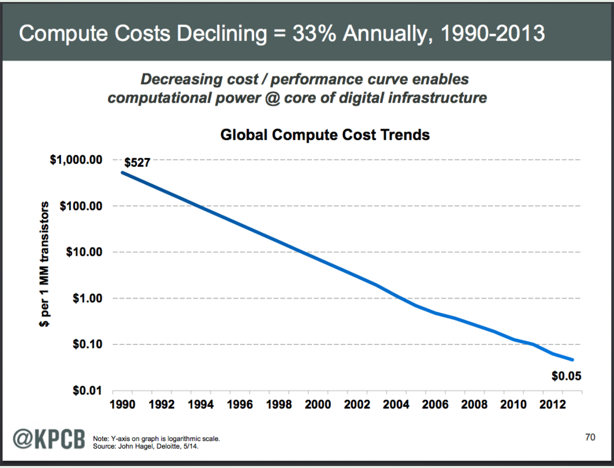
- *What* novel components? *Why* they work? *When* they work?

Reason #3: Computing power is exponentially cheaper

Reason #3: Computing power is exponentially cheaper

Algorithm Configuration in the Cloud [Geschwender et al., 2014]

Amazon EC2, 8 cores, 7GB memory, $ 0.58/hour



Reason #4: AC tools are becoming better

- Complex parameter spaces: numerical, categorical, ordinal, subordinate (conditional), constraints
- Large parameter spaces (hundreds of parameters)
- Heterogeneous problem instances
- Medium to large configuration budgets
  (few hundred to many thousands of runs)
- Individual runs may require from seconds to hours
- Multi-core CPUs, MPI, distributed computation clusters

☞ Modern automatic configuration tools (irace, SMAC, . . .)
are general, flexible, powerful and easy to use

Reason #5: More interesting, fun, and useful

✘ Classical optimization research:

1. Human-driven design to outperform other algorithmic designs
2. Analysis of the human-designed algorithm

Reason #5: More interesting, fun, and useful

✗ Classical optimization research:

1. Human-driven design to outperform other algorithmic designs
2. Analysis of the human-designed algorithm

✓ Paradigm shift in optimisation research:

From monolithic algorithms
to flexible frameworks of algorithmic components

1. Humans devise *novel* algorithmic components
2. Data-driven CPU-intensive automatic design
3. Analysis of generated data
4. Human-driven improvement of components

QUIT LIVING IN THE PAST

Acknowledgments

This tutorial has benefited from collaborations and discussions with my colleagues:

Thomas Stützle, Leslie Pérez Cáceres, Prasanna Balaprakash, Leonardo Bezerra, Mauro Birattari, Jérémie Dubois-Lacoste, Alberto Franzin, Holger H. Hoos, Frank Hutter, Kevin Leyton-Brown, Tianjun Liao, Marie-Eléonore Marmion, Franco Mascia, Marco Montes de Oca, Federico Pagnozzi, Zhi Yuan, Marcus Ritt, Marcelo De Souza

References I

T. Achterberg. SCIP: Solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, July 2009.

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 *Lecture Notes in Computer Science*, pages 108–122. Springer, Heidelberg, Germany, 2007. doi: 10.1007/978-3-540-75514-2_9.

A. Benavoli, G. Corani, F. Mangili, and M. Zaffalon. A Bayesian nonparametric procedure for comparing algorithms. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015*, volume 37, pages 1264–1272. PMLR, 2015.

L. C. T. Bezerra, M. López-Ibáñez, and T. Stützle. Automatic component-wise design of multi-objective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 20(3):403–417, 2016. doi: 10.1109/TEVC.2015.2474158.

M. Birattari. *The Problem of Tuning Metaheuristics as Seen from a Machine Learning Perspective*. PhD thesis, IRIDIA, École polytechnique, Université Libre de Bruxelles, Belgium, 2004.

M. Birattari. *Tuning Metaheuristics: A Machine Learning Perspective*, volume 197 of *Studies in Computational Intelligence*. Springer, Berlin/Heidelberg, 2009. doi: 10.1007/978-3-642-00483-4.

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.

M. Birattari, Z. Yuan, P. Balaprakash, and T. Stützle. F-race and iterated F-race: An overview. In T. Bartz-Beielstein, M. Chiarandini, L. Paquete, and M. Preuss, editors, *Experimental Methods for the Analysis of Optimization Algorithms*, pages 311–336. Springer, Berlin/Heidelberg, 2010. doi: 10.1007/978-3-642-02538-9_13.

