Automated Algorithm Configuration and Design

Thomas Stützle

stuetzle@ulb.ac.be
http://iridia.ulb.ac.be/~stuetzle

IRIDIA, CoDE, ULB,
Brussels, Belgium

Manuel López-Ibáñez★

manuel.lopez-ibanez@uma.es
http://lopez-ibanez.eu

University of Málaga, Spain

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

GECCO '21 Companion, July 10–14, 2021, Lille, France

© 2021 Copyright is held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-1-4503-8351-6/21/07.
https://doi.org/10.1145/3449726.3461404

Thomas Stützle is a senior research associate of the Belgian F.R.S.-FNRS working at the IRIDIA laboratory of Université libre de Bruxelles (ULB), Belgium. He received the Diplom (German equivalent of M.S. degree) in business engineering from the Universität Karlsruhe (TH), Karlsruhe, Germany in 1994, and his PhD and his habilitation in computer science both from the Computer Science Department of Technische Universität Darmstadt, Germany, in 1998 and 2004, respectively. He is the co-author of two books about "Stochastic Local Search: Foundations and Applications" and "Ant Colony Optimization". His 2002 GECCO paper "A Racing Algorithm For Configuring Metaheuristics" (joint work with M. Birattari, L. Paquete, and K. Varrentrapp) received the 2012 SIGEVO impact award.

Manuel López-Ibáñez is a "Beatriz Galindo" Senior Distinguished Researcher at University of Málaga since 2020 and a Senior Lecturer (Assistant Professor) at the Alliance Manchester Business School, University of Manchester, UK, since 2015. He received the M.S. degree in computer science from the University of Granada, Granada, Spain, in 2004, and the Ph.D. degree from Edinburgh Napier University, U.K., in 2009. Between 2011 and 2015, he was a Postdoctoral Researcher of the Belgian F.R.S.-FNRS at the IRIDIA laboratory in the Université Libre de Bruxelles (ULB), Brussels, Belgium. http://lopez-ibanez.eu

Part I

Automatic Algorithm Configuration (Overview)

Solving complex optimization problems

The algorithmic solution of hard optimization problems is one of the CS/OR success stories!

- Exact algorithms (systematic search)
  - branch&bound, branch&cut, constraint programming, ...
  - ✔ guarantee of optimality
  - ✘ but often time/memory consuming
  - powerful general-purpose software
- Approximation algorithms
  - heuristics, local search, metaheuristics, hyperheuristics ...
  - ✘ rarely provable guarantees
  - ✔ but often fast and accurate
  - typically special-purpose software

Very active research on hybrids of exact/approximate algorithms!

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
  - CPLEX: 63 parameters that influence search
  - Gurobi (25), lpsolve (47), SCIP (more than 200)
- (Meta)-heuristic solvers
  - Design choices: solution representation, operators, neighborhoods, pre-processing, strategies, . . . + numerical parameters
  - MOACO framework (22 parameters, see part 2),

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]
AI Planning	LGP,	45 – 66	[Vallati et al., 2011]
	Fast-downward		[Fawcett et al., 2011]
Code optimization	GCC	172 flags + 195 numerical	[Pérez Cáceres et al., 2017b]

Design choices and parameters everywhere

- *categorical* parameters **design**

  recombination ∈ { uniform, one-point, two-point }
  localsearch ∈ { tabu search, SA, ILS }
- *ordinal* parameters **design**

  neighborhoods ∈ { small, medium, large }
- *numerical* parameters ***tuning, calibration***
  - population sizes, mutation rates, acceptance temperature, hidden constants, . . .
- Parameters may be *conditional* to specific values of other parameters:

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

Configuring algorithms involves setting categorical, ordinal and numerical 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

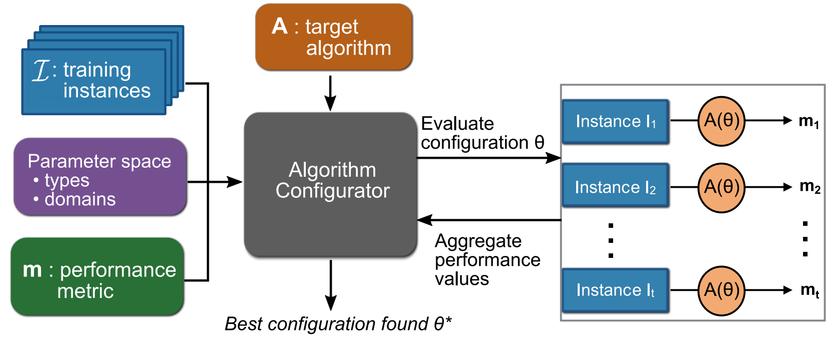
Can we make this approach more principled and automatic?

Towards automatic algorithm configuration

Automatic algorithm configuration

- apply powerful search techniques to design algorithms
- use computation power to explore algorithm design spaces
- free human creativity for higher level tasks

(Offline) Automatic Algorithm Configuration



Typical tuning goals

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

Offline configuration vs. Online control

Offline tuning / Algorithm configuration

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

Online tuning / Parameter control / Reactive search

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

All online methods have parameters that are configured offline

Methods for Automatic Algorithm Configuration

experimental design, ANOVA

CALIBRA [Adenso-Díaz & Laguna, 2006]
others [Coy et al., 2001; Ridge & Kudenko, 2007; Ruiz & Maroto, 2005]

numerical optimization

MADS [Audet & Orban, 2006], CMA-ES, BOBYQA [Yuan et al., 2012]

heuristic optimization

meta-GA [Grefenstette, 1986], ParamILS [Hutter et al., 2007, 2009], gender-based GA [Ansótegui et al., 2009], linear GP [Oltean, 2005], REVAC(++) [Nannen & Eiben, 2006; Smit & Eiben, 2009, 2010] . . .

model-based

SPO [Bartz-Beielstein et al., 2005, 2010], SMAC [Hutter et al., 2011], mlrMBO [Bischl et al., 2017]

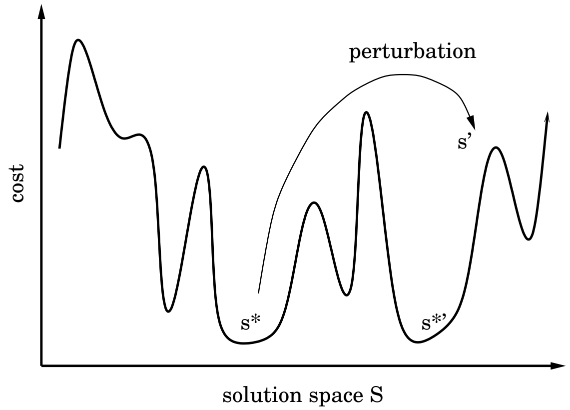
sequential statistical testing

F-race, iterated F-race [Balaprakash et al., 2007; Birattari et al., 2002] irace [López-Ibáñez et al., 2011]

ParamILS Framework

[Hutter et al., 2007, 2009]

ParamILS is an iterated local search method that works in the parameter space



Main design choices for ParamILS

Main design choices

- Parameter encoding: only categorical parameters, numerical parameters need to be discretized
- Initialization: select best configuration among default and several random configurations
- Local search: 1-exchange neighborhood, where exactly one parameter changes a value at a time, is searched in random order
- Perturbation: change several randomly chosen parameters
- Acceptance criterion: always select the better configuration

Main design choices for ParamILS

Evaluation of incumbent

- *BasicILS*: each configuration is evaluated on the same number of N instances
- *FocusedILS*: the number of instances on which the best configuration is evaluated increases at run time (intensification)

Adaptive Capping

- mechanism for early pruning the evaluation of poor candidate configurations
- particularly effective when configuring algorithms for minimization of computation time

Model-based Approaches (SPOT, SMAC)

Idea: Use surrogate models to predict performance

Algorithmic scheme

```
generate and evaluate initial set of configurations Θ_0
choose best-so-far configuration θ* ∈ Θ_0
while tuning budget available do
    learn surrogate model M : Θ ↦ ℝ
    generate set of possible candidate configurations Θ
    use model M to filter promising configurations Θ_p ⊆ Θ
    evaluate configurations in Θ_p
    Θ_0 ← Θ_0 ∪ Θ_p
    update θ* ∈ Θ_0
output: θ*
```

