Automatic Design of Algorithms with iRace for Multi-objective Optimization and Anytime Optimization

Manuel López-Ibáñez

manuel.lopez-ibanez@ulb.ac.be

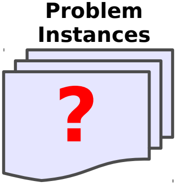
IRIDIA, CoDE, Université Libre de Bruxelles (ULB), Brussels, Belgium

PPSN 2012, Taormina, September 1st, 2012

ULB UNIVERSITÉ LIBRE DE BRUXELLES, UNIVERSITÉ D'EUROPE

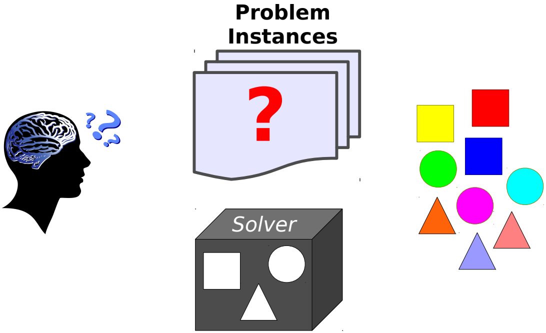
IRIDIA Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle

A non-expert user wants to repeatedly solve a problem

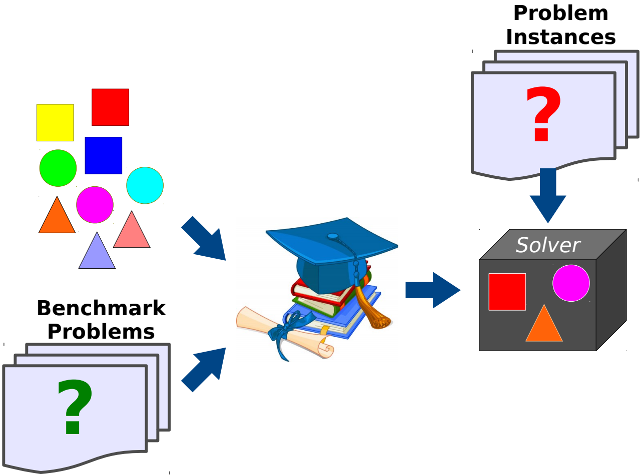


Offline Automatic Design of Algorithms

How to build a solver for this type of problem?

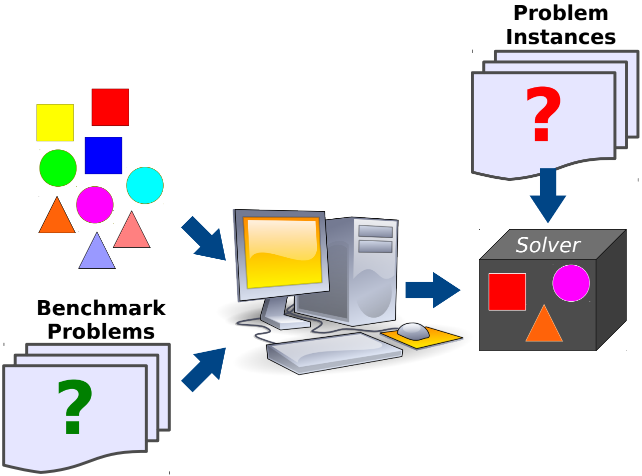


The user could rely on an expert



Offline Automatic Design of Algorithms

Can we replace the expert by an algorithm?



Offline Automatic Design of Algorithms

- ✔ Experts focus on creativity and understanding
- ✔ The machine does the boring experiments and statistics
- ✔ Formalize what is implicitly done in experimental research
- ✔ Avoid human biases

Offline Tuning / Automatic Configuration

- Find the best parameter configuration of a solver from a set of *training instances*
- Repeatedly use this configuration to solve *unseen instances* of the same problem

Automatic Design of Algorithms

Offline Tuning / Automatic Configuration

- Find the best parameter configuration of a solver from a set of *training instances*
- Repeatedly use this configuration to solve *unseen instances* of the same problem

Automatic Design of Algorithms

Goal: Automatically find a good instantiation of an optimization algorithm from a large space of potential designs for a specific problem.

Automatic Design of Algorithms

Offline Tuning / Automatic Configuration

- Find the best parameter configuration of a solver from a set of *training instances*
- Repeatedly use this configuration to solve *unseen instances* of the same problem

Automatic Design of Algorithms

Goal: Automatically find a good instantiation of an optimization algorithm from a large space of potential designs for a specific problem.

Solver	$\Rightarrow$	Flexible algorithmic framework
Parameter space	$\Rightarrow$	Design space of the framework

The algorithm design problem [Birattari, 2009]

Θ : set of potential algorithm designs (possibly infinite)

Θ : set of potential algorithm designs (possibly infinite)

$\mathcal{I}$: set of instances (possibly infinite), from which instances are sampled with certain probability

The algorithm design problem [Birattari, 2009]

Θ : set of potential algorithm designs (possibly infinite)

$\mathcal{I}$: set of instances (possibly infinite), from which instances are sampled with certain probability

$\mathcal{C}\colon \Theta \times \mathcal{I} \to \mathbb{R}$: cost measure, where:

The algorithm design problem [Birattari, 2009]

Θ : set of potential algorithm designs (possibly infinite)

$\mathcal{I}$: set of instances (possibly infinite), from which instances are sampled with certain probability

$\mathcal{C} \colon \Theta \times \mathcal{I} \to \mathbb{R}$: cost measure, where:

$\mathcal{C}(\theta, i)$: cost of design $\theta \in \Theta$ on instance $i \in \mathcal{I}$

The algorithm design problem [Birattari, 2009]

Θ : set of potential algorithm designs (possibly infinite)

$\mathcal{I}$: set of instances (possibly infinite), from which instances are sampled with certain probability

$\mathcal{C} : \Theta \times \mathcal{I} \to \mathbb{R}$: cost measure, where:

$\mathcal{C}(\theta, i)$: cost of design $\theta \in \Theta$ on instance $i \in \mathcal{I}$

$c(\theta, i)$: cost after running once design θ on instance i

The algorithm design problem

Θ : set of potential algorithm designs (possibly infinite)

$\mathcal{I}$: set of instances (possibly infinite), from which instances are sampled with certain probability

$\mathcal{C}\colon \Theta \times \mathcal{I} \to \mathbb{R}$: cost measure, where:

$\mathcal{C}(\theta, i)$: cost of design $\theta \in \Theta$ on instance $i \in \mathcal{I}$

$c(\theta, i)$: cost after running once design θ on instance i

c_θ : function of the cost $\mathcal{C}$ of a design θ with respect to the distribution of the random variable $\mathcal{I}$

The algorithm design problem [Birattari, 2009]

Θ : set of potential algorithm designs (possibly infinite)

$\mathcal{I}$: set of instances (possibly infinite), from which instances are sampled with certain probability

$\mathcal{C} \colon \Theta \times \mathcal{I} \to \mathbb{R}$: cost measure, where:

$\mathcal{C}(\theta, i)$: cost of design $\theta \in \Theta$ on instance $i \in \mathcal{I}$

$c(\theta, i)$: cost after running once design θ on instance i

c_θ : function of the cost $\mathcal{C}$ of a design θ with respect to the distribution of the random variable $\mathcal{I}$

Find the best algorithm design θ^* such that:

$$\theta^* = \arg\min_{\theta \in \Theta} c_\theta$$

The algorithm design problem: How to solve it?

