Westfälische Wilhelms-Universität Münster

# Parameterization of State-of-the-Art Performance Indicators: A Robustness Study Based on Inexact TSP Solvers

Pascal Kerschke, Jakob Bossek, Heike Trautmann

July 16, 2018

Department for Informations Systems and Statistics, University of Münster, Germany

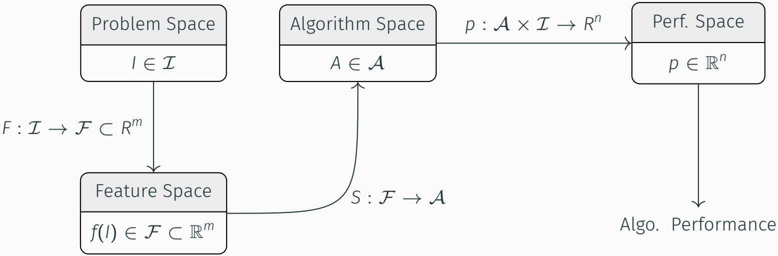
# INTRODUCTION

**Algorithm Selection Problem [8]**

Given a previously unseen problem instance, determine, given a **portfolio of algorithms**, the algorithm, which will most likely perform best.

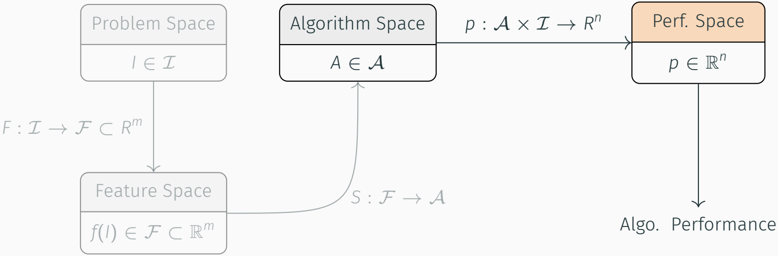
## Algorithm Selection Problem [8]

Given a previously unseen problem instance, determine, given a **portfolio of algorithms**, the algorithm, which will most likely perform best.



## Algorithm Selection Problem [8]

Given a previously unseen problem instance, determine, given a **portfolio of algorithms**, the algorithm, which will most likely perform best.



- Comprehensive benchmark of portfolio solvers required as a foundation for algorithm selection.
- Suitable performance measure needed, e. g., PAR [1], ERT [4].
- Performance measures often parameterized.
  $\leadsto$ How do parameters affect the benchmark results?

- Comprehensive benchmark of portfolio solvers required as a foundation for algorithm selection.
- Suitable performance measure needed, e. g., PAR [1], ERT [4].
- Performance measures often parameterized.
  $\rightsquigarrow$ How do parameters affect the benchmark results?

**Our contribution**

Systematic analysis of parameterizations on a comprehensive benchmark study of inexact TSP solvers.

We consider:

- Set of problem instances $\mathcal{I} = \{I_1, \ldots, I_{n_\mathcal{I}}\}$,

We consider:

- Set of problem instances $\mathcal{I} = \{I_1, \ldots, I_{n_{\mathcal{I}}}\}$,
- Set of algorithms/solvers $\mathcal{A} = \{A_1, \ldots, A_{n_{\mathcal{A}}}\}$,

We consider:

- Set of problem instances $\mathcal{I} = \{I_1, \ldots, I_{n_\mathcal{I}}\}$,
- Set of algorithms/solvers $\mathcal{A} = \{A_1, \ldots, A_{n_\mathcal{A}}\}$,
- $m > 1$ independent runs of each $A \in \mathcal{A}$ on $I \in \mathcal{I}$

# NOTATION

We consider:

- Set of problem instances $\mathcal{I} = \{I_1, \ldots, I_{n_\mathcal{I}}\}$,
- Set of algorithms/solvers $\mathcal{A} = \{A_1, \ldots, A_{n_\mathcal{A}}\}$,
- $m > 1$ independent runs of each $A \in \mathcal{A}$ on $I \in \mathcal{I}$
- $\rightsquigarrow$ empirical runtimes $r_1^{A,I}, \ldots, r_m^{A,I}$.

