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

Pascal Kerschke
University of Münster
Münster, Germany
kerschke@uni-muenster.de

Jakob Bossek
University of Münster
Münster, Germany
bossek@uni-muenster.de

Heike Trautmann
University of Münster
Münster, Germany
trautmann@uni-muenster.de

ABSTRACT

Performance comparisons of optimization algorithms are heavily influenced by the underlying indicator(s). In this paper we investigate commonly used performance indicators for single-objective stochastic solvers, such as the *Penalized Average Runtime (e.g., PAR10)* or the *Expected Running Time (ERT)*, based on exemplary benchmark performances of state-of-the-art inexact TSP solvers. Thereby, we introduce a methodology for analyzing the effects of (usually heuristically set) indicator parametrizations – such as the penalty factor and the method used for aggregating across multiple runs – w.r.t. the robustness of the considered optimization algorithms.

CCS CONCEPTS

• **Computing methodologies** → **Optimization algorithms**; • **Mathematics of computing** → *Combinatorial optimization*;

KEYWORDS

Algorithm Selection, Travelling Salesperson Problem, Transportation, Performance Measures, Optimization

ACM Reference Format:
Pascal Kerschke, Jakob Bossek, and Heike Trautmann. 2018. Parameterization of State-of-the-Art Performance Indicators: A Robustness Study Based on Inexact TSP Solvers. In *GECCO '18 Companion: Genetic and Evolutionary Computation Conference Companion, July 15–19, 2018, Kyoto, Japan,* Jennifer B. Sartor, Theo D'Hondt, and Wolfgang De Meuter (Eds.). ACM, New York, NY, USA, Article 4, 8 pages. https://doi.org/10.1145/3205651.3208233

1 INTRODUCTION

Selecting the best suited solver from a portfolio for an unseen optimization problem (prior to optimization) is termed the *Algorithm Selection Problem* [12]. Much research on combinatorial optimization problems has been conducted, mainly focussing on generating machine-learning-based algorithm selection models, which make use of instance features – see [10] for a general overview and [8] for the most recent study on per-instance-based automated algorithm selection on the Euclidean Traveling Salesperson Problem (TSP).

A crucial part of such sophisticated modeling approaches is a systematic, comprehensive and meaningful benchmark of candidate solvers on the considered instance set(s). Overall ranking of solvers on an instance and across instances requires a suitable and informative performance measure. Specifically for single-objective, inexact and often stochastic solvers those involve considerations regarding robustness (failure rate over several solver runs and / or variability of runtimes on solved instances across runs) and quality (average / best / worst runtimes).

Although commonly used performance indicators are not parameter free, they are often treated as such by using them without thoroughly questioning implicit assumptions. For example, the choice of a penalty factor of ten times the cutoff time in case of the classical PAR10-score [1] is state of the art – but has been heuristically set. Moreover, the way of aggregating runtimes might be varied as well. Both, arithmetic mean and median, are frequently used candidates, but a generalization of the 50%-quantile (i.e., the median) to general p-quantiles allows to vary the required degree of solver robustness across runs. The latter indicator will be introduced as *Penalized Quantile Runtime (PQR)*. In [8], it was used with $p = 0.5$ for the comparison of inexact solvers on the Euclidean TSP. Here, we vary the parametrizations of PAR and PQR, and illustrate their effects based on the benchmark from [8], which is also publicly available in the ASLib [1] (http://www.coseal.net/aslib/). Moreover, we compare their properties to a performance indicator that is predominantly used in single-objective continuous black-box optimization, the *Expected Running Time (ERT, [5])*, and examine the effects of different penalty factors (for unsuccessful runs).

This extensive study presents the means for thoroughly investigating the behaviour of state-of-the-art inexact TSP solvers in terms of robustness across runs and solver quality. Our results support the superiority of EAX+restart within the recent algorithm selection study on TSP [8] – independent from the underlying parametrization of the performance indicators. This is likely caused by the solver's rather low variability and failure rates across its runs.

The presented analysis and visualization therefore provides a structured approach on how to assess the sensitivity of a benchmark study w.r.t. altered requirements in terms of solver robustness across runs. Those requirements can be implicitly set by means of respective performance measure parametrizations.

Section 2 elaborates on the algorithm selection problem in TSP, while Section 3 provides a detailed overview of the performance indicators that are underlying our study – including proposed variations of the latter. The experimental setup is given in Section 4 and respective results are presented in Section 5. Conclusions are drawn in Section 6.

