Best Practices and Pitfalls in the Empirical Analysis of Algorithms

Manuel López-Ibáñez
manuel.lopez-ibanez@manchester.ac.uk

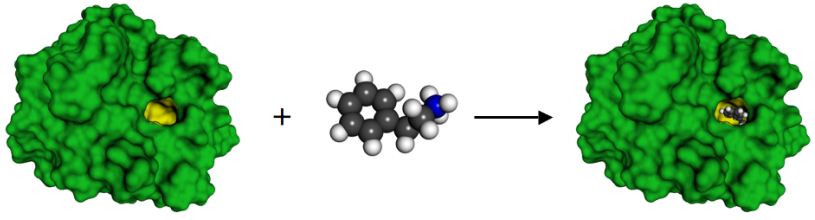
University of Manchester, UK

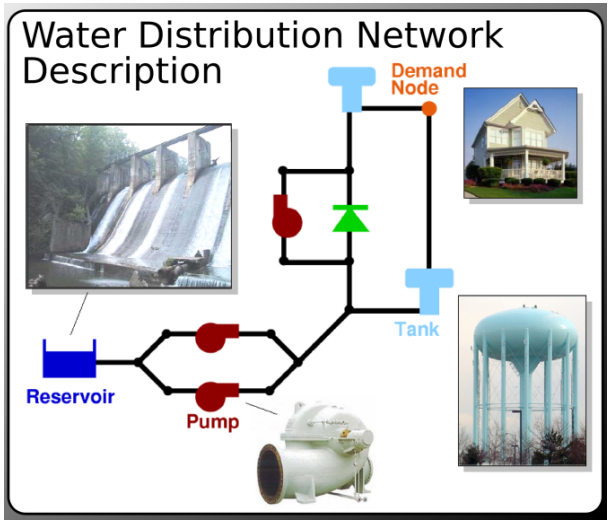
July 14, 2017
GECCO Summer School

The University of Manchester
Alliance Manchester Business School

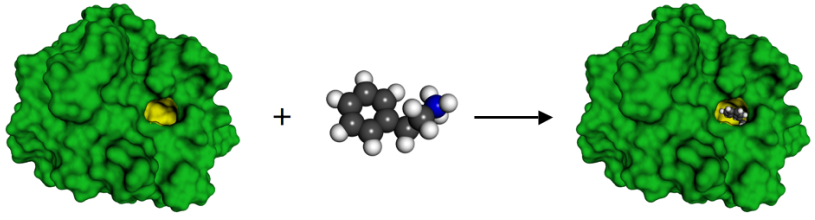

Optimisation
Algorithms
Data Science
Automatic configuration
Analytics
Metaheuristics
Planning
Stochastic
Heuristics
Scheduling
Decision
Making
Local Search
Evolutionary
Tuning
Machine
Learning
Efficiency
Dynamic
Multi-objective
Computational
Intelligence

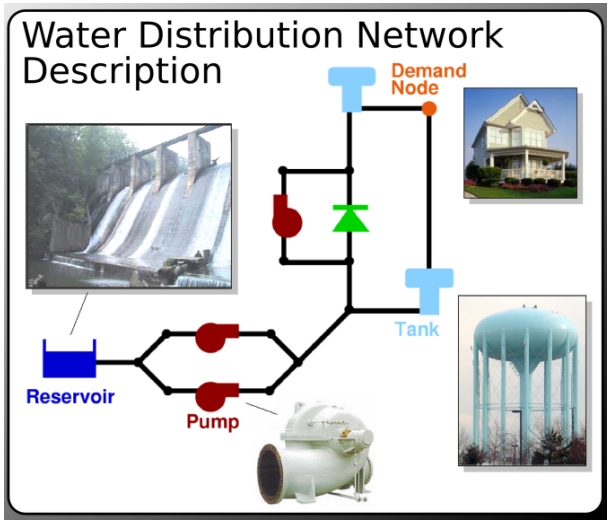
Optimisation problems are everywhere!