We consider:

- Set of problem instances $\mathcal{I} = \{I_1, \ldots, I_{n_{\mathcal{I}}}\}$,
- Set of algorithms/solvers $\mathcal{A} = \{A_1, \ldots, A_{n_{\mathcal{A}}}\}$,
- $m > 1$ independent runs of each $A \in \mathcal{A}$ on $I \in \mathcal{I}$
- $\rightsquigarrow$ empirical runtimes $r_1^{A,I}, \ldots, r_m^{A,I}$.
- Time limit / cutoff time $T \in \mathbb{R}_{>0}$.

# PERFORMANCE MEASURES

## Penalized Average Runtime (PAR, [1])

Arithmetic mean of running times, $r_i^{A,I}, i \in [m]$; expired runs are penalized by factor $f \cdot T$, where $f$ is the **penalty factor**.

## Penalized Average Runtime (PAR, [1])

Arithmetic mean of running times, $r_i^{A,I}, i \in [m]$; expired runs are penalized by factor $f \cdot T$, where $f$ is the **penalty factor**.

$$\mathrm{PAR}_{A,I}(f) := \frac{1}{m}\sum_{i=1}^{m} \tilde{r}_i^{A,I} \quad \text{with} \quad \tilde{r}_i^{A,I} = \begin{cases} f \cdot T, & \text{if } r_i^{A,I} > T \\ r_i^{A,I}, & \text{otherwise.} \end{cases}$$

Penalized Quantile Runtime (PQR)

Replace outlier-sensitive mean by more robust $p$-**quantile**, $p \in (0, 1]$.

### Penalized Quantile Runtime (PQR)

Replace outlier-sensitive mean by more robust $p$-**quantile**, $p \in (0, 1]$.

$$\mathrm{PQR}_{A,I}(p, f) := \begin{cases} f \cdot T, & \text{if } \sum_{i=1}^{m} \mathbb{1}\{r_i^{A,I} < T\} < \lfloor mp + 1 \rfloor \\ q_p(r_1^{A,I}, \ldots, r_m^{A,I}), & \text{otherwise.} \end{cases}$$

**Penalized Expected Runtime (PERT)**

Introducing penalty factor into **Expected Runtime** (ERT, [4]).

## Penalized Expected Runtime (PERT)

Introducing penalty factor into **Expected Runtime** (ERT, [4]).

$$\begin{aligned}\mathrm{PERT}_{A,I}(f) &= \frac{1}{s}\sum_{j=1}^{s} r_{i_j}^{A,I} + \left(\frac{1-p_s}{p_s}\right)\cdot f\cdot T \\ &= \frac{1}{s}\left(\sum_{j=1}^{s} r_{i_j}^{A,I} + (m-s)\cdot f\cdot T\right).\end{aligned}$$

Based on performance data from our previous TSP algorithm selection study [6]:

### Algorithms $\mathcal{A}$

Five state-of-the-art inexact TSP solvers: MAOS [9], EAX [7], LKH [5], EAX+restart and LKH+restart [3].

### Problems $\mathcal{I}$

Five sets of TSP instances: VLSI, TSPLIB, RUE, clustered (netgen) and morphed.

Based on performance data from our previous TSP algorithm selection study [6]:

## Algorithms $\mathcal{A}$

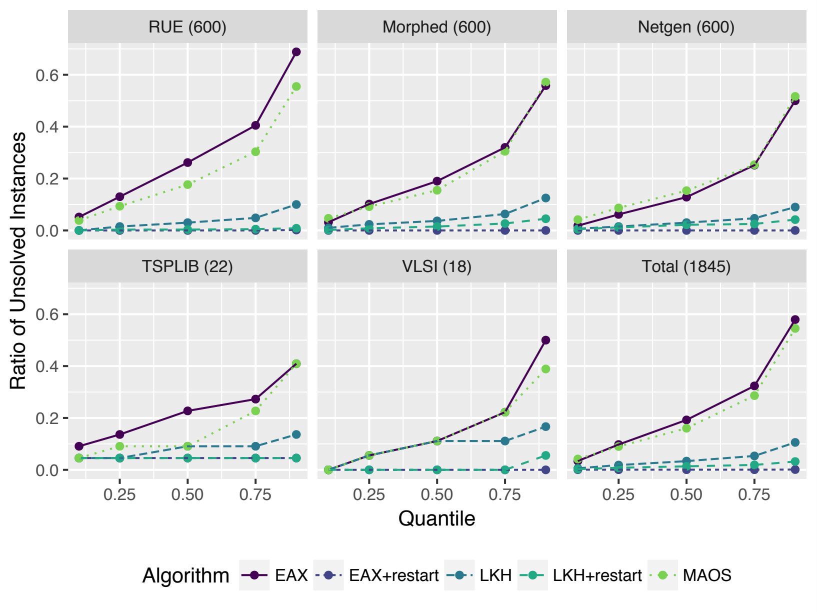
Five state-of-the-art inexact TSP solvers: MAOS [9], EAX [7], LKH [5], EAX+restart and LKH+restart [3].