Traditional approach

1. Expert chooses a number of algorithm designs $\Theta_0 \subset \Theta$
2. Expert chooses a number of benchmark problems $I_0 \subset \mathcal{I}$
3. Estimate c_θ for each $\theta \in \Theta_0$,
   by computing $c(\theta, i)$ for each $i \in I_0$
4. The design $\theta^* \in \Theta_0$ with lowest estimate of c_θ
   is the "winner"

The algorithm design problem: How to solve it?

Traditional approach

1. Expert chooses a number of algorithm designs $\Theta_0 \subset \Theta$
2. Expert chooses a number of benchmark problems $I_0 \subset \mathcal{I}$
3. Estimate c_θ for each $\theta \in \Theta_0$,
   by computing $c(\theta, i)$ for each $i \in I_0$
4. The design $\theta^* \in \Theta_0$ with lowest estimate of c_θ
   is the "winner"

Disadvantages

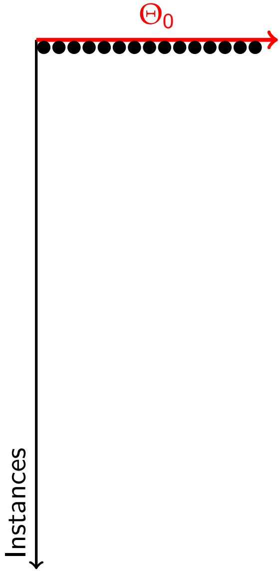
- ✘ Same computational effort spent on good and bad designs
- ✘ Small number of algorithm designs chosen a priori

The racing approach

[Birattari et al., 2002]

The racing approach

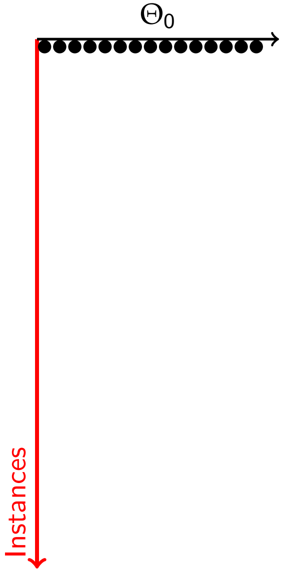
[Birattari et al., 2002]



- start with a set of initial candidates

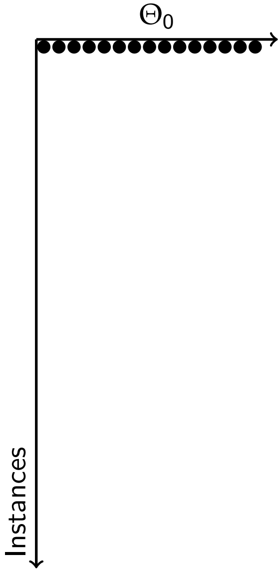
The racing approach

[Birattari et al., 2002]



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

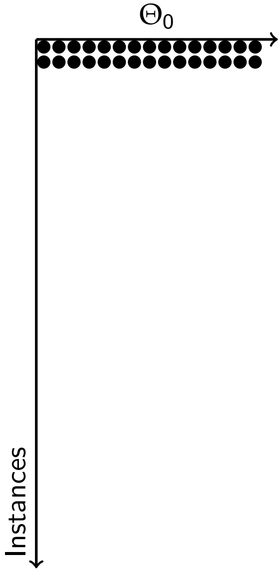
The racing approach
[Birattari et al., 2002]



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

The racing approach

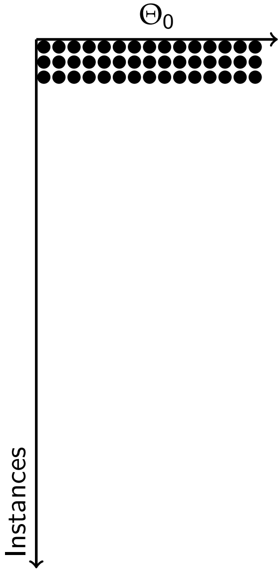
[Birattari et al., 2002]



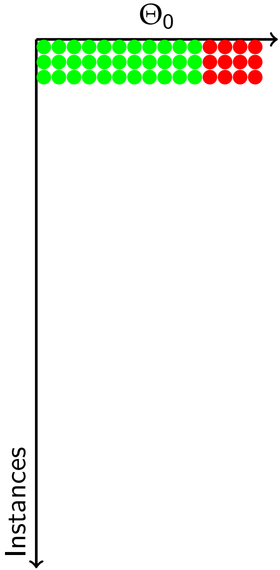
- start with a set of initial candidates
- consider a *stream* of instances
- sequentially evaluate candidates

The racing approach

[Birattari et al., 2002]

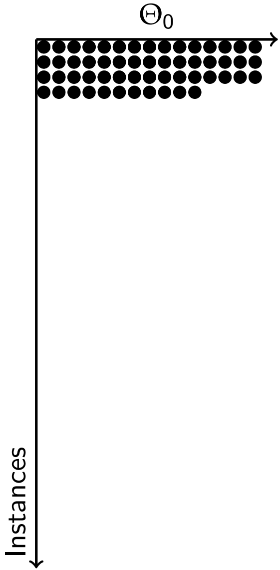


- start with a set of initial candidates
- consider a *stream* of instances
- sequentially evaluate candidates



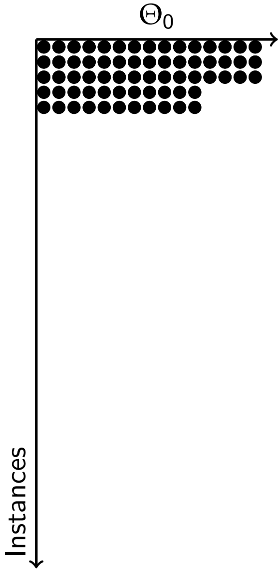
- start with a set of initial candidates
- consider a *stream* of instances
- sequentially evaluate candidates
- discard inferior candidates

  as sufficient evidence is gathered against them



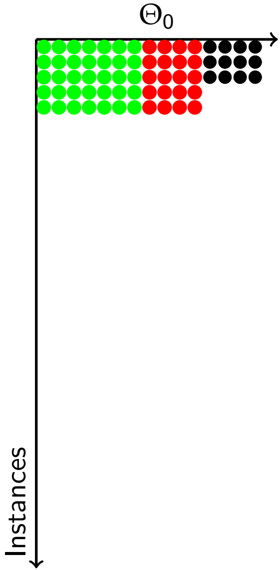
- start with a set of initial candidates
- consider a *stream* of instances
- sequentially evaluate candidates
- discard inferior candidates

  as sufficient evidence is gathered against them



- 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

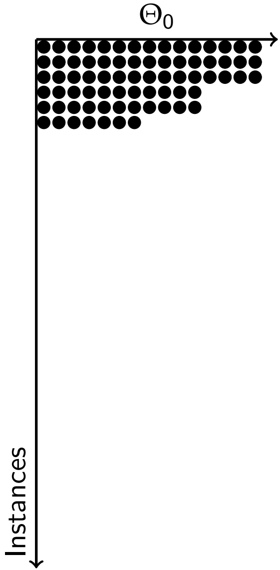


- start with a set of initial candidates
- consider a *stream* of instances
- sequentially evaluate candidates
- discard inferior candidates

  as sufficient evidence is gathered against them

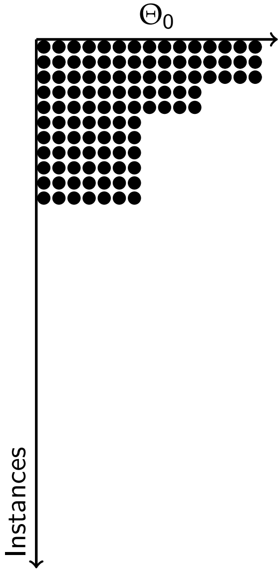
The racing approach

[Birattari et al., 2002]