GECCO '18 Companion, July 15–19, 2018, Kyoto, Japan

ACM ISBN 978-1-4503-5764-7/18/07…$15.00
https://doi.org/10.1145/3205651.3208233

2 ALGORITHM SELECTION IN TSP

The *Travelling Salesperson Problem (TSP)* is a well-known combinatorial optimization problem of high practical relevance, e.g., in printed circuit board (PCB) design. Given an undirected complete graph $G = (V, E)$ with node set $V = \{1, \ldots, n\}$ and edge set E, each edge $\{i, j\} \in E, 1 \leq i, j \leq n$ is associated with a scalar positive real-valued distance $d_{ij} \in \mathbb{R}_{>0}$. A Hamiltonian cycle in G is a (salesperson) tour that starts in a source node, visits each node exactly once and returns back to the source. The classical TSP optimization problem deals with finding a Hamiltonian cycle with minimal sum of edge weights among all possible Hamiltonian cycles.

Unfortunately, TSP is $\mathcal{NP}$-hard[1] and thus – as most experts assume – there seems to be no deterministic solver, that solves each problem instance to optimality within a guaranteed polynomial runtime bound. However, researchers developed several very well-performing inexact heuristic approaches over the past decades such as LKH [6], EAX [11] as well as restart variants of them [3], which partially solve even large problem instances within reasonable time and define the state of the art in inexact TSP solving.

There is no free lunch in optimization due to the NFL-theorems by Wolpert and Macready [14] and thus in general different (TSP) solvers exhibit complementary performance on different problem instances. Per-instance automatic algorithm selection (AS) [12] aims for selecting an algorithm from a so-called portfolio of algorithms on a per-instance basis, which most likely will perform best on the instance at hand (see [10] for a survey). AS methods aim to overcome the limitations of single solvers with a machine-learning-based smart meta-layer. Recently, we improved upon state-of-the-art performance in inexact TSP solving utilizing per-instance AS [8, 9]. The working principles of AS-methods rely, among other components, on the choice of the performance measure utilized to assess solver performance. Next, we discuss two commonly used performance measures and introduce variations of them.

3 VARIATIONS OF EXISTING INDICATORS

Let us consider a set of stochastic algorithms $\mathcal{A} = \{A_1, \ldots, A_{n_\mathcal{A}}\}$ and a set of problem instances $\mathcal{I} = \{I_1, \ldots, I_{n_\mathcal{I}}\}$. In the AS-scenario, the performance of a solver on a set of instances is commonly quantified using a scalar performance measure. The computation of the measure is performed in two steps: 1) aggregation over the algorithm runs and 2) aggregation of the aggregated runtimes over the instance set. Here, we focus on the first step as the most crucial part in our view. In the second step, the arithmetic mean is used, although this might be up to discussion as well.

Let $A \in \mathcal{A}$ be a solver and $I \in \mathcal{I}$ a problem instance. Since A is stochastic we apply A on I for $m > 1$ times each, which results in empirical runtimes $r_1^{A,I}, \ldots, r_m^{A,I}$.

Penalized Average Runtime. A common measure used in the field of combinatorial optimization is the so-called *penalized average runtime* (PAR, [1]). Given a maximum runtime $T > 0$, i.e., the so-called *cutoff-time*, and a penalty factor $f > 1$ the PAR-score is defined as follows:

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

[1] This follows directly from the NP-hardness of the Hamiltonian cycle problem [7].

Hence, the measure averages the runtimes of A on I. Runs which do not find the optimal solution until T are penalized with $f \cdot T$.

Penalized Quantile Runtime. Since the arithmetic mean is very prone to outliers, we propose a more robust and parametrizable version denoted as *Penalized Quantile Runtime* (PQR). Here, as the name suggests, we utilize p-quantiles for aggregation in favour of the arithmetic mean. Based on the above definitions and a p-quantile with $p \in (0, 1]$, the performance indicator is defined as:

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

That is, a run is assumed to be unsuccessful (or failed), if less than $p \cdot 100\%$ of the m runs reached the optimum within the runtime limit. Otherwise, the algorithm is considered successful and the measure corresponds to the p-quantile (q_p) of the runtimes $r_i^{A,I}$.