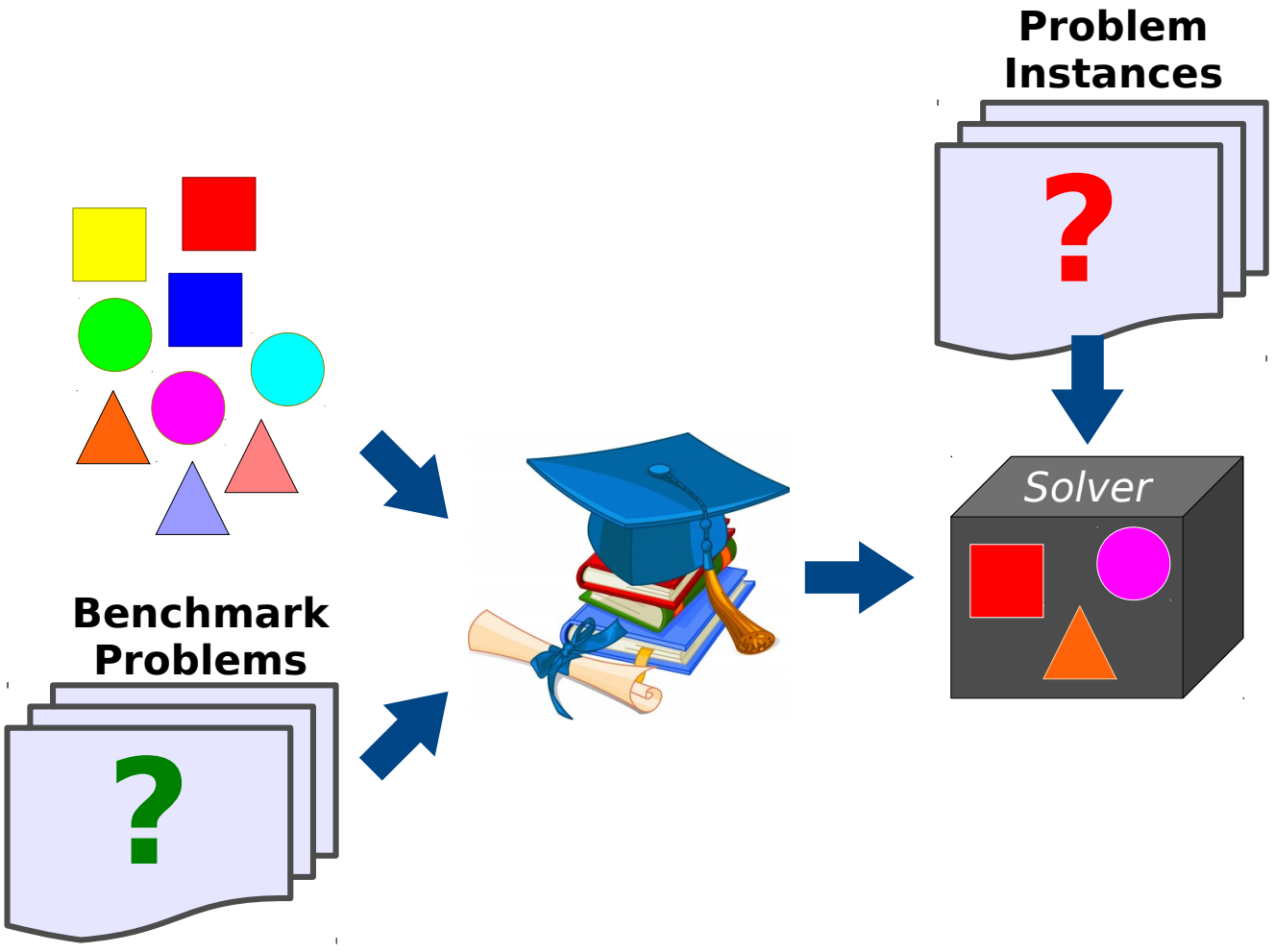
Optimisation problems are everywhere!





Most such problems are computationally hard

Traditional algorithm design



The algorithm design problem

$\mathcal{A}$: set of potential algorithms

The algorithm design problem

$\mathcal{A}$: set of potential algorithms

Θ_A : set of potential parameter configurations of algorithm $A \in \mathcal{A}$

The algorithm design problem

$\mathcal{A}$: set of potential algorithms

Θ_A : set of potential parameter configurations of algorithm $A \in \mathcal{A}$

$\mathcal{I}$: set of problem instances

The algorithm design problem

$\mathcal{A}$: set of potential algorithms

Θ_A : set of potential parameter configurations of algorithm $A \in \mathcal{A}$

$\mathcal{I}$: set of problem instances

$\mathcal{C}: \Theta_A \times \mathcal{I} \to \mathbb{R}$: cost (objective function) measure, where:

The algorithm design problem

$\mathcal{A}$: set of potential algorithms

Θ_A : set of potential parameter configurations of algorithm $A \in \mathcal{A}$

$\mathcal{I}$: set of problem instances

$\mathcal{C}: \Theta_A \times \mathcal{I} \to \mathbb{R}$: cost (objective function) measure, where:

$\mathcal{C}(\theta, i)$: solution cost returned by executing configuration $\theta \in \Theta_A$ on instance $i \in \mathcal{I}$

The algorithm design problem

$\mathcal{A}$: set of potential algorithms

Θ_A : set of potential parameter configurations of algorithm $A \in \mathcal{A}$

$\mathcal{I}$: set of problem instances

$\mathcal{C} : \Theta_A \times \mathcal{I} \to \mathbb{R}$: cost (objective function) measure, where:

$\mathcal{C}(\theta, i)$: solution cost returned by executing configuration $\theta \in \Theta_A$ on instance $i \in \mathcal{I}$

Φ_θ : a statistical parameter of the cost $\mathcal{C}$ returned by configuration θ with respect to the whole set $\mathcal{I}$, e.g., the mean

The algorithm design problem

$\mathcal{A}$: set of potential algorithms

Θ_A : set of potential parameter configurations of algorithm $A \in \mathcal{A}$

$\mathcal{I}$: set of problem instances

$\mathcal{C}: \Theta_A \times \mathcal{I} \to \mathbb{R}$: cost (objective function) measure, where:

$\mathcal{C}(\theta, i)$: solution cost returned by executing configuration $\theta \in \Theta_A$ on instance $i \in \mathcal{I}$