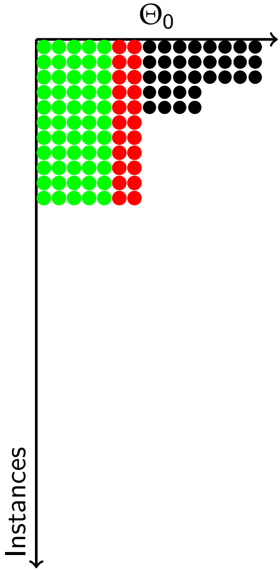
- start with a set of initial candidates
- consider a *stream* of instances
- sequentially evaluate candidates
- discard inferior candidates

  as sufficient evidence is gathered against them



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

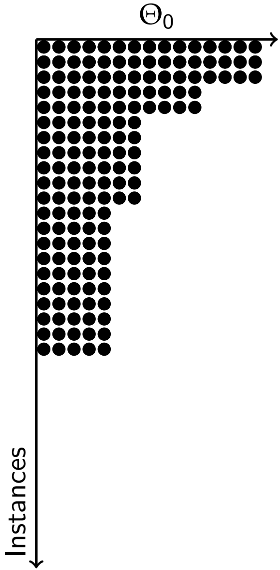
  as sufficient evidence is gathered against them



- 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

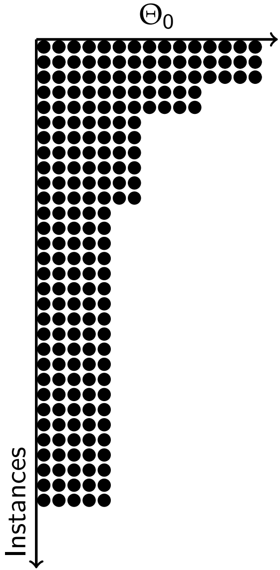
The racing approach



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

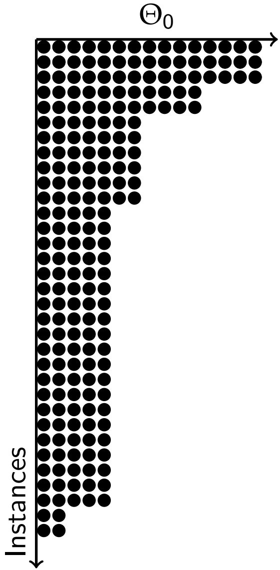
  as sufficient evidence is gathered against them

The racing approach



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

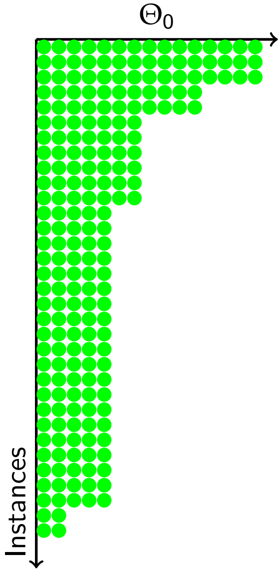
  as sufficient evidence is gathered against them



- 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

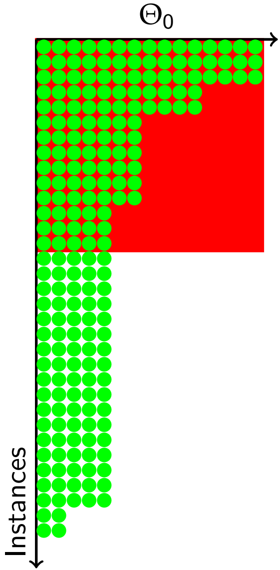
  or until computation time expires



- 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



- 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]

How to discard?

How to discard?

Statistical testing!

- *F-Race:* Friedman two-way analysis of variance by ranks + Friedman post-hoc test
- Alternative: paired t-test with/without p-value correction (against the best)

F-race is a method for the *selection of the best*
among a given set of algorithm designs

F-race is a method for the *selection of the best*
among a given set of algorithm designs

How to sample algorithm designs?

Sampling Algorithm Designs

F-race is a method for the *selection of the best* among a given set of algorithm designs

How to sample algorithm designs?

- Full factorial

Sampling Algorithm Designs

F-race is a method for the *selection of the best*
among a given set of algorithm designs

How to sample algorithm designs?

- Full factorial
- Random sampling

Sampling Algorithm Designs

F-race is a method for the *selection of the best*
among a given set of algorithm designs

How to sample algorithm designs?

- Full factorial
- Random sampling
- Iterative refinement of a sampling model
  ⇒ *Iterated F-Race (I/F-Race)* [Balaprakash et al., 2007]

Iterated F-Race [Balaprakash et al., 2007]

Require:

Training instances: $\{I_1, I_2, \dots\} \sim \mathcal{I}$,
Parameter space: X,
Cost measure: $\mathcal{C}\colon \Theta \times \mathcal{I} \to \mathbb{R}$,
Tuning budget: B

$\Theta_1 \sim \texttt{SampleUniform}(X)$
$\Theta^{\text{elite}} := \texttt{Race}(\Theta_1,\ B_1)$
$i := 2$
while $B_{\text{used}} \leq B$ **do**
    $\Theta^{\text{new}} \sim \texttt{Sample}(X, \Theta^{\text{elite}})$
    $\Theta_i := \Theta^{\text{new}} \cup \Theta^{\text{elite}}$
    $\Theta^{\text{elite}} := \texttt{Race}(\Theta_i,\ B_i)$
    $i := i + 1$
Output: Θ^{elite}

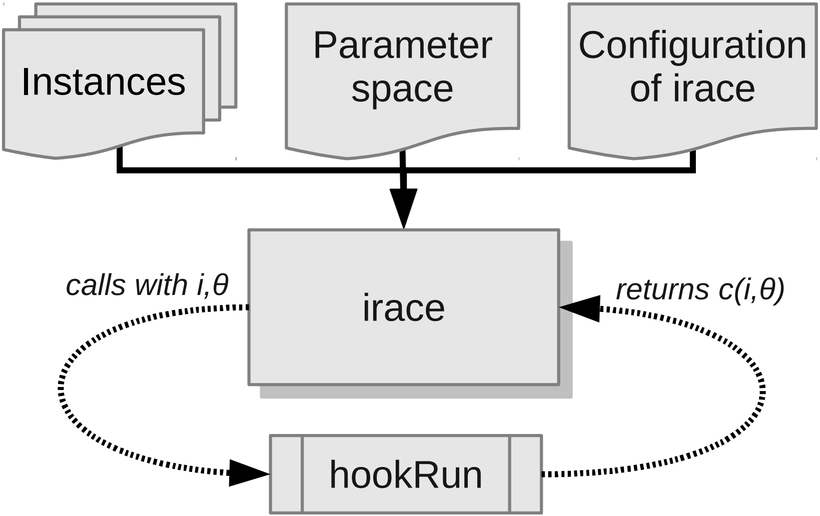
The `irace` Package

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

Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Thomas Stützle, and Mauro Birattari. **The irace package, Iterated Race for Automatic Algorithm Configuration.** *Technical Report TR/IRIDIA/2011-004*, IRIDIA, Université Libre de Bruxelles, Belgium, 2011.