Sequential parameter optimization (SPO) toolbox

[Bartz-Beielstein et al., 2005, 2010]

Main design decisions

- Gaussian stochastic processes for $\mathcal{M}$ (in most variants)
- Expected improv. criterion (EIC) $\Rightarrow$ promising configurations
- Intensification mechanism $\Rightarrow$ increase num. of evals. of θ^*

Practicalities

- SPO is implemented in the comprehensive SPOT R package
- Most applications to numerical parameters on one instance
- SPOT includes various analysis and visualization tools

Sequential model-based algorithm configuration (SMAC)

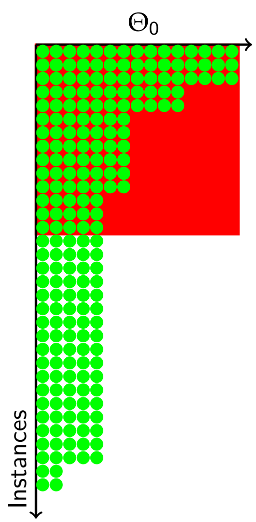
[Hutter et al., 2011]

SMAC extends surrogate model-based configuration to complex algorithm configuration tasks and across multiple instances

Main design decisions

- Random forests for $\mathcal{M}$ $\Rightarrow$ categorical & numerical parameters
- Aggregate predictions from $\mathcal{M}_i$ for each instance i
- Local search on the surrogate model surface (EIC) $\Rightarrow$ promising configurations
- Instance features $\Rightarrow$ improve performance predictions
- Intensification mechanism (inspired by FocusedILS)
- Further extensions $\Rightarrow$ adaptive capping

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



- 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

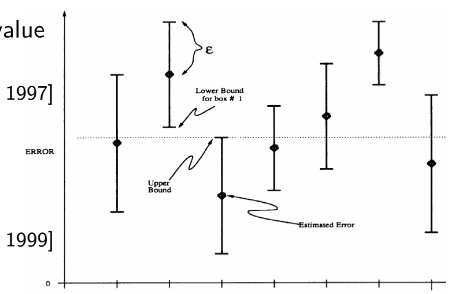
The racing approach [Birattari et al., 2002; Maron & Moore, 1997]

How to discard?

Successive Halving: discard 50% worst [Becker et al., 2005; Li et al., 2018]

or statistical testing:

- 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]



Taken from Maron & Moore [1997]

Sampling configuration

Racing (SH, 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?

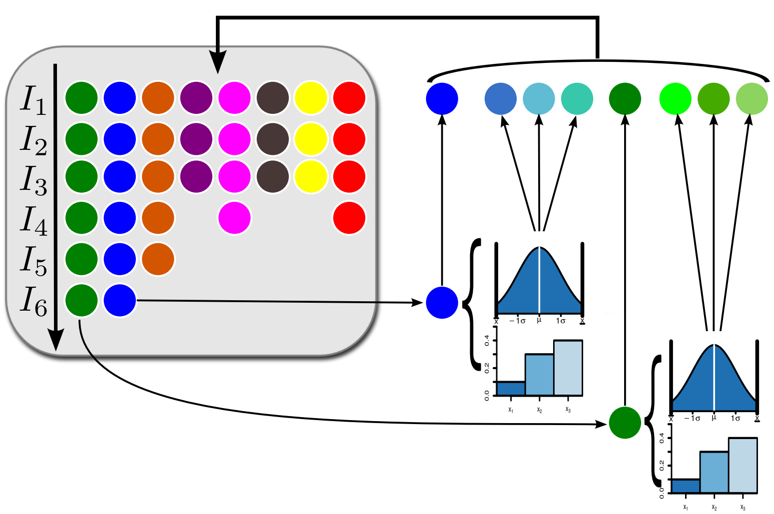
- Full factorial
- Random sampling
- Iterative update of a probabilistic sampling model ($\approx$ EDA)
  $\Rightarrow$ *Iterated F-Race (I/F-Race)* [Balaprakash et al., 2007]

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

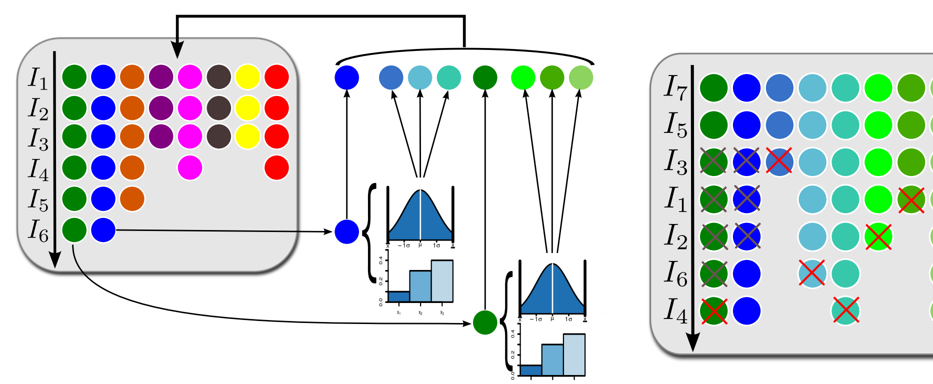
I/F-race: Balaprakash, Birattari, and Stützle [2007], Birattari, Yuan, Balaprakash, and Stützle [2010]
irace: López-Ibáñez, Dubois-Lacoste, Stützle, and Birattari [2011]
elitist irace: López-Ibáñez, Dubois-Lacoste, Pérez Cáceres, Stützle, and Birattari [2016]
elitist irace + adaptive capping:
Pérez Cáceres, López-Ibáñez, Hoos, and Stützle [2017a]

Iterated Racing



Elitist Iterated Racing [López-Ibáñez et al., 2016]

✘ Each new iteration (race) forgets the results of the previous one
$\Rightarrow$ Iterated F-race may "*lose*" the best-so-far configuration
✔ Protect the best configurations (*elites*) from being discarded unless all their results are considered



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

- Implementation of Iterated Racing in R

  Goal 1: Flexible

  Goal 2: Easy to use

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

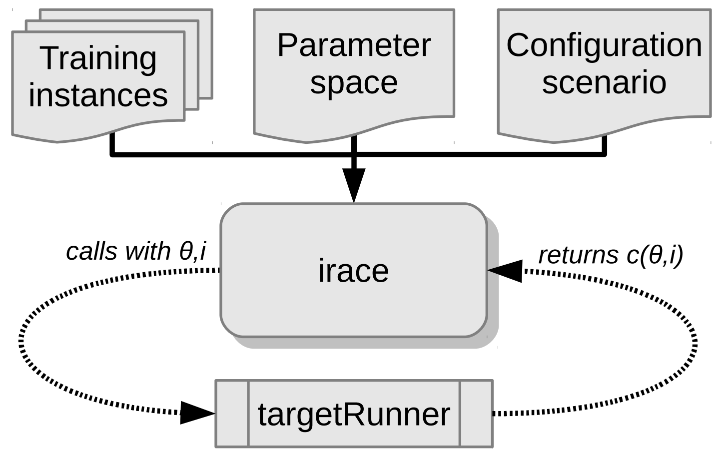
  http://cran.r-project.org/package=irace

- 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)
- *digits*: number of decimal places to be considered for the real parameters (default: 4)
- *testType*: either F-test or t-test
- *firstTest*: specifies how many instances are seen before the first test is performed (default: 5)
- *eachTest*: specifies how many instances are seen between tests (default: 1)