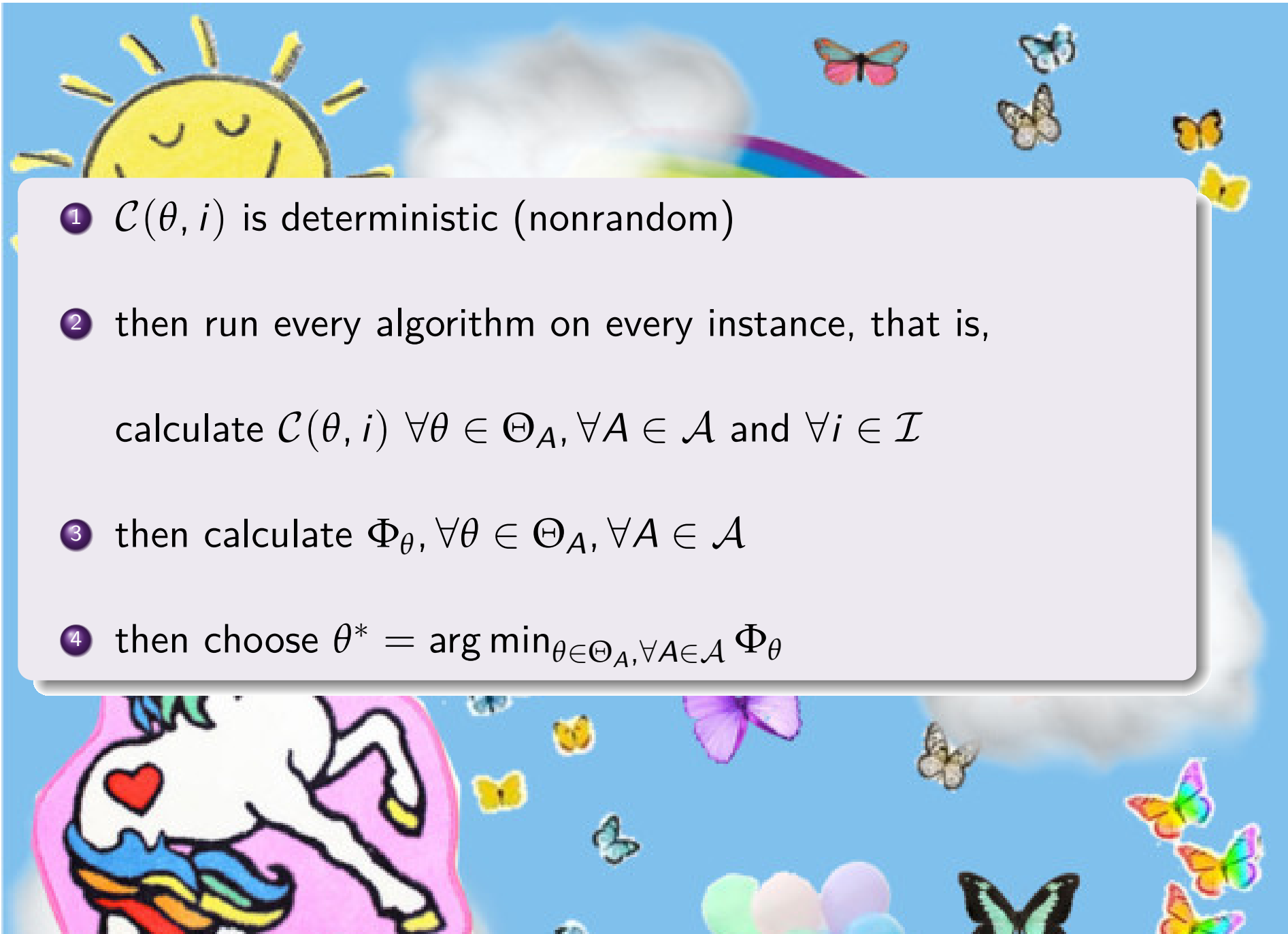
Φ_θ : a statistical parameter of the cost $\mathcal{C}$ returned by configuration θ with respect to the whole set $\mathcal{I}$, e.g., the mean

Find the best configuration θ^* such that:

$$\theta^* = \underset{\theta \in \Theta_A, \forall A \in \mathcal{A}}{\arg\min} \; \Phi_\theta$$

In the ideal world . . .

In the ideal world . . .



1. $\mathcal{C}(\theta, i)$ is deterministic (nonrandom)
2. then run every algorithm on every instance, that is,

   calculate $\mathcal{C}(\theta, i)\ \forall \theta \in \Theta_A, \forall A \in \mathcal{A}$ and $\forall i \in \mathcal{I}$
3. then calculate $\Phi_\theta, \forall \theta \in \Theta_A, \forall A \in \mathcal{A}$
4. then choose $\theta^* = \arg\min_{\theta \in \Theta_A, \forall A \in \mathcal{A}} \Phi_\theta$

. . . in the real world

$\mathcal{A}$: set of potential algorithms
(practically infinite)

. . . in the real world

$\mathcal{A}$: set of potential algorithms
(practically infinite)

Θ_A : set of potential parameter configurations of algorithm $A \in \mathcal{A}$
(practically infinite)

. . . in the real world

$\mathcal{A}$: set of potential algorithms
(practically infinite)

Θ_A : set of potential parameter configurations of algorithm $A \in \mathcal{A}$
(practically infinite)

$\mathcal{I}$: set of problem instances
(random variable, practically infinite)

. . . in the real world

$\mathcal{A}$: set of potential algorithms
(practically infinite)

Θ_A : set of potential parameter configurations of algorithm $A \in \mathcal{A}$
(practically infinite)

$\mathcal{I}$: set of problem instances
(random variable, practically infinite)

$\mathcal{C}(\theta, I)$: cost of executing configuration $\theta \in \Theta_A$ on instance $I \in \mathcal{I}$
(random variable, practically infinite)

. . . in the real world

$\mathcal{A}$: set of potential algorithms
(practically infinite)

Θ_A : set of potential parameter configurations of algorithm $A \in \mathcal{A}$
(practically infinite)

$\mathcal{I}$: set of problem instances
(random variable, practically infinite)

$\mathcal{C}(\theta, I)$: cost of executing configuration $\theta \in \Theta_A$ on instance $I \in \mathcal{I}$
(random variable, practically infinite)

. . . in the real world

$\mathcal{A}$: set of potential algorithms
(practically infinite)

Θ_A : set of potential parameter configurations of algorithm $A \in \mathcal{A}$
(practically infinite)

$\mathcal{I}$: set of problem instances
(random variable, practically infinite)

$\mathcal{C}(\theta, I)$: cost of executing configuration $\theta \in \Theta_A$ on instance $I \in \mathcal{I}$
(random variable, practically infinite)

How to tackle it?