- Implementation of I/F-Race with a few extensions
- R package available at CRAN
- Flexible
- Easy to use
- No knowledge of R needed
- Command-line wrapper: `irace --help`

The irace Package



The irace Package

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

- Parameter space:

```
LS          c  {SA, best, first}
rate        o  {low, med, high }
population  i  (1, 100)
temp        r  (0.5, 1)          if LS == "SA"
```

The irace Package

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

- Parameter space:

```
LS          c  {SA, best, first}
rate        o  {low, med, high }
population  i  (1, 100)
temp        r  (0.5, 1)          if LS == "SA"
```

- Initial configurations may be explicitly provided

The irace Package

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

- Parameter space:

```
LS          c   {SA, best, first}
rate        o   {low, med, high }
population  i   (1, 100)
temp        r   (0.5, 1)          if LS == "SA"
```

- Initial configurations may be explicitly provided
- Parallel execution:
  - on a single machine (`multicore` package)
  - using MPI (`Rmpi` package)
  - using a Grid Engine cluster (with `qsub` and `qstat`)

Advance Applications of irace

1. Automatic design of multi-objective optimization algorithms
2. Automatically improving the anytime behavior of algorithms

Multi-objective Optimization

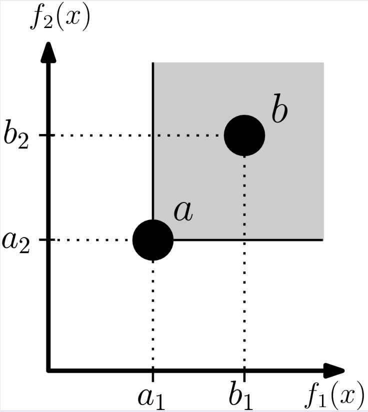
Multiple objective functions: $\vec{f} = (f_1(x), f_2(x), \dots)$

No a priori knowledge $\Rightarrow$ *Pareto-optimality*

Multiple objective functions: $\vec{f} = (f_1(x), f_2(x), \ldots)$

No a priori knowledge $\Rightarrow$ *Pareto-optimality*

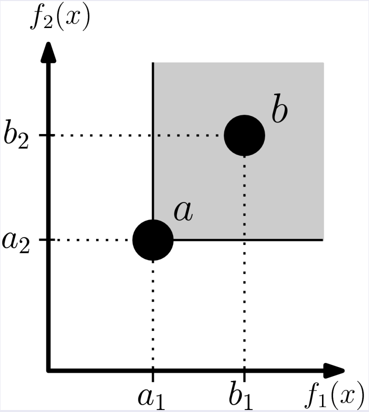
a dominates *b*



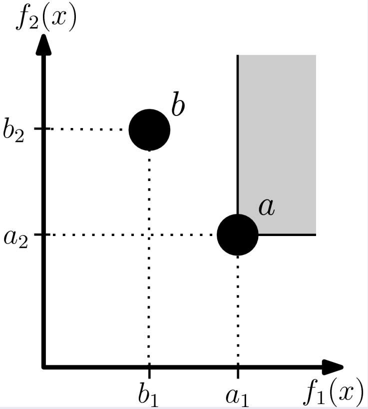
Multiple objective functions: $\vec{f} = (f_1(x), f_2(x), \ldots)$