Expected Runtime. The *Expected Runtime (ERT)*, as proposed in [5], is usually adopted in benchmarking single-objective continuous black-box optimization algorithms. It is defined as:

$$\mathrm{ERT}_{A,I} = \frac{1}{s} \sum_{j=1}^{s} r_{i_j}^{A,I} + \left(\frac{1 - p_s}{p_s} \right) \cdot T = \frac{1}{s} \left(\sum_{j=1}^{s} r_{i_j}^{A,I} + (m - s) \cdot T \right),$$

where $s = \sum_{i=1}^{n} \mathbb{1}\{r_i^{A,I} < T\}$ is the number of successful runs, $p_s = s/m$ is the (estimated) probability of success and $i_1, \ldots, i_s$ are the indices of successful runs.

Penalized Expected Runtime. As an extension to the regular ERT, we introduce a slightly modified version, which – similar to PAR and PQR – penalizes failed runs with a penalty factor f:

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

A penalty factor $f = 1$ obviously results in the definition of the regular ERT. Note that in contrast to the rather robust PQR, single runs have a much higher leverage on the overall performance of PAR, ERT and PERT.

4 EXPERIMENTAL SETUP

In the following we analyze different aspects of the performance measures which were introduced in Section 3. First, we take a detailed look at the effect of varying the quantile q of the PQR indicator. Next, keeping q fixed, we address the effect of the penalty factor f. The interaction between both parameters is studied subsequently. Finally, we compare PQR (mainly used in the discrete domain) and PERT (originated in the continuous domain). We base our analyses on the most recent comprehensive performance benchmark [8] of five state-of-the-art inexact TSP solvers which were compared on a set of 1 845 instances out of six TSP benchmark sets.

4.1 TSP Solvers and Benchmarks

The optimization algorithms considered in [8] are a genetic algorithm [4] with edge assembly crossover (EAX, [11]), Helsgaun's

Table 1: This table is a slightly modified version of a table from [8]. It lists for how many instances a solver was the only one to achieve the best performance (unique), tied with others for the best performance (shared), or failed to find an optimal solution (failed). In addition, it also aggregates the performances based on PQR(0.5, 10)-scores (in seconds).

TSP Set	Measure	EAX	EAX +rest.	LKH	LKH +rest.	MAOS
RUE (600)	Unique	52	**66**	223	226	22
	Shared	1	**1**	10	10	0
	Failed	157	**0**	18	2	106
	PQR(0.5, 10)	9430.92	**21.21**	1135.67	159.79	6377.80
VLSI (18)	Unique	3	**5**	4	6	0
	Shared	0	**0**	0	0	0
	Failed	2	**0**	2	0	2
	PQR(0.5, 10)	4004.18	**6.35**	4008.29	47.23	4011.35
TSPLIB (22)	Unique	8	**2**	5	6	1
	Shared	0	**0**	0	0	0
	Failed	5	**1**	2	1	2
	PQR(0.5, 10)	8185.08	**1649.51**	3332.99	1679.79	3282.30
National (5)	Unique	3	**0**	1	1	0
	Shared	0	**0**	0	0	0
	Failed	0	**0**	0	0	1
	PQR(0.5, 10)	4.92	**5.14**	29.03	16.53	7209.53
Netgen (600)	Unique	156	**216**	92	108	20
	Shared	7	**7**	1	1	0
	Failed	77	**0**	18	13	92
	PQR(0.5, 10)	4627.24	**12.99**	1190.55	862.97	5532.35
Morphed (600)	Unique	152	**229**	94	87	27
	Shared	8	**7**	3	4	0
	Failed	114	**0**	22	9	93
	PQR(0.5, 10)	6846.10	**16.72**	1444.99	654.08	5593.25
Total (1845)	Unique	374	**518**	419	434	70
	Shared	16	**15**	14	15	0
	Failed	355	**1**	62	25	296
	PQR(0.5, 10)	6934.81	**36.30**	1305.34	565.85	5789.98

variant of the Lin-Kernighan Heuristic (LKH, [6]), a multi-agent optimization system (MAOS, [15]) and modified versions of EAX and LKH (denoted EAX+restart and LKH+restart), which use an additional restart mechanism as proposed by [3][2]. Each of these five solvers was executed on three artificial TSP benchmarks (RUE, Netgen and Morphed), as well as on three sets of real-world problems (TSPLIB, VLSI and National). Further information on each of these TSP sets is given in [8].