One traditional approach

1. Expert chooses a number of algorithms $A_1, A_2, \ldots \in \mathcal{A}$

One traditional approach

1. Expert chooses a number of algorithms $A_1, A_2, \ldots \in \mathcal{A}$
2. Expert chooses a number of configurations $\theta_1, \theta_2, \ldots \in \Theta_{A_i}$

One traditional approach

1. Expert chooses a number of algorithms $A_1, A_2, \ldots \in \mathcal{A}$
2. Expert chooses a number of configurations $\theta_1, \theta_2, \ldots \in \Theta_{A_i}$
3. Expert chooses benchmark instances $I_1, I_2, \ldots \in \mathcal{I}$

One traditional approach

1. Expert chooses a number of algorithms $A_1, A_2, \ldots \in \mathcal{A}$
2. Expert chooses a number of configurations $\theta_1, \theta_2, \ldots \in \Theta_{A_i}$
3. Expert chooses benchmark instances $I_1, I_2, \ldots \in \mathcal{I}$
4. Expert chooses a number of runs N

One traditional approach

1. Expert chooses a number of algorithms $A_1, A_2, \ldots \in \mathcal{A}$
2. Expert chooses a number of configurations $\theta_1, \theta_2, \ldots \in \Theta_{A_i}$
3. Expert chooses benchmark instances $I_1, I_2, \ldots \in \mathcal{I}$
4. Expert chooses a number of runs N
5. Run N times each algorithm configuration on each instance, that is, sample N times from $\mathcal{C}(\theta_j, I_k)$

One traditional approach

1. Expert chooses a number of algorithms $A_1, A_2, \ldots \in \mathcal{A}$
2. Expert chooses a number of configurations $\theta_1, \theta_2, \ldots \in \Theta_{A_i}$
3. Expert chooses benchmark instances $I_1, I_2, \ldots \in \mathcal{I}$
4. Expert chooses a number of runs N
5. Run N times each algorithm configuration on each instance, that is, sample N times from $\mathcal{C}(\theta_j, I_k)$
6. Calculate $\hat{\phi}_{\theta_j}$ and choose the best $\theta^j_{A_i}$

One traditional approach

1. Expert chooses a number of algorithms $A_1, A_2, \ldots \in \mathcal{A}$
2. Expert chooses a number of configurations $\theta_1, \theta_2, \ldots \in \Theta_{A_i}$
3. Expert chooses benchmark instances $I_1, I_2, \ldots \in \mathcal{I}$
4. Expert chooses a number of runs N
5. Run N times each algorithm configuration on each instance, that is, sample N times from $\mathcal{C}(\theta_j, I_k)$
6. Calculate $\hat{\phi}_{\theta_j}$ and choose the best $\theta^j_{A_i}$

One traditional approach

1. Expert chooses a number of algorithms $A_1, A_2, \ldots \in \mathcal{A}$
2. Expert chooses a number of configurations $\theta_1, \theta_2, \ldots \in \Theta_{A_i}$
3. Expert chooses benchmark instances $I_1, I_2, \ldots \in \mathcal{I}$
4. Expert chooses a number of runs N
5. Run N times each algorithm configuration on each instance, that is, sample N times from $\mathcal{C}(\theta_j, I_k)$
6. Calculate $\hat{\phi}_{\theta_j}$ and choose the best $\theta^j_{A_i}$

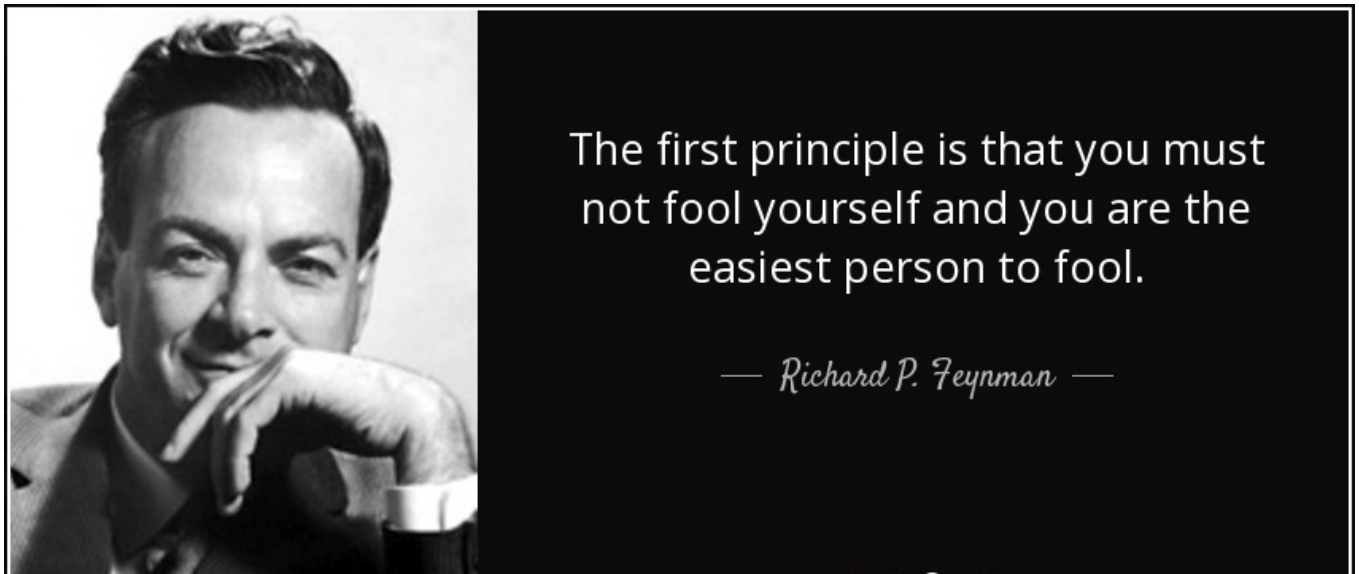
So many things can go wrong!