No a priori knowledge $\Rightarrow$ *Pareto-optimality*

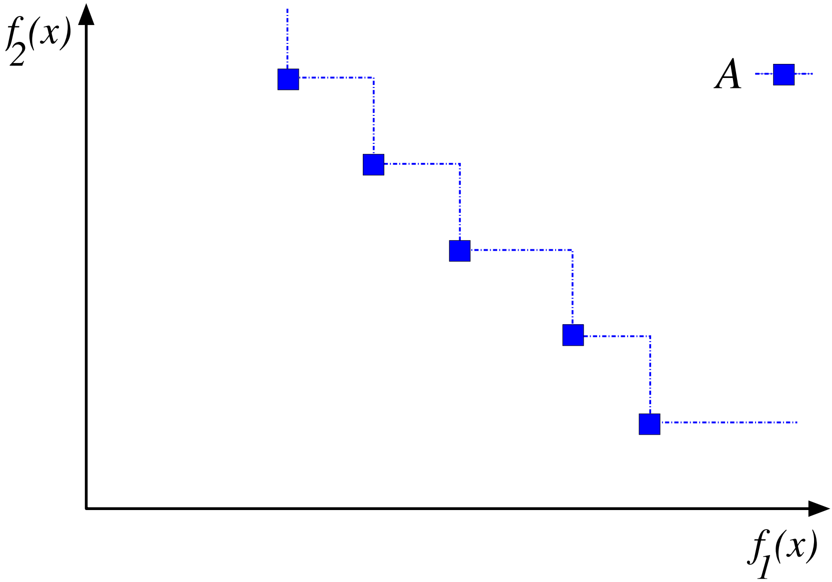
a dominates *b*



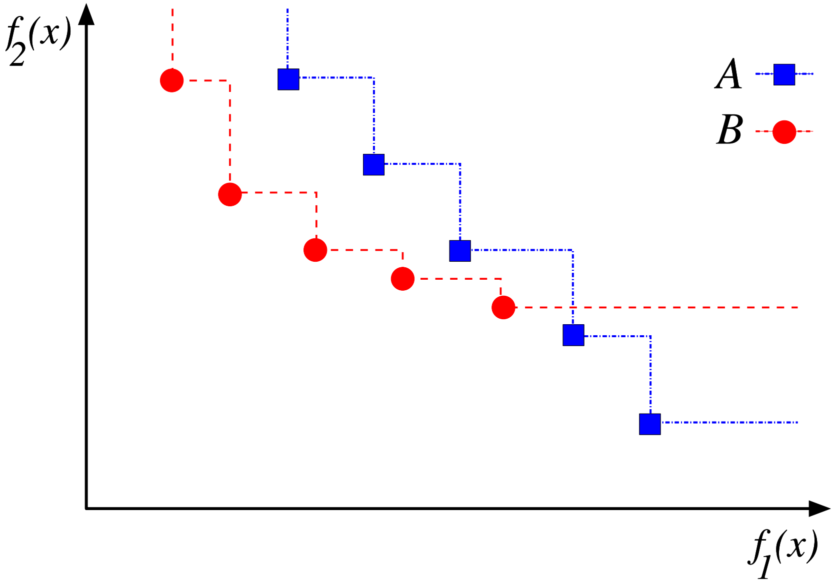
a and *b* nondominated



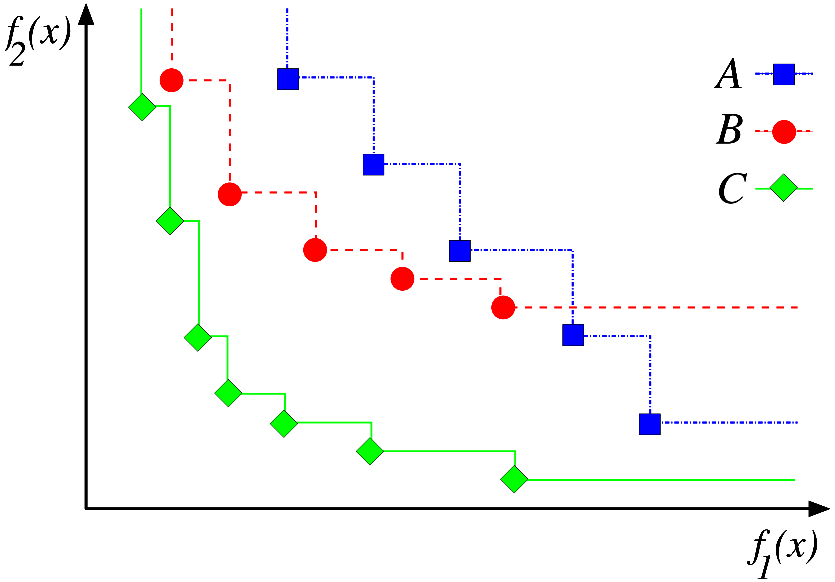
Multi-objective Optimization



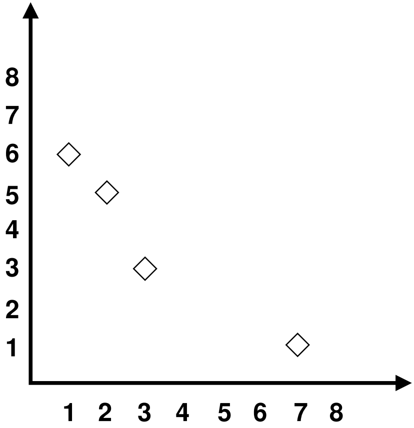
Multi-objective Optimization



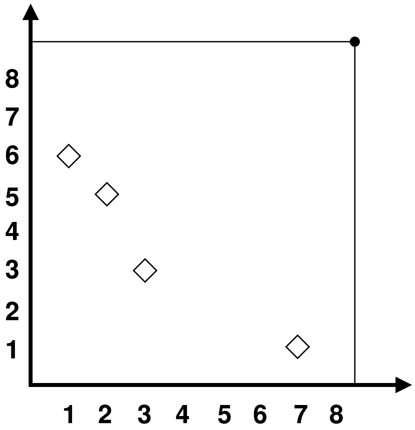
Multi-objective Optimization



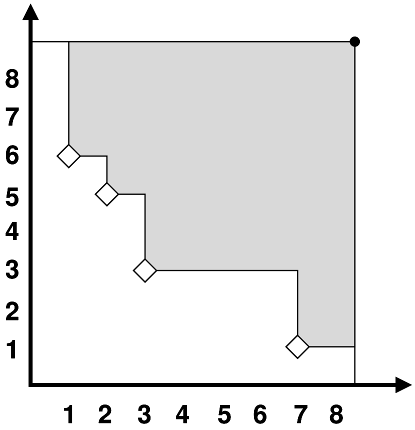
Hypervolume measure



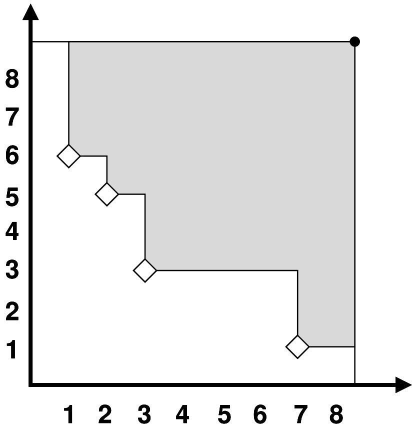
Hypervolume measure



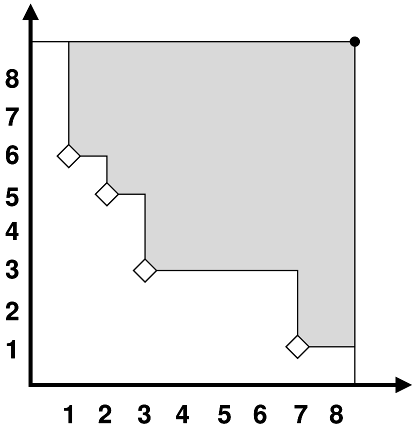
Hypervolume measure



Hypervolume measure



Hypervolume measure



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

Automatic Design of MOACO Algorithms

Automatic Design of MOACO Algorithms

- Multiple objective Ant-Q (MOAQ) [Mariano & Morales, 1999] [García-Martínez et al., 2007]
- MACS-VRPTW [Gambardella et al., 1999]
- BicriterionAnt [Iredi et al., 2001]
- SACO [T'Kindt et al., 2002]
- Multiobjective Network ACO [Cardoso et al., 2003]
- Multicriteria Population-based ACO [Guntsch & Middendorf, 2003]
- MACS [Barán & Schaerer, 2003]
- COMPETants [Doerner et al., 2003]
- Pareto ACO [Doerner et al., 2004]
- Multiple Objective ACO Metaheuristic [Gravel et al., 2002]
- MOACO-bQAP [López-Ibáñez et al., 2004]
- MOACO-ALBP [Baykasoglu et al., 2005]
- mACO-$\{1, 2, 3, 4\}$ [Alaya et al., 2007]
- Population-based ACO [Angus, 2007]

M. López-Ibáñez and T. Stützle. The automatic design of multi-objective ant colony optimization algorithms. *IEEE Transactions on Evolutionary Computation*, 2012.

Automatic Design of MOACO Algorithms

M. López-Ibáñez and T. Stützle. The automatic design of multi-objective ant colony optimization algorithms. *IEEE Transactions on Evolutionary Computation*, 2012.

1. Isolate design choices ⇒ Algorithmic components:
    - Algorithmic components used in existing MOACO algorithms
    - Algorithmic components never proposed before

Automatic Design of MOACO Algorithms

M. López-Ibáñez and T. Stützle. The automatic design of multi-objective ant colony optimization algorithms. *IEEE Transactions on Evolutionary Computation*, 2012.

1. Isolate design choices ⇒ Algorithmic components:
   - Algorithmic components used in existing MOACO algorithms
   - Algorithmic components never proposed before
2. Synthesize knowledge into a configurable MOACO framework able to instantiate existing *and new* MOACO algorithms

Automatic Design of MOACO Algorithms

M. López-Ibáñez and T. Stützle. The automatic design of multi-objective ant colony optimization algorithms. *IEEE Transactions on Evolutionary Computation*, 2012.

