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.

\[ F : \mathcal{I} \rightarrow \mathcal{F} \subset \mathbb{R}^m \]

\[ f(l) \in \mathcal{F} \subset \mathbb{R}^m \]

\[ S : \mathcal{F} \rightarrow \mathcal{A} \]

\[ p \in \mathbb{R}^n \]

\[ p : \mathcal{A} \times \mathcal{I} \rightarrow \mathbb{R}^n \]
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

• 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.

How do parameters affect the benchmark results?
ALGORITHM SELECTION

- 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.
  \[ \rightsquigarrow \text{How do parameters affect the benchmark results?} \]

Our contribution

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

- Set of problem instances $\mathcal{I} = \{I_1, \ldots, I_n\}$,
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 \( \mathcal{I} = \{I_1, \ldots, I_n\} \),
- Set of algorithms/solvers \( \mathcal{A} = \{A_1, \ldots, A_n\} \),
- \( m > 1 \) independent runs of each \( A \in \mathcal{A} \) on \( I \in \mathcal{I} \)
We consider:

- Set of problem instances \( \mathcal{I} = \{I_1, \ldots, I_{n_\mathcal{I}}\} \),
- Set of algorithms/solvers \( \mathcal{A} = \{A_1, \ldots, A_{n_\mathcal{A}}\} \),
- \( m > 1 \) independent runs of each \( A \in \mathcal{A} \) on \( I \in \mathcal{I} \)
- \( r_{A,I}^1, \ldots, r_{A,I}^m \) empirical runtimes.
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 $I \in \mathcal{I}$
- Empirical runtimes $r_{A,i}^I, \ldots, r_{A,m}^I$.
- Time limit / cutoff time $T \in \mathbb{R}_{>0}$. 
PERFORMANCE MEASURES
Penalized Average Runtime (PAR, [1])

Arithmetic mean of running times, $r_{i}^{A:L}, i \in [m]$; 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**.

\[
\text{PAR}_{A,I}(f) := \frac{1}{m} \sum_{i=1}^{m} \tilde{r}_{i}^{A,I} \quad \text{with} \quad \tilde{r}_{i}^{A,I} = \begin{cases} 
 f \cdot T, & \text{if } r_{i}^{A,I} > T \\
 r_{i}^{A,I}, & \text{otherwise}.
\end{cases}
\]
Penalized Quantile Runtime (PQR)

Replace outlier-sensitive mean by more robust \( p \)-\textit{quantile}, \( p \in (0, 1] \).
Penalized Quantile Runtime (PQR)

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

$$\text{PQR}_{A,i}(p, f) := \begin{cases} f \cdot T, & \text{if } \sum_{i=1}^{m} 1\{r_{i}^{A,i} < T\} < \lfloor mp + 1 \rfloor \\ q_{p}(r_{1}^{A,i}, \ldots, r_{m}^{A,i}), & \text{otherwise.} \end{cases}$$
Penalized Expected Runtime (PERT)

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

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

\[
\text{PERT}_{A,I}(f) = \frac{1}{S} \sum_{j=1}^{S} r_{ij}^{A,I} + \left( \frac{1 - p_s}{p_s} \right) \cdot f \cdot T \\
= \frac{1}{S} \left( \sum_{j=1}^{S} r_{ij}^{A,I} + (m - s) \cdot f \cdot T \right)
\]
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

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>EAX</th>
<th>EAX+restart</th>
<th>LKH</th>
<th>LKH+restart</th>
<th>MAOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RUE (600)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Morphed (600)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Netgen (600)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TSPLIB (22)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VLSI (18)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total (1845)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
RESULTS

![Graphs showing results for different algorithms and datasets](image)

- **Algorithm**
  - EAX
  - EAX+restart
  - LKH
  - LKH+restart
  - MAOS

- **Datasets**
  - RUE (600)
  - Netgen (600)
  - National (5)
  - TSPLIB (22)
  - VLSI (18)
  - Total (1845)

- **Scores**
  - PQR(0.5, f)-Score (scaled by EAX+restart)
  - Penalty Factor f

- **Scores Range**
  - 0 to 4000
  - 0 to 10000
  - 0 to 5000
  - 0 to 3000
  - 0 to 4000
  - 0 to 1000
  - 0 to 2000

- **Penalty Factor Range**
  - 0 to 100

---

12/20
RESULTS

The diagram illustrates the results of various algorithms (RUE, Morphed, Netgen, National, TSPLIB, VLSI, Total) under different conditions. The y-axis represents the Penalty-Factor f, and the x-axis represents the p (used for p-Quantile q_p). The color scale indicates the (log-scaled) PQR(p,f)-Ratio.
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 $\sim$ 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 $\sim$ huge impact of single failed runs.

**Outlook**

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