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: MPI, multiple cores, batch job clusters (SGE, PBS, Torque, Slurm)
3. Forbidden configurations:

```
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 3.4.1 (31/03/2020)

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

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
  - anytime optimization (improve time-quality trade-offs)
  - command-line flags of GCC compiler [Pérez Cáceres et al., 2017b]
- Automatic algorithm design
  - From a flexible framework of algorithm components
  - From a grammar description
- Machine learning
  - Automatic model selection for survival analysis [Lang et al., 2014]
  - **mlr** R package [Bischl et al., 2013, 2016]
- Design of control software for robots [Francesca et al., 2015]
- *Theoretical research [Dang & Doerr, 2019; Friedrich et al., 2018]*

1086 citations in Google Scholar, 81 000 downloads

An overview of applications of irace

irace (and other AC methods) works great for

- Complex parameter spaces:
  numerical, categorical, ordinal, subordinate (conditional)
- Large parameter spaces (few hundred parameters)
- Heterogeneous instances
- Medium to large tuning budgets (thousands of target runs)
- Target algorithm runs require from seconds to hours
- Minimize the runtime of target algorithms
- Multi-core CPUs, MPI, Grid-Engine clusters

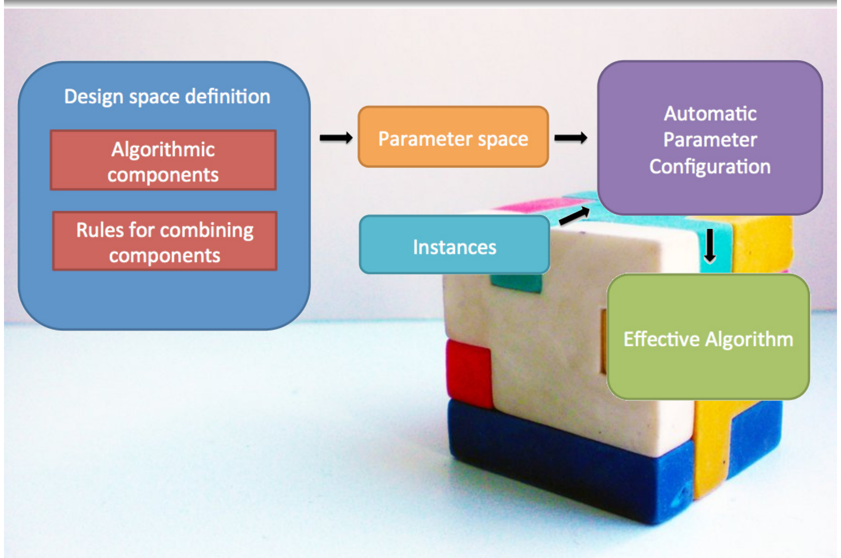
Thomas Stützle and Manuel López-Ibáñez Automated Algorithm Configuration and Design

Part II

Automated Algorithm Design

Thomas Stützle and Manuel López-Ibáñez Automated Algorithm Configuration and Design

Automatic Algorithm Design: General approach



Thomas Stützle and Manuel López-Ibáñez Automated Algorithm Configuration and Design

Main approaches for automatic algorithm design

Top-down approaches

- *Fixed algorithm template* with fixed alternatives for template parameters
- Examples:

  MIP solvers: *CPLEX* [Hutter et al., 2010], *SCIP* [López-Ibáñez & Stützle, 2014]
  SATenstein [KhudaBukhsh et al., 2009]
  MOACO framework [López-Ibáñez & Stützle, 2012]
  AutoMOEA [Bezerra, López-Ibáñez, and Stützle, 2016]

Bottom-up approaches

- Library of algorithm components
- Rules for composing algorithms from components, e.g., via *grammars*
- Examples:

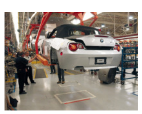
  GP + trees [Vázquez-Rodríguez & Ochoa, 2010]
  GE + grammars [Burke et al., 2012]
  irace + grammars [Mascia et al., 2014]
  [De Souza & Ritt, 2018]

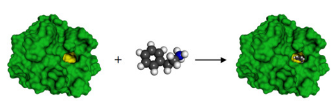
Example #1

AutoMOEA

Multi-objective Evolutionary Algorithms

- +30 years of research
- Most researched MO metaheuristic
- Real-world applications in many domains



- Numerous high-quality libraries/frameworks: Shark, jMetal, PaGMO, . . .

MOEAs: Which one?

- MOGA [Fonseca & Fleming, 1993]
- PAES [Knowles & Corne, 2000]
- NSGA-II [Deb et al., 2002]
- SPEA2 [Zitzler et al., 2002]
- IBEA [Zitzler & Künzli, 2004]
- SMS-EMOA [Beume et al., 2007]
- MO-CMA-ES [Igel et al., 2007]
- MOEA/D [Li & Zhang, 2009]
- HypE [Bader & Zitzler, 2011]
- NSGA-III [Deb & Jain, 2014]
- GDE3 [Kukkonen & Lampinen, 2005]
- DEMO [Robič & Filipič, 2005]
- DEMO^{SP2}, DEMO^{IB} [Tušar & Filipič, 2007]
- Indicator-based Differential Evolution [Tagawa et al., 2011]
- Genetic Diversity Evolutionary Algorithm (GDEA)
- Δ_p-Differential Evolution (DDE)
- neighbourhood exploring evolution strategy (NEES)
- OPTIMOGA
- Biogeography-based multi-objective evolutionary algorithm (BBMOEA)

AutoMOEA

- ✔ Replicate as many well-known MOEAs as possible from the same *template*
- ✔ The template has a number of configurable algorithmic *components*
- ✔ Each component can be configured by choosing one *option* from various alternatives
- ✔ Aim to maximise the number of different configurations that are valid MOEAs

Leonardo C. T. Bezerra, Manuel López-Ibáñez, and Thomas Stützle. **Automatic Component-Wise Design of Multi-Objective Evolutionary Algorithms.** *IEEE Transactions on Evolutionary Computation*, 2016.

Leonardo C. T. Bezerra, Manuel López-Ibáñez, and Thomas Stützle. **Automatically Designing State-of-the-Art Multi- and Many-Objective Evolutionary Algorithms.** *Evolutionary Computation*, 2020.

AutoMOEA: A MOEA template