Pitfalls [Johnson, 2002; Gent et al., 1997]

- ✘ Ignoring stochasticity of algorithms
- ✘ Ignoring stochasticity of problem instances
- ✘ Under-tuning algorithm configurations
- ✘ Over-tuning algorithm configurations
- ✘ Unfair comparisons
  - ✘ Incomparable computational effort (iterations vs FEs vs time)
- ✘ Lack of details: language, hardware, settings, ...
- ✘ Lack of insights or contributions

Pitfalls [Johnson, 2002; Gent et al., 1997]

- ✘ Ignoring stochasticity of algorithms
- ✘ Ignoring stochasticity of problem instances
- ✘ Under-tuning algorithm configurations
- ✘ Over-tuning algorithm configurations
- ✘ Unfair comparisons
  - ✘ Incomparable computational effort (iterations vs FEs vs time)
- ✘ Lack of details: language, hardware, settings, ...
- ✘ Lack of insights or contributions



Heuristic algorithms are often stochastic

$\mathcal{C}(\theta_A, I)$ is often stochastic:

$\Rightarrow$ Every run gives a different result,

$\Rightarrow$ We are interested in the *expected* behaviour of a future run

$\Rightarrow$ The only way to estimate is to perform multiple runs (sample)

Heuristic algorithms are often stochastic

$\mathcal{C}(\theta_A, I)$ is often stochastic:

$\Rightarrow$ Every run gives a different result,

$\Rightarrow$ We are interested in the *expected* behaviour of a future run

$\Rightarrow$ The only way to estimate is to perform multiple runs (sample)

✘✘ A single run on a single problem instance

Heuristic algorithms are often stochastic

$\mathcal{C}(\theta_A, I)$ is often stochastic:

$\Rightarrow$ Every run gives a different result,

$\Rightarrow$ We are interested in the *expected* behaviour of a future run

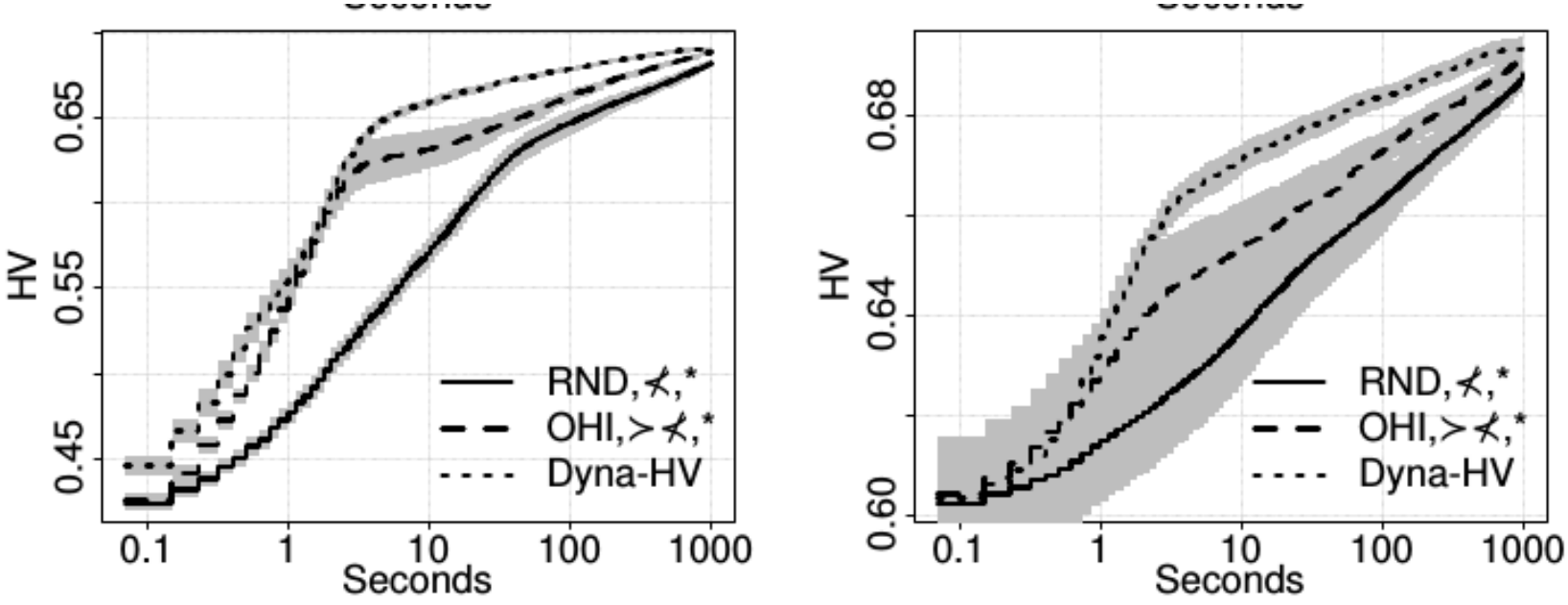
$\Rightarrow$ The only way to estimate is to perform multiple runs (sample)

✘✘ A single run on a single problem instance

✘ Mean/median without variance

! Small/large variance may often indicate the presence of outliers

Heuristic algorithms are often stochastic



Taken from Dubois-Lacoste, López-Ibáñez & Stützle [2015]