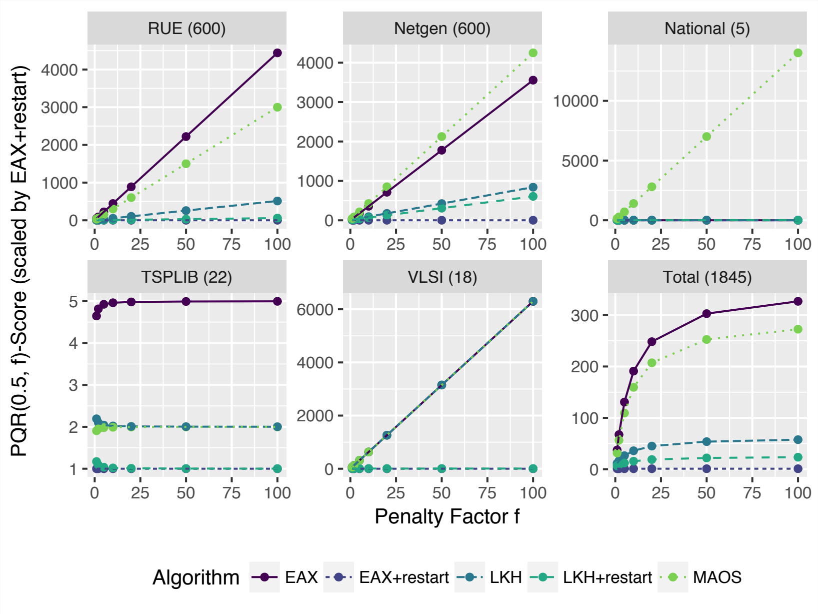
## Problems $\mathcal{I}$

Five sets of TSP instances: VLSI, TSPLIB, RUE, clustered (netgen) and morphed.

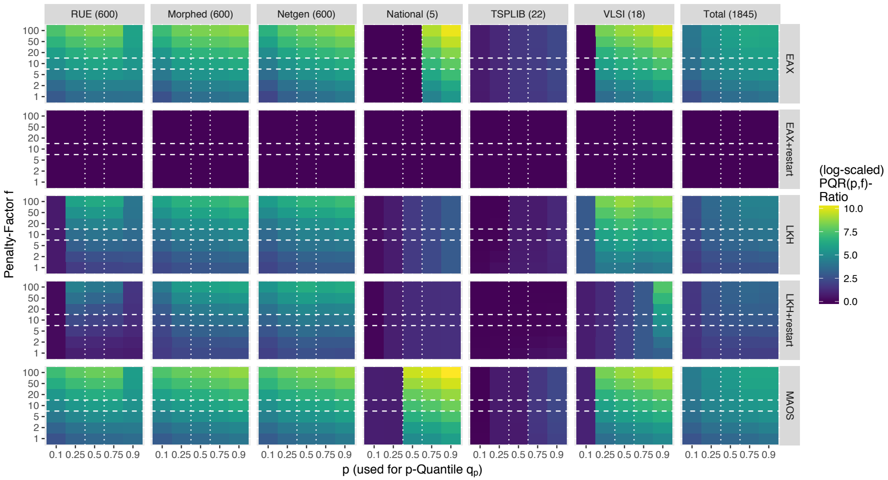
EAX+restart was single-best-solver (SBS) regarding PAR-10.

# RESULTS


RUE (600)
Morphed (600)
Netgen (600)
TSPLIB (22)
VLSI (18)
Total (1845)
Ratio of Unsolved Instances
0.0
0.2
0.4
0.6
0.25
0.50
0.75
Quantile
Algorithm
EAX
EAX+restart
LKH
LKH+restart
MAOS


RUE (600)
Netgen (600)
National (5)
TSPLIB (22)
VLSI (18)
Total (1845)
PQR(0.5, f)-Score (scaled by EAX+restart)
Penalty Factor f
0
25
50
75
100
Algorithm
EAX
EAX+restart
LKH
LKH+restart
MAOS

# RESULTS


RUE (600)
Morphed (600)
Netgen (600)
National (5)
TSPLIB (22)
VLSI (18)
Total (1845)
EAX
EAX+restart
LKH
LKH+restart
MAOS
Penalty-Factor f
100
50
20
10
5
2
1
0.1 0.25 0.5 0.75 0.9
p (used for p-Quantile $q_p$)
(log-scaled) PQR(p,f)-Ratio
10.0
7.5
5.0
2.5
0.0

# CONCLUSION

## Conclusions

We systematically analyzed effects of different parameterizations of performance indicators.