```
pop ← Initialization ()
if type(arch_ext) != none then
   arch_ext ← pop
repeat
   pool ← BuildMatingPool (pop)
   pop_new ← Variation (pool)
   pop_new ← Evaluation (pop_new)
   pop     ← Replacement (pop, pop_new)
   if type(arch_ext) == bounded then
      arch_ext ← Archiving (arch_ext, pop_new)
   else if type(arch_ext) == unbounded then
      arch_ext ← arch_ext ∪ pop
until termination criteria met
if type(arch_ext) == none then
   return pop
else
   return arch_ext
```


AutoMOEA: Main components

Component	Parameters
BuildMatingPool	$\langle$ Preference$_{Mat}$, Selection $\rangle$
Replacement	$\langle$ Preference$_{Rep}$, Removal $\rangle$
Archiving	$\langle$ Preference$_{Ext}$, Removal$_{Ext}$ $\rangle$
Preference	$\langle$ Fitness, Diversity $\rangle$

Algorithm	Fitness	Diversity
NSGA-II	dominance depth	crowding distance
SPEA2	dom. strength	kNN
IBEA	binary indicator	
HypE	I_H^h	
SMS-EMOA	dom. depth-rank	I_H^1

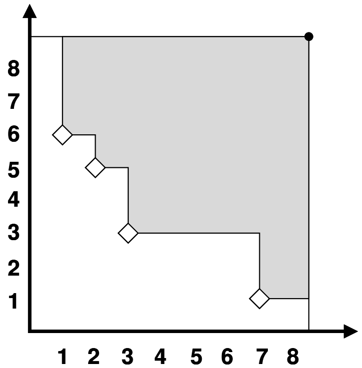
AutoMOEA: Redesign of main components [Zitzler et al., 2010]

Component	Parameters
BuildMatingPool	$\langle$ Preference$_{Mat}$, Selection $\rangle$
Replacement	$\langle$ Preference$_{Rep}$, Removal $\rangle$
Archiving	$\langle$ Preference$_{Ext}$, Removal$_{Ext}$ $\rangle$
Preference	~~$\langle$ Fitness, Diversity $\rangle$~~
	$\langle$ Set-partitioning, Quality, Diversity $\rangle$

	BuildMatingPool			Replacement		
	SetPart	Quality	Diversity	SetPart	Quality	Diversity
MOGA	dom. rank	—	niche-sharing	—	—	—
NSGA-II	dom. depth	—	crowding dist.	dom. depth	—	crowding dist.
SPEA2	dom. strength	—	kNN	dom. strength	—	kNN
IBEA	—	binary indicator	—	—	binary ind.	—
HypE	—	I_H^h	—	dom. depth	I_H^h	—
SMS-EMOA	—	—	—	dom. depth-rank	I_H^1	—

Automatic configuration of MO algorithms

✘ Multi-objective!
Output is a set, an approximation to the Pareto front!



irace + hypervolume = automatic configuration
of multi-objective solvers!

AutoMOEA: Automatic Design

Automatic configuration (**irace**)
+ Flexible algorithmic framework (AutoMOEA)
= Automatic design of state-of-the-art MOEAs

	BuildMatingPool			Replacement		
	SetPart	Quality	Diversity	SetPart	Quality	Diversity
MOGA	rank	—	niche-sharing	—	—	—
NSGA-II	depth	—	crowding dist.	depth	—	crowding dist.
SPEA2	strength	—	kNN	strength	—	kNN
IBEA	—	binary indicator	—	—	binary ind.	—
HypE	—	I_H^h	—	depth	I_H^h	—
SMS-EMOA	—	—	—	depth-rank	I_H^1	—
DTLZ 2-obj	—	—	crowding	depth-rank	I_ϵ	sharing
DTLZ 3-obj	depth-rank	I_ϵ	kNN	rank	I_H^1	sharing
DTLZ 5-obj	rank	I_H^1	crowding	depth	I_H^1	—
WFG 2-obj	rank	—	crowding	depth-rank	I_H^1	—
WFG 3-obj	count	I_H^1	crowding	strength	I_H^1	sharing
WFG 5-obj	count	I_H^h	crowding	—	I_H^1	—

Experimental results

DTLZ			WFG		
2-obj $\Delta R = 126$	3-obj $\Delta R = 127$	5-obj $\Delta R = 107$	2-obj $\Delta R = 169$	3-obj $\Delta R = 130$	5-obj $\Delta R = 97$
$\texttt{Auto}_{D2}$ **(1339)**	$\texttt{Auto}_{D3}$ **(1500)**	$\texttt{Auto}_{D5}$ **(1002)**	$\texttt{Auto}_{W2}$ **(1692)**	$\texttt{Auto}_{W3}$ **(1375)**	$\texttt{Auto}_{W5}$ **(1170)**
SPEA2_{D2} (1562)	IBEA_{D3} (1719)	SMS_{D5} (1550)	SPEA2_{W2} (2097)	SMS_{W3} (1796)	SMS_{W5} (1567)
IBEA_{D2} (1940)	SMS_{D3} (1918)	IBEA_{D5} (1867)	NSGA-II_{W2} (2542)	IBEA_{W3} (1843)	IBEA_{W5} (1746)
NSGA-II_{D2} (2143)	HypE_{D3} (2019)	SPEA2_{D5} (2345)	SMS_{W2} (2621)	SPEA2_{W3} (2600)	SPEA2_{W5} (2747)
HypE_{D2} (2338)	SPEA2_{D3} (2164)	NSGA-II_{D5} (2346)	IBEA_{W2} (2777)	NSGA-II_{W3} (3315)	NSGA-II_{W5} (3029)
SMS_{D2} (2406)	NSGA-II_{D3} (2528)	HypE_{D5} (2674)	HypE_{W2} (2851)	HypE_{W3} (3431)	MOGA_{W5} (4268)
MOGA_{D2} (2970)	MOGA_{D3} (2851)	MOGA_{D5} (2915)	MOGA_{W2} (4320)	MOGA_{W3} (4540)	HypE_{W5} (4373)

AutoMOEA: Automatic Design

Automatic configuration (**irace**)
+ Flexible algorithmic framework (AutoMOEA)
= Automatic design of state-of-the-art MOEAs

- Fair to compare with untuned traditional MOEAs?
- Why is our setup representative?
  - ⇒ Different AutoMOEAs for termination criterion in FEs or seconds
- How do you define "state-of-the-art"?
- What is a "novel" MOEA?

Exactly!

Example #2

From Grammars to Parameters:
How to use irace to design algorithms
from a grammar description?

Automatic design of hybrid SLS algorithms

[Marmion, Mascia, López-Ibáñez, and Stützle, 2013]

1. *Decompose* single-point LS methods into components
2. Devise generalized LS (GLS) meta-metaheuristic
3. Describe possible GLS instantiations by a *grammar*
4. Convert the grammar to a *parametric representation*
   - ✔ Allows use of standard automatic configuration tools (*irace*)
   - ✔ Shows good performance when compared to grammatical evolution [Mascia, López-Ibáñez, Dubois-Lacoste, and Stützle, 2014]

Marie-Eléonore Marmion, Franco Mascia, Manuel López-Ibáñez, and Thomas Stützle.
Automatic Design of Hybrid Stochastic Local Search Algorithms.
In *Hybrid Metaheuristics*, vol. 7919 of LNCS, 2013.

Franco Mascia, Manuel López-Ibáñez, Jérémie Dubois-Lacoste, and Thomas Stützle.
Grammar-based Generation of Stochastic Local Search Heuristics Through Automatic Algorithm Configuration Tools. *Computers & Operations Research*, 2014.

Manuel López-Ibáñez, Marie-Eléonore Marmion, and Thomas Stützle.
Automatic Design of Hybrid Metaheuristics from Algorithmic Components.
TR/IRIDIA/2017-012, 2017.

Generalized Local Search

GLS Algorithm

$s_0 \leftarrow$ `Initialisation()`
`best_found` $\leftarrow$ **GLS(**s_0, `Perturb`, `Local_search`, `Acceptance`, `Stop`**)**
return `best_found`

Function GLS(s_0, `Perturb`, `Local_search`, `Acceptance`, `Stop`)

$s^* \leftarrow$ `Local_search`(s_0)
while not `Stop()` **do**
  $s' \leftarrow$ `Perturb`(s^*)
  $s'' \leftarrow$ `Local_search`(s')
  $s^* \leftarrow$ `Acceptance`(s'', s^*)
return s^*

- ✔ Many SLS methods may be instantiated from this template
- ✔ Hybridization when `Local_search` is another GLS

Classical SLS as instantiations of GLS

	Perturbation	Local Search	Acceptance
SA	One-move	$\emptyset$	Metropolis
PII	One-move	$\emptyset$	Metropolis (fixed temp.)
RII	One-move	$\emptyset$	Probabilistic
VNS	Variable-move	"any"	Variable-Accept
IG	Deconst-Construc	"any"	"any"
ILS	any	any	any
TS	$\emptyset$	TS-ls	Always

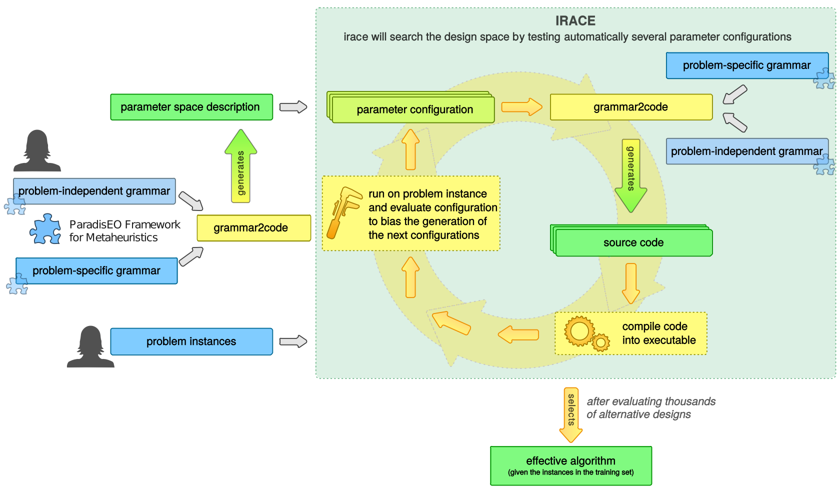
Grammar

```
       <start> ::= <Initialisation>; <GLS>; return best_found;
         <GLS> ::= <known_MH> | <Hybrid>
    <known_MH> ::= <ILS> | <SA> |<PII> | <RII> | <IG> | <VNS> | <TS>
      <Hybrid> ::= GLS( <perturbation>, <GLS>, <acceptance>, <stop> )
```

```
<ILS> ::= GLS(<perturbation>, <local_search>, <acceptance>)
<PII> ::= GLS(<pert_one_move>, LSNone, AcceptMetropolis(<temperature>))
 <SA> ::= GLS(<pert_one_move>, LSNone, <accept_metropolis>)
<RII> ::= GLS(<pert_one_move>, LSNone, AcceptProbabilistic( <probability> ))
<VNS> ::= GLS(<pert_VNS>, <local_search>, <accept_VNS>)
 <IG> ::= GLS(<ps:pert_destruct_construct>, <local_search>, <acceptance>)
 <TS> ::= GLS(PertNone, <ls_TS>, AcceptAlways)
```