Reporting biased statistics

> *We improved 11 best-known results*

> *Our algorithm obtained the best solution in most benchmark instances*

✘ Comparing best (worst) values out of N runs is *wrong*

- Best / worst (the sample min / max) is a biased estimator of expected value
- Any heuristic that constructs a random initial solution may find the optimal by chance
- Deceive yourself by making successive improvements looking at single/best results

Problem instances are random variables

Optimization algorithms are designed to solve problem instances that were not known by the algorithm designer*

Problem instances are random variables

Optimization algorithms are designed to solve problem instances that were not known by the algorithm designer*

* Example of exception: The Eternity Puzzle ($1 million prize)

Problem instances are random variables

Optimization algorithms are designed to solve problem instances that were not known by the algorithm designer*

- We design and evaluate for a benchmark $\mathrm{I}^{\mathrm{bench}} \subset \mathcal{I}$
- When deployed, the algorithm solves $I_k \sim \mathcal{I}$
- ✘ The best algorithm for $\mathrm{I}^{\mathrm{bench}}$ may not be the best algorithm for $\mathcal{I}/\mathrm{I}^{\mathrm{bench}}$

Problem instances are random variables

Comparison of 10 algorithms on hard but not impossible instances (run the state-of-the-art method and discard all instances that are always solved to $\leq 1\%$ or $\geq 10\%$ of the optimal)

Problem instances are random variables

Comparison of 10 algorithms on hard but not impossible instances (run the state-of-the-art method and discard all instances that are always solved to $\leq 1\%$ or $\geq 10\%$ of the optimal)

- Some of the algorithms are statistically better than the SOA !

Problem instances are random variables

Comparison of 10 algorithms on hard but not impossible instances (run the state-of-the-art method and discard all instances that are always solved to $\leq 1\%$ or $\geq 10\%$ of the optimal)

- Some of the algorithms are statistically better than the SOA !

✔ *PROFIT !!!????*

Problem instances are random variables

Comparison of 10 algorithms on hard but not impossible instances (run the state-of-the-art method and discard all instances that are always solved to $\leq 1\%$ or $\geq 10\%$ of the optimal)

- Some of the algorithms are statistically better than the SOA !

✔ *PROFIT !!!????*

✘ The algorithms are just the SOA + relabeling the input variables, which should have *NO effect*

What went wrong?

Problem instances are random variables

Comparison of 10 algorithms on hard but not impossible instances (run the state-of-the-art method and discard all instances that are always solved to $\leq 1\%$ or $\geq 10\%$ of the optimal)

- Some of the algorithms are statistically better than the SOA !

✔ *PROFIT !!!????*

✘ The algorithms are just the SOA + relabeling the input variables, which should have *NO effect*

What went wrong?

Fischetti & Monaci [2014]. **"Exploiting erraticism in search".** *Operations Research*, 62:1:114-122.
http://mat.tepper.cmu.edu/blog/?p=1695

Problem instances are random variables

Fischetti & Monaci [2014]. **"Exploiting erraticism in search".** *Operations Research*, 62:1:114-122.
http://mat.tepper.cmu.edu/blog/?p=1695

Conclusion # 1: Biased benchmarks

Do not choose benchmark instances
depending on the algorithms under comparison.

Problem instances are random variables

Fischetti & Monaci [2014]. **"Exploiting erraticism in search".** *Operations Research*, 62:1:114-122.
http://mat.tepper.cmu.edu/blog/?p=1695

Conclusion # 1: Biased benchmarks

Do not choose benchmark instances
depending on the algorithms under comparison.

Conclusion # 2: Problem instances introduce stochasticity

Even if your algorithm is *deterministic*, the input (the instance) is not and non-random decisions based on irrelevant details of the input may lead to *variance and biases*

Under-tuning

- All interesting optimization algorithms have (hidden) parameters
- Even *"adaptive"* methods have parameters that can be tuned
- The perfect tuner: Returns the best θ_A for solving $\mathcal{I}$
- Human designer $\Rightarrow$ a non-automatic, biased, unspecified tuner

Under-tuning

- All interesting optimization algorithms have (hidden) parameters
- Even *"adaptive"* methods have parameters that can be tuned
- The perfect tuner: Returns the best θ_A for solving $\mathcal{I}$
- Human designer $\Rightarrow$ a non-automatic, biased, unspecified tuner

- ✘ Comparisons between un-tuned methods are useless
- ✘ Default settings are often very poor
- ✘ Unfair comparisons against settings tuned for a different experimental setup