1. Isolate design choices ⇒ Algorithmic components:
   - Algorithmic components used in existing MOACO algorithms
   - Algorithmic components never proposed before
2. Synthesize knowledge into a configurable MOACO framework able to instantiate existing *and new* MOACO algorithms
3. Use `irace` + hypervolume to find the best MOACO designs

A flexible MOACO framework

- Multi-objective algorithmic design: 10 parameters
- Instantiates 9 MOACO algorithms from the literature
- $> 25\,000$ potential designs
- Underlying ACO settings are also configurable
- Implemented for bi-objective TSP and bi-objective Knapsack

Experiments

Worst	⇓	Best MOACO of literature + fixed ACO settings
		Tuned MOACO design + fixed ACO settings
		Best MOACO of literature + *tuned* ACO settings
		Tuned MOACO design + *tuned* ACO settings
Best		*Tuned* (MOACO design + ACO settings)

Conclusions

- ✔ `irace` + hypervolume = automatic design of MO algorithms

Conclusions

- ✔ `irace` + hypervolume = automatic design of MO algorithms
- ✔ `irace` typically better than humans . . .
  . . . if given a good design space

Conclusions

- ✔ `irace` + hypervolume = automatic design of MO algorithms
- ✔ `irace` typically better than humans ...
  ... if given a good design space
- ✔ Another example:

  J. Dubois-Lacoste, M. López-Ibáñez, and T. Stützle. **Automatic configuration of state-of-the-art multi-objective optimizers using the TP+PLS framework**. *GECCO*, 2011.

Automatically Improving the Anytime Behavior of Optimization Algorithms

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

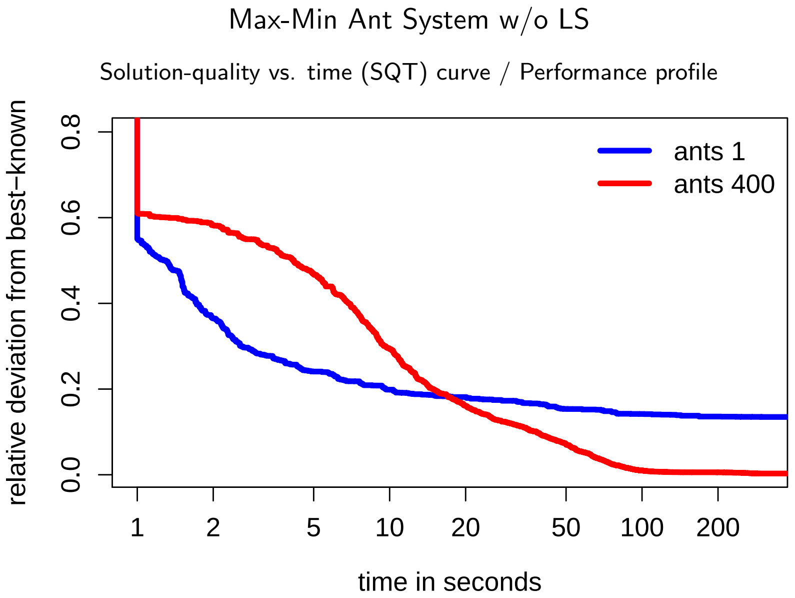
Automatically Improving the Anytime Behavior of Optimization Algorithms

Anytime Algorithm [Dean & Boddy, 1988]

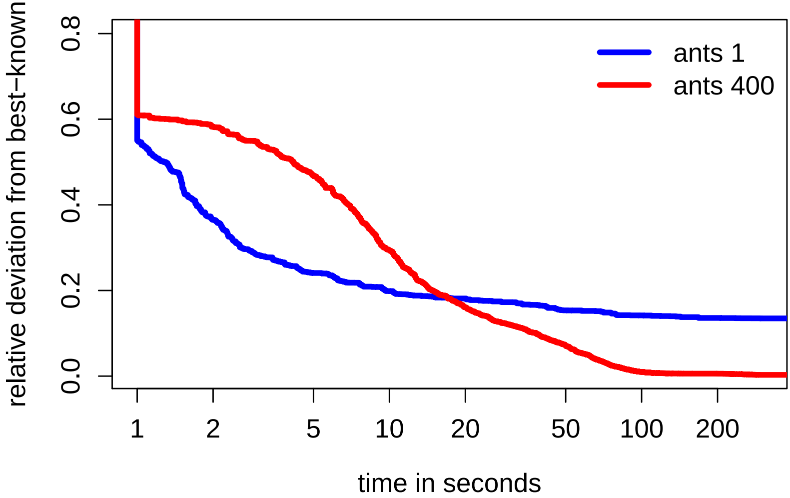
- May be interrupted at any moment and returns a solution
- Keeps improving its solution until interrupted
- Eventually finds the optimal solution

Good Anytime Behavior [Zilberstein, 1996]

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

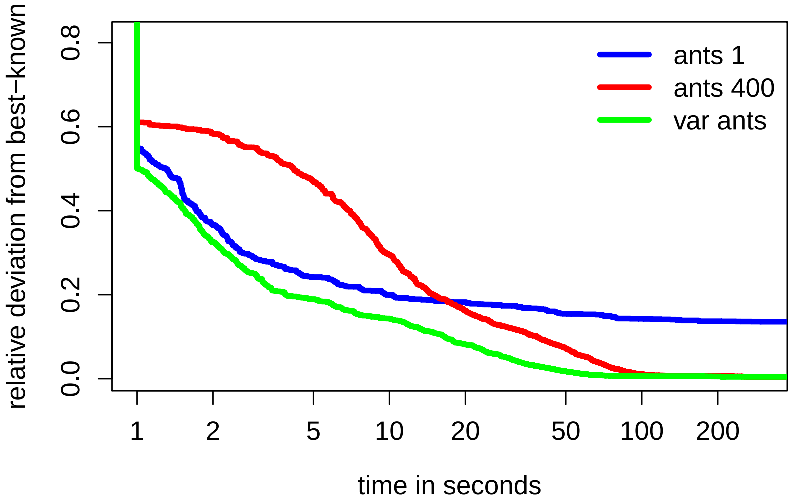

Max-Min Ant System w/o LS
Solution-quality vs. time (SQT) curve / Performance profile
relative deviation from best−known
0.0
0.2
0.4
0.6
0.8
1
2
5
10
20
50
100
200
time in seconds
ants 1
ants 400

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



Quality vs. Time Trade-off

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



Improving "Anytime" Behaviour

How to improve the anytime behaviour of MMAS?

☞ 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?

☞ 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?