```
       <accept_VNS> ::= AcceptBetter | <accept_skewed>
         <pert_VNS> ::= <ps:pert_repeatable> (<pert_strength_dyn_incr>)
     <perturbation> ::= PertNone | PertRestart | <pert_one_move>
                      | <pert_k_moves> | <pert_variable>
     <pert_k_moves> ::= <ps:pert_repeatable> (<pert_strength_value>)
    <pert_variable> ::= <ps:pert_repeatable> (<pert_strength_schedule>)
     <local_search> ::= LSNone| <ls_first_imp> | <ls_best_imp > | <...>
       <acceptance> ::= AcceptAlways | AcceptBetter | AcceptBetterEqual
                      | <accept_probab> | <accept_threshold> | <accept_metropolis>
<accept_metropolis> ::= AcceptMetropolis(<temp_init>, <temp_factor>, <temp_final>,
                                         <temp_reheat_mode>)
```

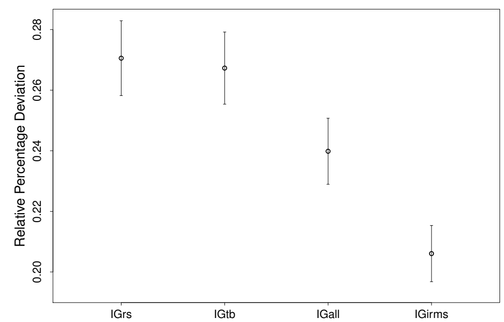

System overview [Marmion, Mascia, López-Ibáñez, and Stützle, 2013]



Flow-shop problem with makespan objective

[Pagnozzi and Stützle, 2019]

- **IGrs** [Ruiz & Stützle, 2007]
- **IGtb**: IGrs + tie-breaking rule by [Fernandez-Viagas & Framiñán, 2014]
- **IGall**: IG + LS on partial solutions [Dubois-Lacoste et al., 2017]
- **IGirms**: automatically designed by irace
  - max. three levels of recursion
  - biased / unbiased grammar resulting in 262 and 502 parameters, respectively
  - budget: 200 000 runs of $0.03 \cdot n \cdot m$ CPU-sec

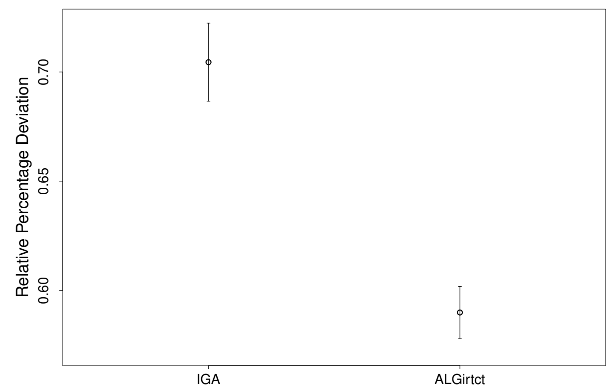


Results are clearly superior to state-of-the-art

Flow-shop problem with total completion time objective

[Pagnozzi and Stützle, 2019]

- **IGA** [Pan & Ruiz, 2012]
- **ALGirtct**: automatically designed by irace
- Automatic configuration:
  - max. three levels of recursion
  - budget: 200 000 runs of $0.03 \cdot n \cdot m$ sec

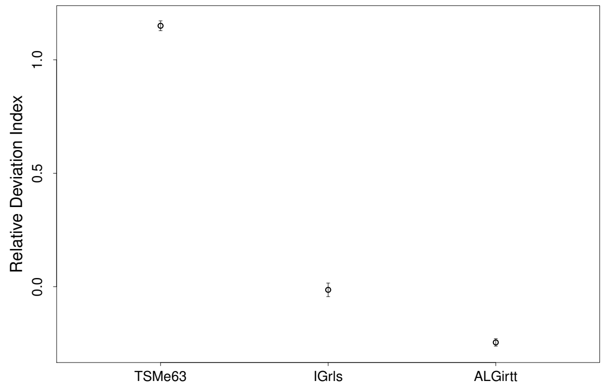


Results are clearly superior to state-of-the-art

Flow-shop problem with total tardiness objective

[Pagnozzi and Stützle, 2019]

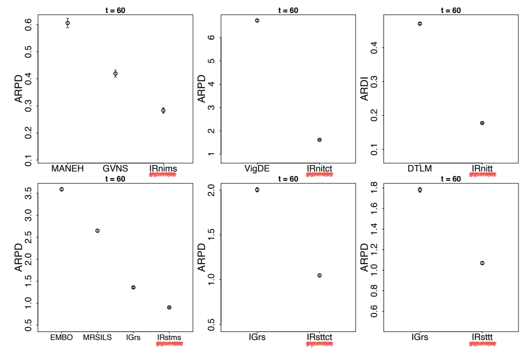
- **TSMe63**: Trajectory Scheduling Method [Li et al., 2015]
- **IGrls**: [Karabulut, 2016]
- **ALGirtt**: automatically designed by irace
  - max. three levels of recursion
  - budget: 200 000 runs of $0.03 \cdot n \cdot m$ sec



Results are clearly superior to state-of-the-art

Additional constraints

- we tackled the same three objectives with no-idle and sequence dependent setup times



Results are clearly superior to state-of-the-art

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

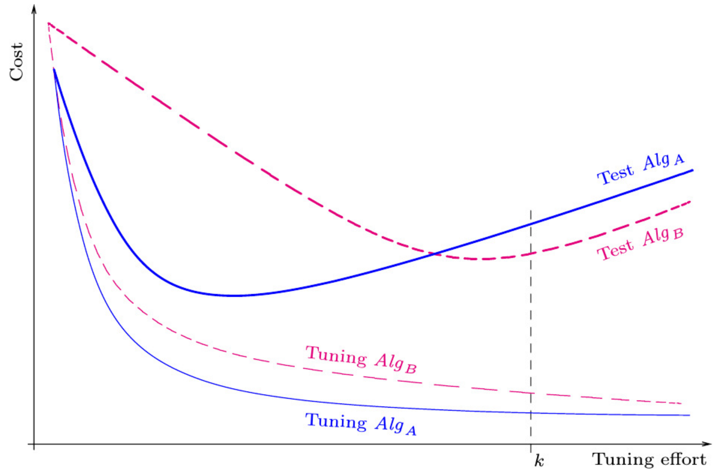
Reason #1: More scientific, more principled

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

> “*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.* [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":

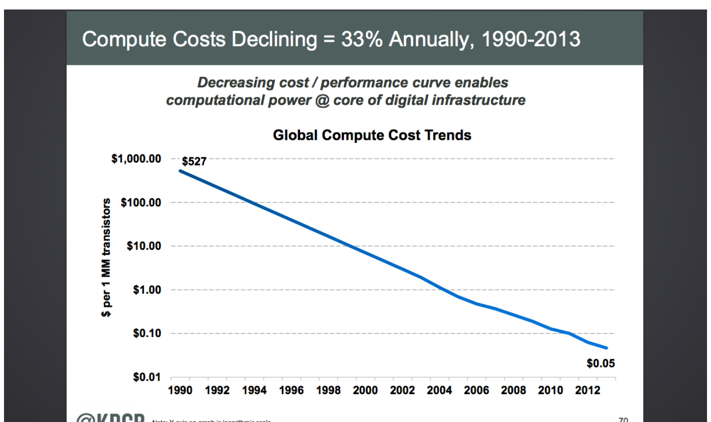
  problem modeling + algorithmic components + computing power

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

Reason #3: Computing power is exponentially cheaper

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

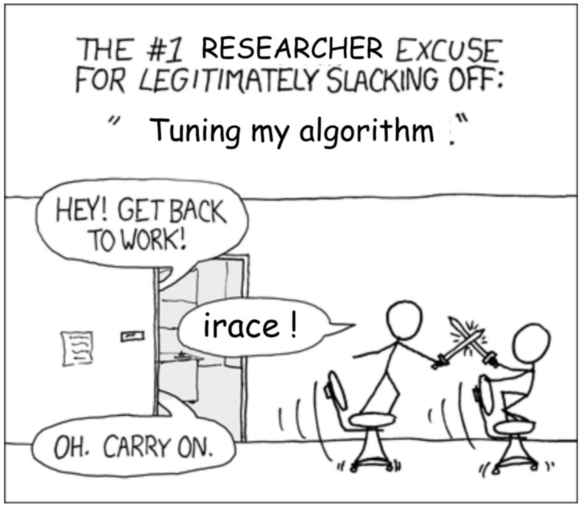
Reason #5: More interesting, fun, and useful

- ✘ Classical optimization research:
  1. Human-driven design to outperform other algorithmic designs
  2. Analysis of the human-designed algorithm
- ✔ 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

The End



Based on https://xkcd.com/303/

Acknowledgments

The tutorial has benefited from collaborations and discussions with our colleagues:

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, Leslie Pérez Cáceres, Zhi Yuan, Marcus Ritt, Marcelo De Souza

The research leading to the results presented here has received funding from diverse projects:

- Thomas Stützle acknowledges support of the F.R.S.-FNRS of which he is a senior research associate.
- M. López-Ibáñez is a "Beatriz Galindo" Senior Distinguished Researcher (BEAGAL 18/00053) funded by the Spanish Ministry of Science and Innovation (MICINN).
- European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement nº 246939
- PAI project COMEX funded by the Interuniversity Attraction Poles Programme of the Belgian Science Policy Office
- and the EU FP7 ICT Project COLOMBO, Cooperative Self-Organizing System for Low Carbon Mobility at Low Penetration Rates (agreement no. 318622)

References I

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

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

C. Ansótegui, M. Sellmann, and K. Tierney. A gender-based genetic algorithm for the automatic configuration of algorithms. In 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.

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

J. Bader and E. Zitzler. HypE: An algorithm for fast hypervolume-based many-objective optimization. *Evolutionary Computation*, 19(1):45–76, 2011. doi: 10.1162/EVCO_a_00009.

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.

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.

T. Bartz-Beielstein, C. Lasarczyk, and M. Preuss. The sequential parameter optimization toolbox. In T. Bartz-Beielstein, M. Chiarandini, L. Paquete, and M. Preuss, editors, *Experimental Methods for the Analysis of Optimization Algorithms*, pages 337–360. Springer, Berlin, Germany, 2010.

S. Becker, J. Gottlieb, and T. Stützle. Applications of racing algorithms: An industrial perspective. In E.-G. Talbi, P. Liardet, P. Collet, E. Lutton, and M. Schoenauer, editors, *Artificial Evolution*, volume 3871 of *Lecture Notes in Computer Science*, pages 271–283. Springer, Heidelberg, Germany, 2005.

N. Beume, B. Naujoks, and M. T. M. Emmerich. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 2007. doi: 10.1016/j.ejor.2006.08.008.