Under-tuning

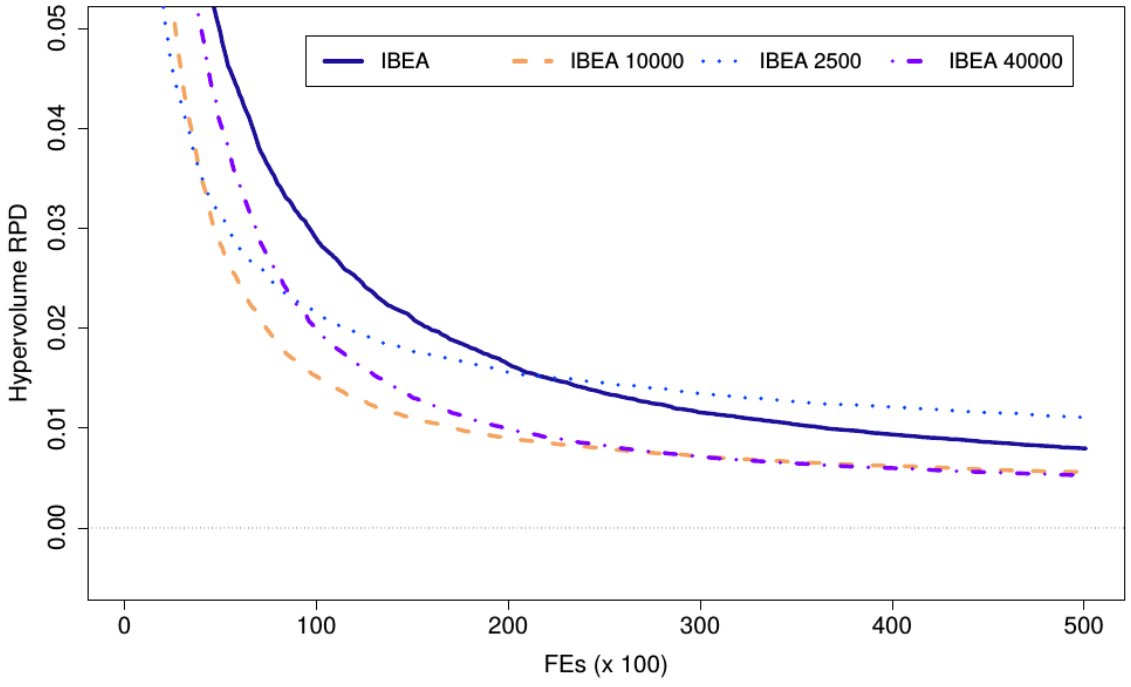
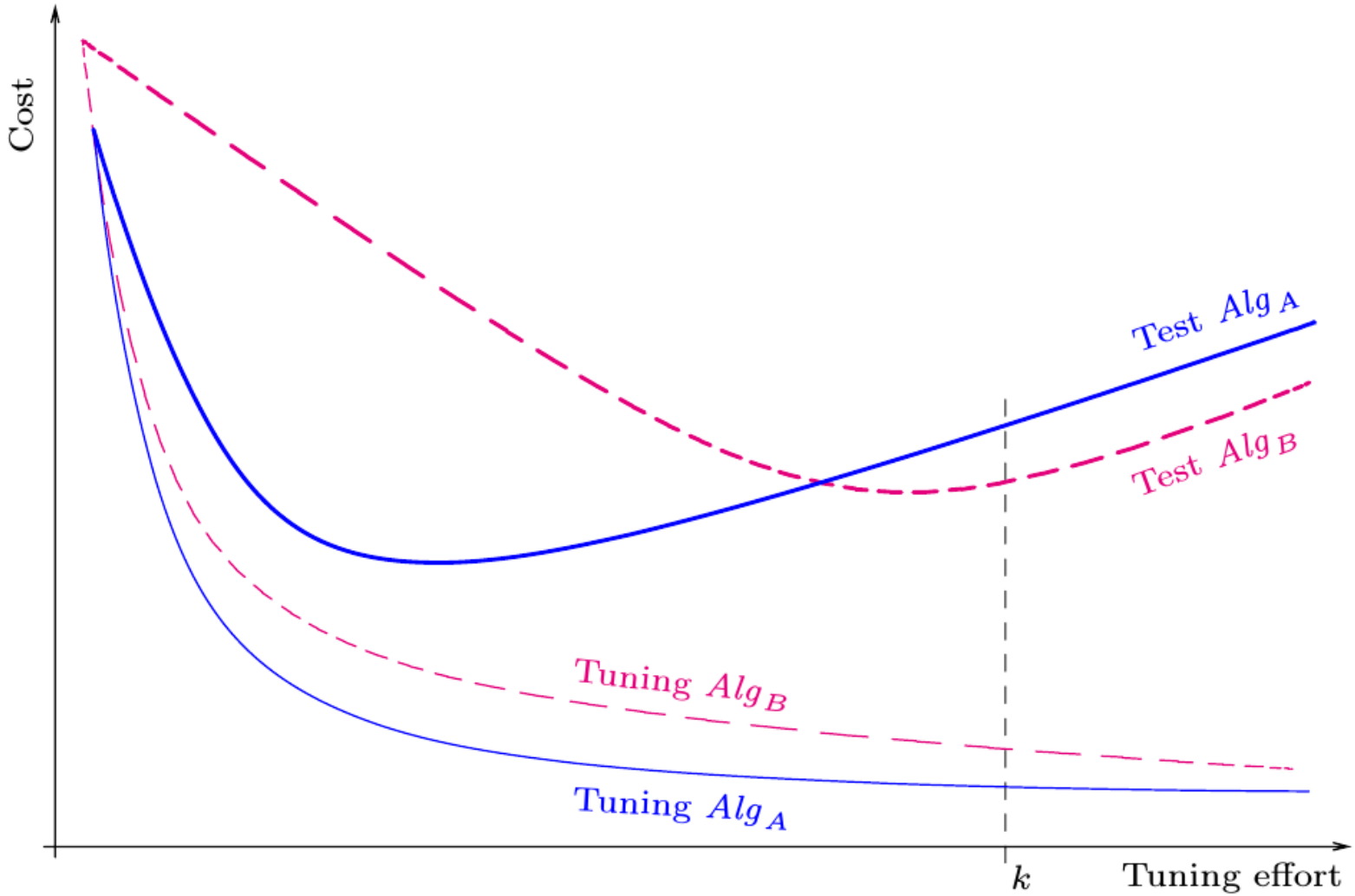


Figure 2: I_H^{rpd} performance of IBEA using DE as underlying EA with different numerical parameter settings on WFG8 ($M = 3, n_{\text{var}} = 30$).