4.2 Benchmark Results

Within [8], we compared the performances of the aforementioned five state-of-the-art inexact TSP solvers on six benchmark sets. Although the performance measures within [8] were called PAR10, their correct notation should have been PQR(0.5, 10). Table 1 summarizes the results per TSP set and solver. It lists 1) how often a solver was the only solver (*unique*) to have the best performance of all five solvers on an instance, and 2) how often two (or more) solvers *shared* the best performance. For instance, for 223 of the 600 RUE instances LKH found the optimal solution faster than any of the other four solvers and for ten further instances, it again found the optimum fastest – but another solver (probably LKH+restart) found the optimum within the same time. In addition, Table 1 shows how often a solver did not find the optimum within the given cutoff time (of 1 hour) meaning less than six out of ten runs were successful on an instance as the 50%-quantile is used. For instance, LKH failed to find the optimum for 18 of the 600 RUE instances.

Performances were also measured using PQR(0.5, 10). According to that score, EAX+restart (highlighted in **bold face**) was the portfolio's single best solver (SBS). That is, on average (across the entire benchmark) it found the optimum of an instance in 36.30*s* and thus much faster than any of the other four solvers. Although the second restart-approach (LKH+restart) found the optimal solution on average approximately 15 times slower (565.85*s*) than the SBS, it clearly outperformed the remaining solvers.

However, it is unclear how much these findings are influenced by the chosen performance measure, PQR(0.5, 10), specifically by the size of the penalty factor and the considered quantile probability. The influence of these parameters will be analyzed in detail within the subsequent results section.

5 RESULTS

5.1 Quantile Effects on the Performance

Table 2 summarizes the effects of varying quantile probabilities on the performance of the five considered TSP solvers. More precisely, in our experiments, the runtimes across the ten runs on an instance are aggregated by five different quantiles q_p, $p \in \{0.10, 0.25, 0.50, 0.75, 0.90\}$. While the 10%-quantile provides a *soft* (= solver-friendly) aggregation approach (only two *successful* runs are needed such that the entire instance is considered to be solved successfully and consequently its PQR-score is based on non-penalized runtimes), the 90%-quantile can be seen as a *hard* threshold as all ten runs need to be solved successfully in order to avoid runtime penalties. In previous studies, the median (i.e., the 50%-quantile) was used by default and we thus highlighted the corresponding performances by solid lines above and below the respective rows.

The left half of Table 2 lists the number of *failed* instances per TSP set, quantile and solver. While it is not surprising that the number of failed instances is non-decreasing for increasing quantiles, one can detect clear differences among the solvers. Noticeably, the two restart solvers (EAX+restart and LKH+restart) are very robust solvers, as the number of failed instances increases – if at all – just by a few instances for growing quantiles (apart from LKH+restart on the more structured instance sets Netgen and Morphed). In contrast to that, the performances of the remaining solvers much stronger depend on the quantiles and – especially in case of EAX and MAOS – rapidly grow for the two larger considered quantiles (75% and 90%). So, the current standard approach from the literature (i.e., using the median) seems to be rather solver-friendly. For instance, an increase from the 50%- to 75%-quantile leads (on average across all instances) to an increase of 65% (from 355 to 597 instances) to 80% (296 to 528) of failed instances for EAX and MAOS, respectively. Similarly, an increase from the 75%- to the

[2] As our study focusses on the parametrization of performance indicators, we applied our method to an existing, publicly available data set (http://www.coseal.net/aslib/) and hence did not consider recent developments in inexact TSP solving such as [13].

Table 2: Influence of quantiles q_p, $p \in \{0.10, 0.25, 0.50, 0.75, 0.90\}$, used for aggregating the runs of an instance. Their impacts are measured per solver and TSP set w.r.t. the number of failed instances (left), and the corresponding PQR(p, 10)-scores (right).