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.

References II

L. C. T. Bezerra, M. López-Ibáñez, and T. Stützle. Automatically designing state-of-the-art multi- and many-objective evolutionary algorithms. *Evolutionary Computation*, 28(2):195–226, 2020a. doi: 10.1162/evco_a_00263.

L. C. T. Bezerra, M. López-Ibáñez, and T. Stützle. Automatic configuration of multi-objective optimizers and multi-objective configuration. In T. Bartz-Beielstein, B. Filipič, P. Korošec, and E.-G. Talbi, editors, *High-Performance Simulation-Based Optimization*, pages 69–92. Springer International Publishing, Cham, Switzerland, 2020b. doi: 10.1007/978-3-030-18764-4_4.

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, Germany, 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.

B. Bischl, J. Richter, J. Bossek, D. Horn, J. Thomas, and M. Lang. mlrMBO: A modular framework for model-based optimization of expensive black-box functions. *Arxiv preprint arXiv:1703.03373 [stat.ML]*, 2017. URL http://arxiv.org/abs/1703.03373.

A. Blot, H. H. Hoos, L. Jourdan, M.-E. Kessaci-Marmion, and H. Trautmann. MO-ParamILS: A multi-objective automatic algorithm configuration framework. In P. Festa, M. Sellmann, and J. Vanschoren, editors, *Learning and Intelligent Optimization, 10th International Conference, LION 10*, volume 10079 of *Lecture Notes in Computer Science*, pages 32–47. Springer, Cham, Switzerland, 2016.

E. K. Burke, M. R. Hyde, and G. Kendall. Grammatical evolution of local search heuristics. *IEEE Transactions on Evolutionary Computation*, 16(7):406–417, 2012. doi: 10.1109/TEVC.2011.2160401.

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

References III

S. P. Coy, B. L. Golden, G. C. Runger, and E. A. Wasil. Using experimental design to find effective parameter settings for heuristics. *Journal of Heuristics*, 7(1):77–97, 2001.

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 and M. Ritt. Automatic grammar-based design of heuristic algorithms for unconstrained binary quadratic programming. In A. Liefooghe and M. López-Ibáñez, editors, *Proceedings of EvoCOP 2018 – 18th European Conference on Evolutionary Computation in Combinatorial Optimization*, volume 10782 of *Lecture Notes in Computer Science*, pages 67–84. Springer, Heidelberg, Germany, 2018. doi: 10.1007/978-3-319-77449-7_5.

K. Deb and S. Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601, 2014.

K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002. doi: 10.1109/4235.996017.

J. Dubois-Lacoste, F. Pagnozzi, and T. Stützle. An iterated greedy algorithm with optimization of partial solutions for the permutation flowshop problem. *Computers & Operations Research*, 81:160–166, 2017. doi: 10.1016/j.cor.2016.12.021.

C. Fawcett, M. Helmert, H. H. Hoos, E. Karpas, G. Röger, and J. Seipp. FD-Autotune: Domain-specific configuration using fast-downward. In E. Karpas, S. Jiménez Celorrio, and S. Kambhampati, editors, *Proceedings of ICAPS-PAL11*, 2011.

V. Fernandez-Viagas and J. M. Framiñán. On insertion tie-breaking rules in heuristics for the permutation flowshop scheduling problem. *Computers & Operations Research*, 45:60–67, 2014.

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.

C. M. Fonseca and P. J. Fleming. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In S. Forrest, editor, *ICGA*, pages 416–423. Morgan Kaufmann Publishers, 1993. ISBN 1-55860-299-2.

References IV

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.

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

F. Hutter, H. H. Hoos, and T. Stützle. Automatic algorithm configuration based on local search. In R. C. Holte and A. Howe, editors, *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1152–1157. AAAI Press/MIT Press, Menlo Park, CA, 2007.

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, Oct. 2009. doi: 10.1613/jair.2861.

F. Hutter, H. H. Hoos, and K. Leyton-Brown. Automated configuration of mixed integer programming solvers. In A. Lodi, M. Milano, and P. Toth, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 7th International Conference, CPAIOR 2010*, volume 6140 of *Lecture Notes in Computer Science*, pages 186–202. Springer, Heidelberg, Germany, 2010. doi: 10.1007/978-3-642-13520-0_23.

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. doi: 10.1007/978-3-642-25566-3_40.

C. Igel, N. Hansen, and S. Roth. Covariance matrix adaptation for multi-objective optimization. *Evolutionary Computation*, 15(1): 1–28, 2007.

K. Karabulut. A hybrid iterated greedy algorithm for total tardiness minimization in permutation flowshops. *Computers and Industrial Engineering*, 98(Supplement C):300 – 307, 2016.

References V

A. R. KhudaBukhsh, L. Xu, H. H. Hoos, and K. Leyton-Brown. SATenstein: Automatically building local search SAT solvers from components. In C. Boutilier, editor, *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI-09)*, pages 517–524. AAAI Press, Menlo Park, CA, 2009.

J. D. Knowles and D. Corne. Approximating the nondominated front using the Pareto archived evolution strategy. *Evolutionary Computation*, 8(2):149–172, 2000. doi: 10.1162/106365600568167.

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.

S. Kukkonen and J. Lampinen. GDE3: the third evolution step of generalized differential evolution. In *Proceedings of the 2005 Congress on Evolutionary Computation (CEC 2005)*, pages 443–450. IEEE Press, Piscataway, NJ, Sept. 2005.

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.

H. Li and Q. Zhang. Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II. *IEEE Transactions on Evolutionary Computation*, 13(2):284–302, 2009.