Taken from Bezerra, López-Ibáñez & Stützle [2017]

Over-tuning

$$\mathrm{I}^{\text{tuning}}, \mathrm{I}^{\text{test}} \quad \subset \quad \mathcal{I}$$



Taken from Birattari [2009]

Over-tuning

> *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)]*

> *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]*

Over-tuning

> *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)]*

> *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]*

Any design step is tuning, even if the tuner is a human!

Good practices [Johnson, 2002; Gent et al., 1997]

- ✔ Statistical analysis and tests
- ✔ Graphical analysis that shows both average behavior and variance
- ✔ Automatic parameter configuration
- ✔ Analysis of algorithm components
- ✔ Replicability
  - ✔ Provide source code, problem instances, raw data
  - ✔ Replicable computational effort
- ✔ Report negative results [Fanelli, 2012]

Automatic Parameter Configuration

A stochastic black-box optimisation problem

- Mixed decision variables:

  discrete (categorical, ordinal, integer) and continuous

- Stochasticity from algorithm and problem instances
- Black-box: evaluation requires running the algorithm

Automatic Parameter Configuration

A stochastic black-box optimisation problem

- Mixed decision variables:

  discrete (categorical, ordinal, integer) and continuous

- Stochasticity from algorithm and problem instances
- Black-box: evaluation requires running the algorithm

Methods for Automatic Algorithm Configuration

- **SPO** [Bartz-Beielstein, Lasarczyk & Preuss, 2005]
- **ParamILS** [Hutter, Hoos & Stützle, 2007]
- **GGA** [Ansótegui, Sellmann & Tierney, 2009]
- **SMAC** [Hutter, Hoos & Leyton-Brown, 2011]
- **IRACE** [López-Ibáñez, Dubois-Lacoste, Stützle & Birattari, 2011]

Automatic Parameter Configuration: Benefits

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

The Up-The-Wall Game

> *The Journal of Heuristics does not endorse the up-the-wall game.* *[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]*

The Up-The-Wall Game

> *The Journal of Heuristics does not endorse the up-the-wall game.* *[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":

  irace + algorithmic components + computing power

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

Conclusions

- "Testing Heuristics: We Have It All Wrong" [Hooker, 1996] two decades after, still sometimes wrong
- ✘ It is *very easy* to fool oneself

  $\Rightarrow$ Be aware of pitfalls, follow best practices
- ✔ Automatic tools for benchmarking and analysis help to avoid pitfalls
- ✔ Automatic configuration and design tools will (hopefully) be the end of the up-the-wall game

Assignment

- Survey of pitfalls and good practices
- Attend many presentations / posters during GECCO 2017
- Listen, read original papers, *DO NOT* annoy presenters:

  The goal is NOT to criticise individual papers but to give a general overview of what are most common practices and demonstrate self-reflection

- Deliverable: Fully referenced report (3-5 pages), detailing which are the most typical good and bad practices observed

Challenge

Do you need to use the hypervolume measure in order to tune multi-objective optimizers by means of irace?

Tutorial: **Automated Offline Design of Algorithms** (Saturday 08:30 - 10:20)

Best Practices and Pitfalls in the Empirical Analysis of Algorithms

Manuel López-Ibáñez
manuel.lopez-ibanez@manchester.ac.uk

University of Manchester, UK

July 14, 2017
GECCO Summer School

http://lopez-ibanez.eu

The University of Manchester
Alliance Manchester Business School

References I

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 *Lecture Notes in Computer Science*, pages 142–157. Springer, Heidelberg, Germany, 2009. doi: 10.1007/978-3-642-04244-7_14.

T. Bartz-Beielstein, C. Lasarczyk, and M. Preuss. Sequential parameter optimization. In *Proceedings of the 2005 Congress on Evolutionary Computation (CEC 2005)*, pages 773–780, Piscataway, NJ, Sept. 2005. IEEE Press.

L. C. T. Bezerra, M. López-Ibáñez, and T. Stützle. A large-scale experimental evaluation of high-performing multi- and many-objective evolutionary algorithms. Technical Report TR/IRIDIA/2017-005, IRIDIA, Université Libre de Bruxelles, Belgium, Brussels, Belgium, Feb. 2017.

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

J. Dubois-Lacoste, M. López-Ibáñez, and T. Stützle. Anytime Pareto local search. *European Journal of Operational Research*, 243(2):369–385, 2015. doi: 10.1016/j.ejor.2014.10.062.

D. Fanelli. Negative results are disappearing from most disciplines and countries. *Scientometrics*, 90(3):891–904, 2012. doi: 10.1007/s11192-011-0494-7.

M. Fischetti and M. Monaci. Exploiting erraticism in search. *Operations Research*, 62(1):114–122, 2014. doi: 10.1287/opre.2013.1231.

I. P. Gent, S. A. Grant, E. MacIntyre, P. Prosser, P. Shaw, B. M. Smith, and T. Walsh. How not to do it. Technical Report 97.27, School of Computer Studies, University of Leeds, May 1997.

J. N. Hooker. Testing heuristics: We have it all wrong. *Journal of Heuristics*, 1(1):33–42, 1996.

F. Hutter, H. H. Hoos, and T. Stützle. Automatic algorithm configuration based on local search. In R. C. Holte and A. Howe, editors, *Proc. of the Twenty-Second Conference on Artifical Intelligence (AAAI '07)*, pages 1152–1157. AAAI Press/MIT Press, Menlo Park, CA, 2007.

References II

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 *Lecture Notes in Computer Science*, pages 507–523. Springer, Heidelberg, Germany, 2011.

D. S. Johnson. A theoretician's guide to the experimental analysis of algorithms. In M. H. Goldwasser, D. S. Johnson, and C. C. McGeoch, editors, *Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implementation Challenges*, pages 215–250. American Mathematical Society, Providence, RI, 2002.

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/IridiaTr2011-004.pdf.

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