B. Bischl, M. Lang, J. Bossek, L. Judt, J. Richter, T. Kuehn, and E. Studerus. *mlr: Machine Learning in R*, 2013. URL http://cran.r-project.org/package=mlr. R package.

B. Bischl, M. Lang, L. Kotthoff, J. Schiffner, J. Richter, E. Studerus, G. Casalicchio, and Z. M. Jones. mlr: Machine learning in R. *Journal of Machine Learning Research*, 17(170):1–5, 2016.

W. J. Conover. *Practical Nonparametric Statistics*. John Wiley & Sons, New York, NY, 3rd edition, 1999.

N. Dang and C. Doerr. Hyper-parameter tuning for the $(1 + (\lambda, \lambda))$ GA. In M. López-Ibáñez, A. Auger, and T. Stützle, editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2019*, pages 889–897. ACM Press, New York, NY, 2019. ISBN 978-1-4503-6111-8. doi: 10.1145/3321707.3321725.

M. De Souza, M. Ritt, and M. López-Ibáñez. Capping methods for the automatic configuration of optimization algorithms. *Computers & Operations Research*, 139: 105615, 2022. doi: 10.1016/j.cor.2021.105615.

References II

T. Dean and M. S. Boddy. An analysis of time-dependent planning. In H. E. Shrobe, T. M. Mitchell, and R. G. Smith, editors, *Proceedings of the 7th National Conference on Artificial Intelligence, AAAI-88*, pages 49–54. AAAI Press/MIT Press, Menlo Park, CA, 1988. URL http://www.aaai.org/Conferences/AAAI/aaai88.php.

J. E. Diaz and M. López-Ibáñez. Incorporating decision-maker's preferences into the automatic configuration of bi-objective optimisation algorithms. *European Journal of Operational Research*, 289(3):1209–1222, 2021. doi: 10.1016/j.ejor.2020.07.059.

M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter. Efficient and robust automated machine learning. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NIPS 28)*, pages 2962–2970, 2015. URL http://papers.nips.cc/book/advances-in-neural-information-processing-systems-28-2015.

G. Francesca, M. Brambilla, A. Brutschy, L. Garattoni, R. Miletitch, G. Podevijn, A. Reina, T. Soleymani, M. Salvaro, C. Pinciroli, F. Mascia, V. Trianni, and M. Birattari. AutoMoDe-Chocolate: Automatic design of control software for robot swarms. *Swarm Intelligence*, 2015. doi: 10.1007/s11721-015-0107-9.

T. Friedrich, F. Quinzan, and M. Wagner. Escaping large deceptive basins of attraction with heavy-tailed mutation operators. In H. E. Aguirre and K. Takadama, editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2018*, pages 293–300. ACM Press, New York, NY, 2018. doi: 10.1145/3205455.3205515.

D. Geschwender, F. Hutter, L. Kotthoff, Y. Malitsky, H. H. Hoos, and K. Leyton-Brown. Algorithm configuration in the cloud: A feasibility study. In P. M. Pardalos, M. G. C. Resende, C. Vogiatzis, and J. L. Walteros, editors, *Learning and Intelligent Optimization, 8th International Conference, LION 8*, volume 8426 of *Lecture Notes in Computer Science*, pages 41–46. Springer, Heidelberg, Germany, 2014. doi: 10.1007/978-3-319-09584-4_5.

G. T. Hall, P. S. Oliveto, and D. Sudholt. On the impact of the cutoff time on the performance of algorithm configurators. In M. López-Ibáñez, A. Auger, and T. Stützle, editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2019*, pages 907–915. ACM Press, New York, NY, 2019. ISBN 978-1-4503-6111-8. doi: 10.1145/3321707.3321879.

H. H. Hoos. Programming by optimization. *Communications of the ACM*, 55(2):70–80, Feb. 2012. doi: 10.1145/2076450.2076469.

L. Kotthoff, C. Thornton, H. H. Hoos, F. Hutter, and K. Leyton-Brown. Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA. *Journal of Machine Learning Research*, 17:1–5, 2016.

M. Lang, H. Kotthaus, P. Marwedel, C. Weihs, J. Rahnenführer, and B. Bischl. Automatic model selection for high-dimensional survival analysis. *Journal of Statistical Computation and Simulation*, 85(1):62–76, 2014. doi: 10.1080/00949655.2014.929131.