L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52, 2018. URL http://jmlr.org/papers/v18/16-558.html.

X. Li, L. Chen, H. Xu, and J. N. Gupta. Trajectory scheduling methods for minimizing total tardiness in a flowshop. *Operations Research Perspectives*, 2:13–23, 2015. ISSN 2214-7160. doi: 10.1016/j.orp.2014.12.001.

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.

References VI

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. 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.-E. Marmion, F. Mascia, M. López-Ibáñez, and T. Stützle. Automatic design of hybrid stochastic local search algorithms. In M. J. Blesa, C. Blum, P. Festa, A. Roli, and M. Sampels, editors, *Hybrid Metaheuristics*, volume 7919 of *Lecture Notes in Computer Science*, pages 144–158. Springer, Heidelberg, Germany, 2013. ISBN 978-3-642-38515-5. doi: 10.1007/978-3-642-38516-2_12.

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

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.

M. Maur, M. López-Ibáñez, and T. Stützle. Pre-scheduled and adaptive parameter variation in $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System. In H. Ishibuchi et al., editors, *Proceedings of the 2010 Congress on Evolutionary Computation (CEC 2010)*, pages 3823–3830. IEEE Press, Piscataway, NJ, 2010. doi: 10.1109/CEC.2010.5586332.

P. Miranda, R. M. Silva, and R. B. Prudêncio. I/S-Race: An iterative multi-objective racing algorithm for the SVM parameter selection problem. In *European Symposium on Artificial Neural Networks, ESSAN*, pages 573–578, 2015.

V. Nannen and A. E. Eiben. A method for parameter calibration and relevance estimation in evolutionary algorithms. In M. Cattolico et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2006*, pages 183–190. ACM Press, New York, NY, 2006. doi: 10.1145/1143997.1144029.

A. J. Nebro, M. López-Ibáñez, C. Barba-González, and J. García-Nieto. Automatic configuration of NSGA-II with jMetal and irace. In M. López-Ibáñez, A. Auger, and T. Stützle, editors, *GECCO'19 Companion*, pages 1374–1381. ACM Press, New York, NY, 2019. ISBN 978-1-4503-6748-6. doi: 10.1145/3319619.3326832.

M. Oltean. Evolving evolutionary algorithms using linear genetic programming. *Evolutionary Computation*, 13(3):387–410, 2005. doi: 10.1162/1063656054794815.

References VII

F. Pagnozzi and T. Stützle. Automatic design of hybrid stochastic local search algorithms for permutation flowshop problems. *European Journal of Operational Research*, 276:409–421, 2019. doi: 10.1016/j.ejor.2019.01.018.

Q.-K. Pan and R. Ruiz. Local search methods for the flowshop scheduling problem with flowtime minimization. *European Journal of Operational Research*, 222(1):31–43, 2012.

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.

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.

E. Ridge and D. Kudenko. Tuning the performance of the MMAS heuristic. In T. Stützle, M. Birattari, and H. H. Hoos, editors, *Engineering Stochastic Local Search Algorithms. Designing, Implementing and Analyzing Effective Heuristics. SLS 2007*, volume 4638 of *Lecture Notes in Computer Science*, pages 46–60. Springer, Heidelberg, Germany, 2007.

T. Robič and B. Filipič. DEMO: Differential evolution for multiobjective optimization. In C. A. Coello Coello, A. H. Aguirre, and E. Zitzler, editors, *Evolutionary Multi-criterion Optimization, EMO 2005*, volume 3410 of *Lecture Notes in Computer Science*, pages 520–533. Springer, Heidelberg, Germany, 2005.

R. Ruiz and C. Maroto. A comprehensive review and evaluation of permutation flowshop heuristics. *European Journal of Operational Research*, 165(2):479–494, 2005.

R. Ruiz and T. Stützle. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 177(3):2033–2049, 2007.

S. K. Smit and A. E. Eiben. Comparing parameter tuning methods for evolutionary algorithms. In *Proceedings of the 2009 Congress on Evolutionary Computation (CEC 2009)*, pages 399–406. IEEE Press, Piscataway, NJ, 2009.

S. K. Smit and A. E. Eiben. Beating the 'world champion' evolutionary algorithm via REVAC tuning. In H. Ishibuchi et al., editors, *Proceedings of the 2010 Congress on Evolutionary Computation (CEC 2010)*, pages 1–8. IEEE Press, Piscataway, NJ, 2010. doi: 10.1109/CEC.2010.5586026.

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

References VIII

K. Tagawa, H. Shimizu, and H. Nakamura. Indicator-based differential evolution using exclusive hypervolume approximation and parallelization for multi-core processors. In N. Krasnogor and P. L. Lanzi, editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2011*, pages 657–664. ACM Press, New York, NY, 2011.

T. Tušar and B. Filipič. Differential evolution versus genetic algorithms in multiobjective optimization. In S. Obayashi et al., editors, *Evolutionary Multi-criterion Optimization, EMO 2007*, volume 4403 of *Lecture Notes in Computer Science*, pages 257–271. Springer, Heidelberg, Germany, 2007.

M. Vallati, C. Fawcett, A. E. Gerevini, H. H. Hoos, and A. Saetti. Generating fast domain-optimized planners by automatically configuring a generic parameterised planner. In E. Karpas, S. Jiménez Celorrio, and S. Kambhampati, editors, *Proceedings of ICAPS-PAL11*, 2011.

J. A. Vázquez-Rodríguez and G. Ochoa. On the automatic discovery of variants of the NEH procedure for flow shop scheduling using genetic programming. *Journal of the Operational Research Society*, 62(2):381–396, 2010.

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.

T. Zhang, M. Georgiopoulos, and G. C. Anagnostopoulos. S-Race: A multi-objective racing algorithm. In C. Blum and E. Alba, editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2013*, pages 1565–1572. ACM Press, New York, NY, 2013. ISBN 978-1-4503-1963-8.