Brute-Force Approach

1. Choose *many* different parameter variation strategies
2. Run lots of experiments
3. Visually compare SQT plots

Brute-Force Approach

1. Choose *many* different parameter variation strategies
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.

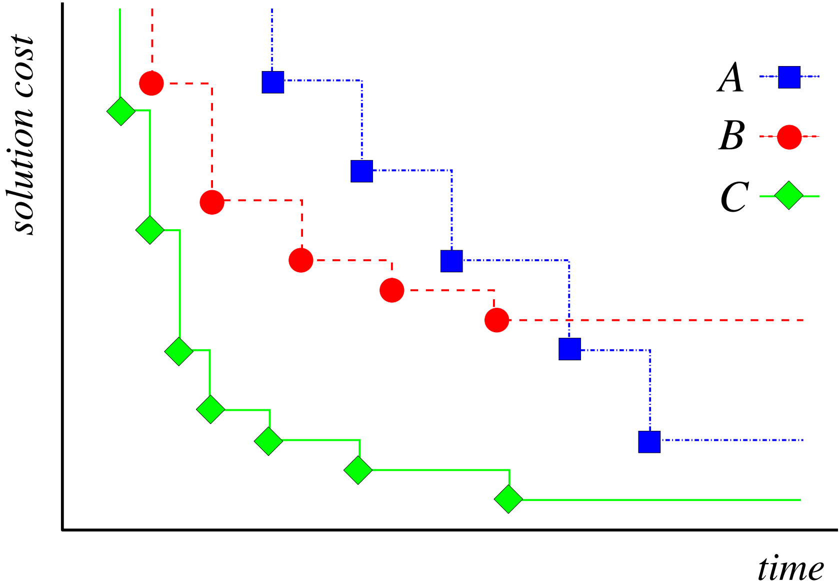
Brute-Force Approach

1. Choose *many* different parameter variation strategies
2. Run lots of experiments
3. Visually compare SQT plots

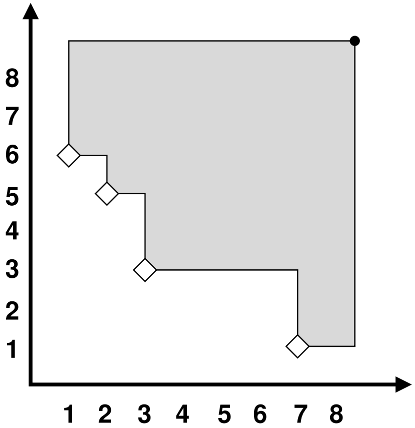
After one year and a master thesis: [Maur et al., 2010]

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

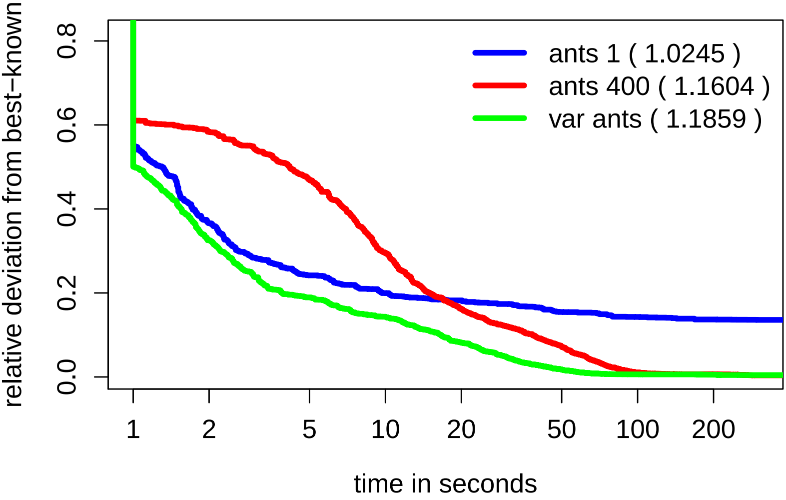
Bi-Objective Optimisation



Hypervolume measure



Hypervolume measure



Our Proposed Approach

`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

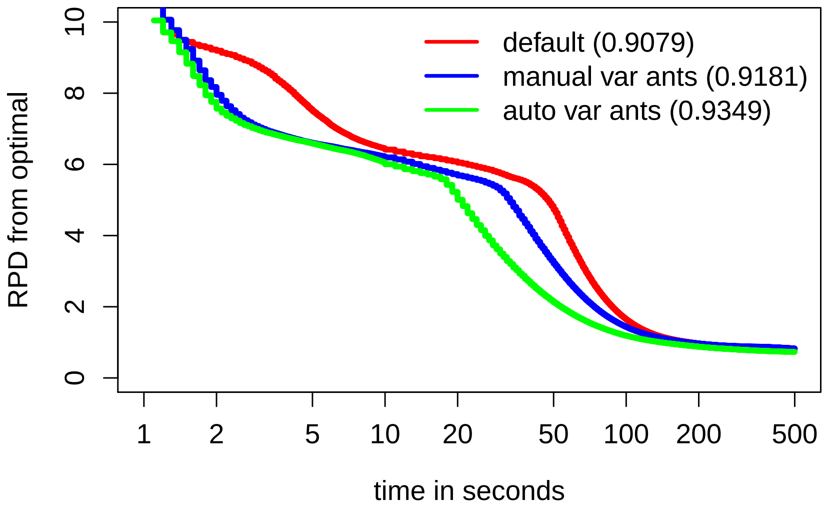
Experiments

- Time-varying *ants* (m): 6 parameters

Param.	Domain	Condition
m_{var}	{ *delta*, *switch*, *none* }	
m	[1, 100]	if var = *none*
Δm	$\{0.01, 0.05, 0.1, 0.25, 0.5, 1, 2, 5\}$	if var = *delta*
m_{switch}	[1, 500]	if var = *switch*
m_{start} m_{end}	1 [1, 500]	if var $\in$ {*delta*, *switch*}

- Other parameters are set to default
- Tuning budget: 1000 runs

Experiments

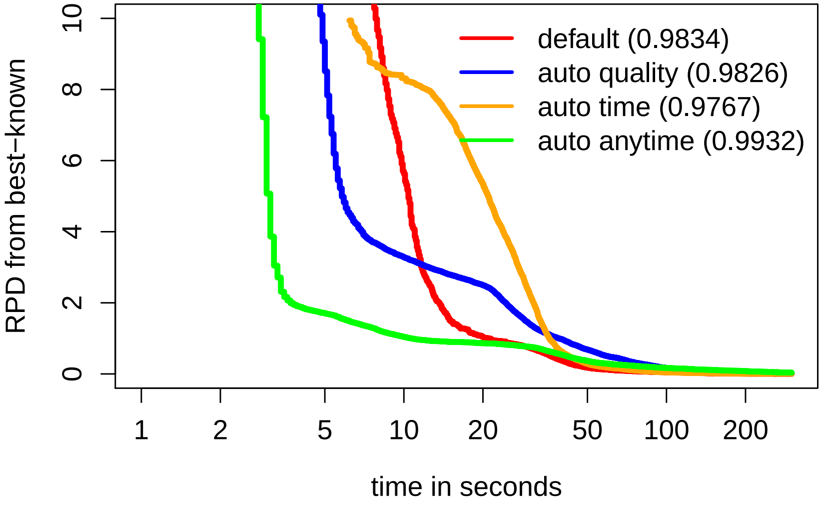


Automatically Improving the Anytime Behaviour of SCIP

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
- Tuning budget: 5 000 runs

Automatically Improving the Anytime Behaviour of SCIP



Automatically Improving Anytime Behavior

M. López-Ibáñez and T. Stützle. **Automatically improving the anytime behaviour of optimisation algorithms**. Technical Report TR/IRIDIA/2012-012, IRIDIA, Université Libre de Bruxelles, Belgium, 2012.

Automatically Improving Anytime Behavior

M. López-Ibáñez and T. Stützle. **Automatically improving the anytime behaviour of optimisation algorithms**. Technical Report TR/IRIDIA/2012-012, IRIDIA, Université Libre de Bruxelles, Belgium, 2012.