- Varying quantile has no effect on EAX+restart (our SBS)
- Varying penalty factor allow for altering leverage of failed runs.
- (P)ERT is much more prone to single runs $\rightsquigarrow$ huge impact of single failed runs.

## Conclusions

We systematically analyzed effects of different parameterizations of performance indicators.

- Varying quantile has no effect on EAX+restart (our SBS)
- Varying penalty factor allow for altering leverage of failed runs.
- (P)ERT is much more prone to single runs $\rightsquigarrow$ huge impact of single failed runs.

## Outlook

- Theoretical investigations of indicators.
- Introducation of alternative (multi-objective) indicators (see Bossek and Trautmann [2]).
- Application in context of algorithm selection.

# Questions?

# REFERENCES


[1] Bernd Bischl, Pascal Kerschke, Lars Kotthoff, Marius Lindauer, Yuri Malitsky, Alexandre Fréchette, Holger H. Hoos, Frank Hutter, Kevin Leyton-Brown, Kevin Tierney, and Joaquin Vanschoren. ASlib: A Benchmark Library for Algorithm Selection. **Artificial Intelligence Journal (AIJ)**, 237:41 – 58, 2016. ISSN 0004-3702. URL `https://www.sciencedirect.com/science/article/pii/S0004370216300388`.

[2] Jakob Bossek and Heike Trautmann. Multi-Objective Performance Measurement: Alternatives to PAR10 and Expected Running Time. In **Proceedings of 12th International Conference on Learning and Intelligent Optimization (LION)**, June 2018. Publication status: Accepted.

[3] Jérémie Dubois-Lacoste, Holger H. Hoos, and Thomas Stützle. On the Empirical Scaling Behaviour of State-of-the-art Local Search Algorithms for the Euclidean TSP. In **Proceedings of the 17th Annual Conference on Genetic and Evolutionary Computation (GECCO)**, pages 377 – 384. ACM, 2015. ISBN 978-1-4503-3472-3. doi: 10.1145/2739480.2754747. URL `http://dl.acm.org/citation.cfm?id=2754747`.

[4] Nikolaus Hansen, Anne Auger, Steffen Finck, and Raymond Ros. Real-Parameter Black-Box Optimization Benchmarking 2009: Experimental Setup. Technical Report RR-6828, INRIA, 2009. URL `https://hal.inria.fr/inria-00362649v3/document`.

[5] Keld Helsgaun. General k-opt Submoves for the Lin-Kernighan TSP Heuristic. **Mathematical Programming Computation**, 1(2-3): 119 – 163, 2009. doi: 10.1007/s12532-009-0004-6. URL `https://link.springer.com/article/10.1007/s12532-009-0004-6`.

[6] Pascal Kerschke, Lars Kotthoff, Jakob Bossek, Holger H. Hoos, and Heike Trautmann. Leveraging TSP Solver Complementarity through Machine Learning. **Evolutionary Computation Journal (ECJ)**, pages 1 – 24, 2017. doi: 10.1162/evco_a_00215. URL `https://www.mitpressjournals.org/doi/abs/10.1162/evco_a_00215`.

[7] Yuichi Nagata and Shigenobu Kobayashi. A powerful genetic algorithm using edge assembly crossover for the traveling salesman problem. **INFORMS Journal on Computing**, 25(2): 346 – 363, 2013. doi: 10.1287/ijoc.1120.0506. URL `http://dl.acm.org/citation.cfm?id=2466704`.

[8] John Rischard Rice. The Algorithm Selection Problem. **Advances in Computers**, 15:65 – 118, 1976. doi: 10.1016/S0065-2458(08)60520-3. URL `http://www.sciencedirect.com/science/article/pii/S0065245808605203`.

[9] Xiao-Feng Xie and Jiming Liu. Multiagent Optimization System for Solving the Traveling Salesman Problem (TSP). IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 39(2): 489 – 502, 2009. URL `http://ieeexplore.ieee.org/document/4717264/`.