K. Leyton-Brown, M. Pearson, and Y. Shoham. Towards a universal test suite for combinatorial auction algorithms. In A. Jhingran et al., editors, *ACM Conference on Electronic Commerce (EC-00)*, pages 66–76. ACM Press, New York, NY, 2000. doi: 10.1145/352871.352879.

References III

M. López-Ibáñez and J. D. Knowles. Machine decision makers as a laboratory for interactive EMO. In A. Gaspar-Cunha, C. H. Antunes, and C. A. Coello Coello, editors, *Evolutionary Multi-criterion Optimization, EMO 2015 Part II*, volume 9019 of *Lecture Notes in Computer Science*, pages 295–309. Springer, Heidelberg, Germany, 2015. doi: 10.1007/978-3-319-15892-1_20.

M. López-Ibáñez and T. Stützle. The automatic design of multi-objective ant colony optimization algorithms. *IEEE Transactions on Evolutionary Computation*, 16(6): 861–875, 2012. doi: 10.1109/TEVC.2011.2182651.

M. López-Ibáñez and T. Stützle. Automatically improving the anytime behaviour of optimisation algorithms. *European Journal of Operational Research*, 235(3):569–582, 2014. doi: 10.1016/j.ejor.2013.10.043.

M. López-Ibáñez, J. Dubois-Lacoste, T. Stützle, and M. Birattari. The irace package, iterated race for automatic algorithm configuration. Technical Report TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, Belgium, 2011. URL http://iridia.ulb.ac.be/IridiaTrSeries/link/IridiaTr2011-004.pdf. Published in Operations Research Perspectives López-Ibáñez et al. [2016].

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, 2016. doi: 10.1016/j.orp.2016.09.002.

M. López-Ibáñez, J. Branke, and L. Paquete. Reproducibility in evolutionary computation. *ACM Transactions on Evolutionary Learning and Optimization*, 1(4):1–21, 2021. doi: 10.1145/3466624.

O. Maron and A. W. Moore. The racing algorithm: Model selection for lazy learners. *Artificial Intelligence Research*, 11(1–5):193–225, 1997. doi: 10.1023/A:1006556606079.

R. Martín-Santamaría, M. López-Ibáñez, T. Stützle, and J. M. Colmenar. On the automatic generation of metaheuristic algorithms for combinatorial optimization problems. *European Journal of Operational Research*, 318(3):740–751, 2024. doi: 10.1016/j.ejor.2024.06.001.

F. Mascia, M. López-Ibáñez, J. Dubois-Lacoste, and T. Stützle. Grammar-based generation of stochastic local search heuristics through automatic algorithm configuration tools. *Computers & Operations Research*, 51:190–199, 2014. doi: 10.1016/j.cor.2014.05.020.

P. Miranda, R. M. Silva, and R. B. Prudêncio. Fine-tuning of support vector machine parameters using racing algorithms. In *European Symposium on Artificial Neural Networks, ESSAN*, pages 325–330, 2014.

L. Pérez Cáceres, M. López-Ibáñez, H. H. Hoos, and T. Stützle. An experimental study of adaptive capping in irace. In R. Battiti, D. E. Kvasov, and Y. D. Sergeyev, editors, *Learning and Intelligent Optimization, 11th International Conference, LION 11*, volume 10556 of *Lecture Notes in Computer Science*, pages 235–250. Springer, Cham, Switzerland, 2017a. doi: 10.1007/978-3-319-69404-7_17.

References IV

L. Pérez Cáceres, F. Pagnozzi, A. Franzin, and T. Stützle. Automatic configuration of GCC using irace: Supplementary material. http://iridia.ulb.ac.be/supp/IridiaSupp2017-009/, 2017b.

M. Silva-Muñoz, A. Franzin, and H. Bersini. Automatic configuration of the Cassandra database using irace. *PeerJ Computer Science*, 7:e634, 2021. doi: 10.7717/peerj-cs.634.

K. Sörensen. Metaheuristics—the metaphor exposed. *International Transactions in Operational Research*, 22(1):3–18, 2015. doi: 10.1111/itor.12001.

S. Zilberstein. Using anytime algorithms in intelligent systems. *AI Magazine*, 17(3):73–83, 1996. doi: 10.1609/aimag.v17i3.1232.