- How to introduce a bias towards final quality?
  - ☞ Compute hypervolume on transformed y-axis
  - ☞ Weighted hypervolume [Zitzler et al., 2007]

Automatically Improving Anytime Behavior

M. López-Ibáñez and T. Stützle. **Automatically improving the anytime behaviour of optimisation algorithms**. Technical Report TR/IRIDIA/2012-012, IRIDIA, Université Libre de Bruxelles, Belgium, 2012.

- How to introduce a bias towards final quality?
  - ☞ Compute hypervolume on transformed y-axis
  - ☞ Weighted hypervolume [Zitzler et al., 2007]
- How to define a cut-off time as short as possible?
  - ☞ Estimate point of diminishing returns [Woodruff et al., 2011]
  - ☞ Survival analysis techniques [Gagliolo & Legrand, 2010]

The End

irace

Easy, flexible, state-of-the-art automatic configuration tool

The End

`irace`

Easy, flexible, state-of-the-art automatic configuration tool

Automatic Design of Algorithms

Automatically find a good instantiation of an optimization algorithm from a large space of potential designs for a specific problem.

The End

irace

Easy, flexible, state-of-the-art automatic configuration tool

Automatic Design of Algorithms

Automatically find a good instantiation of an optimization algorithm from a large space of potential designs for a specific problem.

irace + hypervolume = automatic design of

- multi-objective optimization algorithms
- anytime optimization algorithms

Automatic Design of Algorithms with iRace for Multi-objective Optimization and Anytime Optimization

Manuel López-Ibáñez

manuel.lopez-ibanez@ulb.ac.be

http://iridia.ulb.ac.be/~manuel

UNIVERSITÉ LIBRE DE BRUXELLES, UNIVERSITÉ D'EUROPE

IRIDIA Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle

References I

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

I. Alaya, C. Solnon, and K. Ghédira. Ant colony optimization for multi-objective optimization problems. In *19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007)*, volume 1, pages 450–457. IEEE Computer Society Press, Los Alamitos, CA, 2007.

D. Angus. Population-based ant colony optimisation for multi-objective function optimisation. In M. Randall, H. A. Abbass, and J. Wiles, editors, *Progress in Artificial Life (ACAL)*, volume 4828 of *Lecture Notes in Computer Science*, pages 232–244. Springer, Heidelberg, Germany, 2007. doi: 10.1007/978-3-540-76931-6_21.

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.

B. Barán and M. Schaerer. A multiobjective ant colony system for vehicle routing problem with time windows. In *Proceedings of the Twenty-first IASTED International Conference on Applied Informatics*, pages 97–102, Insbruck, Austria, 2003.

References II

A. Baykasoglu, T. Dereli, and I. Sabuncu. A multiple objective ant colony optimization approach to assembly line balancing problems. In *35th International Conference on Computers and Industrial Engineering (CIE35)*, pages 263–268, Istanbul, Turkey, 2005.

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.

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.

P. Cardoso, M. Jesus, and A. Marquez. Monaco multi-objective network optimisation based on an aco. In *Proc. X Encuentros de Geometría Computacional*, Seville, Spain, 2003.

T. Dean and M. S. Boddy. An analysis of time-dependent planning. In *Proceedings of the 7th National Conference on Artificial Intelligence, AAAI-88*, pages 49–54. AAAI Press, 1988.

K. F. Doerner, R. F. Hartl, and M. Reimann. Are COMPETants more competent for problem solving? The case of a multiple objective transportation problem. *Central European Journal for Operations Research and Economics*, 11(2):115–141, 2003.

K. F. Doerner, W. J. Gutjahr, R. F. Hartl, C. Strauss, and C. Stummer. Pareto ant colony optimization: A metaheuristic approach to multiobjective portfolio selection. *Annals of Operations Research*, 131:79–99, 2004.

M. Gagliolo and C. Legrand. Algorithm survival analysis. In T. Bartz-Beielstein, M. Chiarandini, L. Paquete, and M. Preuss, editors, *Experimental Methods for the Analysis of Optimization Algorithms*, pages 161–184. Springer, Berlin, Germany, 2010. doi: 10.1007/978-3-642-02538-9_7.

L. M. Gambardella, É. D. Taillard, and G. Agazzi. MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 63–76. McGraw Hill, London, UK, 1999.

C. García-Martínez, O. Cordón, and F. Herrera. A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP. *European Journal of Operational Research*, 180(1):116–148, 2007.

M. Gravel, W. L. Price, and C. Gagné. Scheduling continuous casting of aluminum using a multiple objective ant colony optimization metaheuristic. *European Journal of Operational Research*, 143(1):218–229, 2002. doi: 10.1016/S0377-2217(01)00329-0.

References IV

M. Guntsch and M. Middendorf. Solving multi-objective permutation problems with population based ACO. In C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, and L. Thiele, editors, *Evolutionary Multi-criterion Optimization (EMO 2003)*, volume 2632 of *Lecture Notes in Computer Science*, pages 464–478. Springer, Heidelberg, Germany, 2003.

S. Iredi, D. Merkle, and M. Middendorf. Bi-criterion optimization with multi colony ant algorithms. In E. Zitzler, K. Deb, L. Thiele, C. A. Coello Coello, and D. Corne, editors, *Evolutionary Multi-criterion Optimization (EMO 2001)*, volume 1993 of *Lecture Notes in Computer Science*, pages 359–372. Springer, Heidelberg, Germany, 2001.

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.

M. López-Ibáñez, L. Paquete, and T. Stützle. On the design of ACO for the biobjective quadratic assignment problem. In M. Dorigo et al., editors, *Ant Colony Optimization and Swarm Intelligence, 4th International Workshop, ANTS 2004*, volume 3172 of *Lecture Notes in Computer Science*, pages 214–225. Springer, Heidelberg, Germany, 2004. doi: 10.1007/978-3-540-28646-2_19.

References V

C. E. Mariano and E. Morales. MOAQ: An Ant-Q algorithm for multiple objective optimization problems. In W. Banzhaf, J. M. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. J. Jakiela, and R. E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 1999*, pages 894–901. Morgan Kaufmann Publishers, San Francisco, CA, 1999.

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.

V. T'Kindt, N. Monmarché, F. Tercinet, and D. Laügt. An ant colony optimization algorithm to solve a 2-machine bicriteria flowshop scheduling problem. *European Journal of Operational Research*, 142(2):250–257, 2002.

D. L. Woodruff, U. Ritzinger, and J. Oppen. Research note: the point of diminishing returns in heuristic search. *International Journal of Metaheuristics*, 1(3):222–231, 2011. doi: 10.1504/IJMHeur.2011.041195.

S. Zilberstein. Using anytime algorithms in intelligent systems. *AI Magazine*, 17(3): 73–83, 1996.

References VI

E. Zitzler, D. Brockhoff, and L. Thiele. The hypervolume indicator revisited: On the design of Pareto-compliant indicators via weighted integration. In S. Obayashi et al., editors, *Evolutionary Multi-criterion Optimization (EMO 2007)*, volume 4403 of *Lecture Notes in Computer Science*, pages 862–876. Springer, Heidelberg, Germany, 2007. doi: 10.1007/978-3-540-70928-2_64.