T. Zhang, M. Georgiopoulos, and G. C. Anagnostopoulos. Multi-objective model selection via racing. *IEEE Transactions on Cybernetics*, 46(8):1863–1876, 2016.

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

E. Zitzler and S. Künzli. Indicator-based selection in multiobjective search. In X. Yao et al., editors, *Proceedings of PPSN-VIII, Eigth International Conference on Parallel Problem Solving from Nature*, volume 3242 of *Lecture Notes in Computer Science*, pages 832–842. Springer, Heidelberg, Germany, 2004.

E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. In K. C. Giannakoglou, D. T. Tsahalis, J. Periaux, K. D. Papaliliou, and T. Fogarty, editors, *Evolutionary Methods for Design, Optimisation and Control*, pages 95–100. CIMNE, Barcelona, Spain, 2002.

E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. Grunert da Fonseca. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003. doi: 10.1109/TEVC.2003.810758.

E. Zitzler, L. Thiele, and J. Bader. On set-based multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 14 (1):58–79, 2010. doi: 10.1109/TEVC.2009.2016569.

AClib: A Benchmark Library for Algorithm Configuration

F. Hutter, M. López-Ibáñez, C. Fawcett, M. Lindauer, H. H. Hoos, K. Leyton-Brown and T. Stützle. **AClib: a Benchmark Library for Algorithm Configuration**, Learning and Intelligent Optimization Conference (LION 8), 2014.

http://www.aclib.net/

- Standard benchmark for experimenting with configurators
- 326 heterogeneous scenarios
- SAT, MIP, ASP, time-tabling, TSP, multi-objective, machine learning
- Extensible ⇒ new scenarios welcome !

Scaling to expensive instances

What if my problem instances are too difficult/large?

- Cloud computing / Large computing clusters [Geschwender et al., 2014]

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

- J. Styles and H. H. Hoos. **Ordered racing protocols for automatically configuring algorithms for scaling performance.** GECCO, 2013

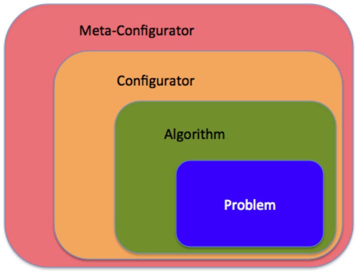
  Tune on easy instances,
  then ordered F-race on increasingly difficult ones

- F. Mascia, M. Birattari, and T. Stützle. **Tuning algorithms for tackling large instances: An experimental protocol.** Learning and Intelligent Optimization, LION 7, 2013.

  Tune on easy instances,
  then scale parameter values to difficult ones

Configuring configurators

What about configuring automatically the configurator?
. . . and configuring the configurator of the configurator?



✔ it can be done [Hutter et al., 2009] but . . .
✘ it is costly and iterating further leads to diminishing returns

"*It may be turtles all the way down, but the turtles get smaller.*"
– Audience member @ ECADA 2017

MO tuning vs. Tuning MO algorithms [Bezerra et al., 2020b]

Multi-objective AAC

- Multiple metrics to evaluate an algorithm configuration

  Optimization: solution quality, computation time, memory . . .
  ML: precision, recall, training time, prediction time . . .

- AAC produces mutually nondominated *set* of configurations

  S-Race [Zhang et al., 2013], I/S-Race [Miranda et al., 2015],
  SPRINT [Zhang et al., 2016], MO-ParamILS [Blot et al., 2016]

MO tuning vs. Tuning MO algorithms [Bezerra et al., 2020b]

Multi-objective AAC

- Multiple metrics to evaluate an algorithm configuration
- AAC produces mutually nondominated *set* of configurations

AAC for multi-objective algorithms

- Running a configuration outputs a *set* of mutually nondominated solutions (and/or *anytime* behavior)
- Unary quality metrics (hypervolume, epsilon, IGD+) evaluate the output [Zitzler et al., 2003]
- Uses *single-objective* AAC methods and produces a single best [Bezerra et al., 2016, 2020a; López-Ibáñez & Stützle, 2012; Nebro et al., 2019]

Multi-objective AAC of multi-objective algorithms is also possible!

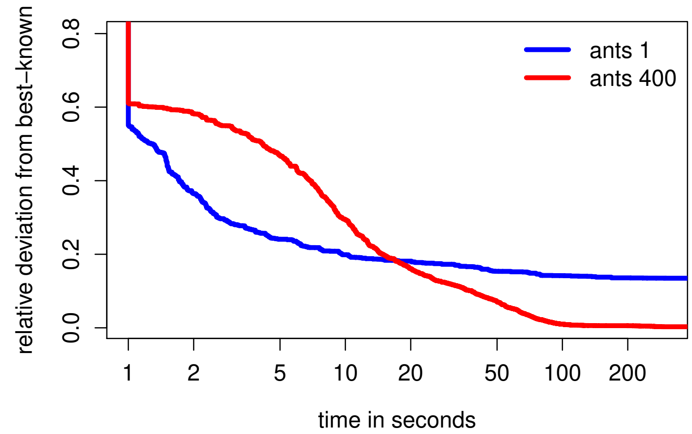
Example #3

Automatically Improving the Anytime Behavior of Optimization Algorithms with irace

Automatically Improving the Anytime Behavior

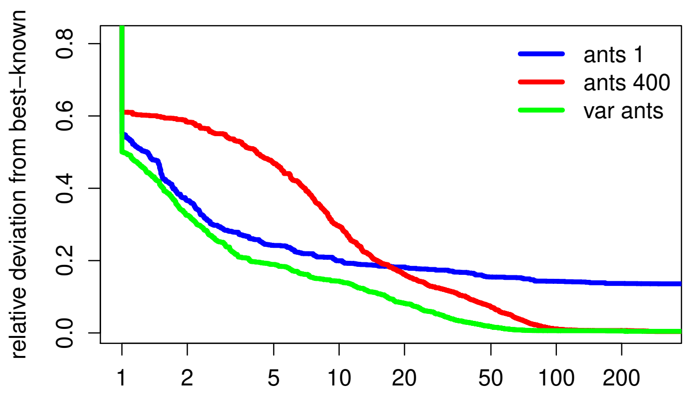
Max-Min Ant System w/o LS

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]



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?

Classical (Human-intensive) Approach

1. Devise *many* online strategies for parameter variation
2. Run lots of experiments
3. Visually compare SQT plots

After one year and a master thesis: [Maur et al., 2010]

- ✔ Strategies for varying *ants*, β, or q_0 that significantly improve the anytime behaviour of MMAS on the TSP.
- ✘ Extremely time consuming
- ✘ Subjective / Bias

Automatically Improving the Anytime Behavior

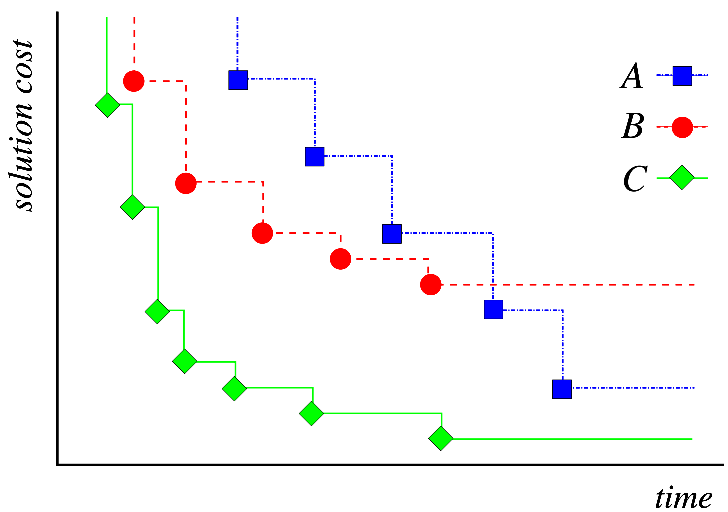
Online parameter control

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

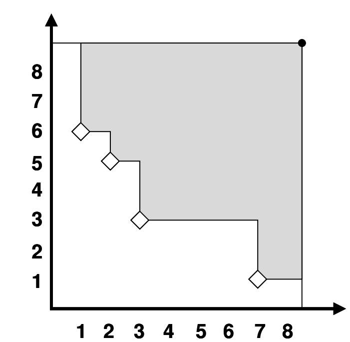
Improve Anytime Behavior

- ✔ More robust to different termination criteria
- ✘ How can irace compare SQT curves?

Automatically Improving the Anytime Behavior



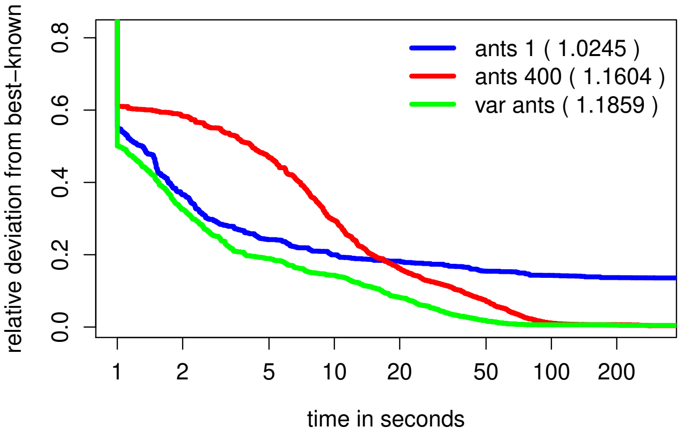
Automatically Improving the Anytime Behavior



Hypervolume measure ≈ Anytime behaviour

Manuel López-Ibáñez and Thomas Stützle.
Automatically improving the anytime behaviour of optimisation algorithms.
European Journal of Operational Research, 2014.

Automatically Improving the Anytime Behavior



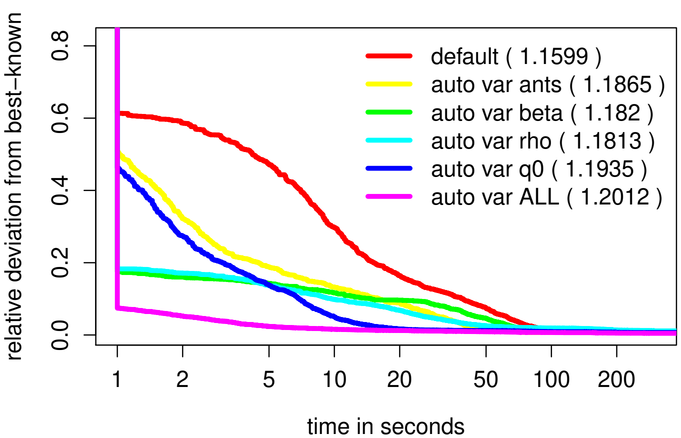
Automatically Improving the Anytime Behavior

`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

- 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

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