TSP Set	Quantile	Number of Failed Instances						PQR(p, 10)-Score					
		EAX	EAX+rest.	LKH	LKH+rest.	MAOS	Total	EAX	EAX+rest.	LKH	LKH+rest.	MAOS	Total
RUE	0.10	31	0	0	0	23	600	1868.9	11.3	21.9	14.0	1398.2	5.1
	0.25	78	0	9	2	56	600	4688.0	14.1	569.1	139.6	3376.4	7.0
	0.50	157	0	18	2	106	600	9430.9	21.2	1135.7	159.8	6377.8	10.7
	0.75	243	0	29	3	182	600	14584.9	34.4	1843.7	240.8	10932.9	17.6
	0.90	413	1	60	5	333	600	24781.9	106.4	3702.8	381.4	19987.3	27.2
VLSI	0.10	0	0	0	0	0	18	4.8	4.8	84.1	10.1	12.0	3.1
	0.25	1	0	1	0	1	18	2004.6	5.3	2141.7	17.7	2011.0	3.6
	0.50	2	0	2	0	2	18	4004.2	6.4	4008.3	47.2	4011.4	4.4
	0.75	4	0	2	0	4	18	8003.8	9.1	4045.3	150.9	8010.9	6.9
	0.90	9	0	3	1	7	18	18002.3	10.8	6098.8	2040.8	14008.4	8.6
TSPLIB	0.10	2	1	1	1	1	22	3276.6	1642.0	1655.3	1647.2	1645.8	3.2
	0.25	3	1	1	1	2	22	4912.7	1643.8	1713.1	1653.9	3281.9	6.2
	0.50	5	1	2	1	2	22	8185.1	1649.5	3333.0	1679.8	3282.3	10.8
	0.75	6	1	2	1	5	22	9821.0	1662.1	3371.1	1745.2	8190.5	23.7
	0.90	9	1	3	1	9	22	14729.7	1671.1	4960.7	1792.9	14733.3	31.6
National	0.10	0	0	0	0	0	5	4.6	4.6	6.1	5.0	10.3	3.4
	0.25	0	0	0	0	0	5	4.8	4.8	9.9	10.7	10.5	3.7
	0.50	0	0	0	0	1	5	4.9	5.1	29.0	16.5	7209.5	4.1
	0.75	1	0	0	0	2	5	7204.6	7.5	69.4	27.1	14408.7	5.3
	0.90	3	0	0	0	4	5	21603.1	8.9	160.2	34.3	28802.1	5.8
Netgen	0.10	11	0	4	3	25	600	668.2	8.8	285.1	221.8	1514.5	6.8
	0.25	37	0	9	7	52	600	2227.9	10.0	612.8	480.9	3133.7	7.9
	0.50	77	0	18	13	92	600	4627.2	13.0	1190.6	863.0	5532.4	10.3
	0.75	151	0	28	15	152	600	9066.1	19.5	1849.8	1039.8	9130.5	14.2
	0.90	300	0	54	25	310	600	18003.3	28.2	3399.6	1655.8	18605.5	20.8
Morphed	0.10	19	0	6	1	28	600	1147.1	9.5	408.5	109.5	1694.1	6.7
	0.25	61	0	14	5	55	600	3666.5	11.9	908.1	369.4	3313.7	8.4
	0.50	114	0	22	9	93	600	6846.1	16.7	1445.0	654.1	5593.3	11.4
	0.75	192	0	38	16	183	600	11526.8	25.1	2473.0	1108.7	10990.2	18.0
	0.90	335	0	75	27	343	600	20102.5	35.4	4660.0	1794.7	20584.9	27.1
Total	0.10	63	1	11	5	77	1845	1237.3	29.2	253.3	132.0	1517.9	6.1
	0.25	180	1	34	15	166	1845	3519.6	31.3	721.0	341.9	3253.5	7.7
	0.50	355	1	62	25	296	1845	6934.8	36.3	1305.3	565.9	5790.0	10.7
	0.75	597	1	99	35	528	1845	11654.6	45.6	2085.2	799.4	10313.6	16.6
	0.90	1069	2	195	59	1006	1845	20861.1	75.3	3944.2	1287.5	19635.2	24.9

90%-quantile even yields an additional 80% to 90% increase of failed instances.

Figure 1 visually supports the previous findings: while the number of failed instances for EAX+restart and LKH+restart increases only slightly, the curves of EAX and MAOS are rather steep for the larger quantiles. Interestingly, LKH (without additional restart mechanism) behaves more similar to the two superior restart variants. Note that we omitted the corresponding plot for the set of *National* instances, as it consisted of only five instances and additionally did not reveal any noteworthy patterns.

Based on the findings from above, we suggest to use the previously described approaches (of summarizing the number of failed instances per solver depending on different quantiles) for analyzing the robustness of the respective solvers.

In addition to the number of failed instances, Table 2 also summarizes the PQR(p, 10)-scores of the respective triplets of TSP set, quantile and solver. Observing the results of EAX+restart, which exhibits – with the exception of the RUE data set – a constant number of failed instances per TSP set, one can confirm that the runtimes (and thereby the corresponding PQR-scores) continuously grow with increasing quantiles. Of course, this is a very intuitive finding, but it nonetheless confirms the expectations.

Comparing the PQR(p, 10)-scores to the corresponding number of failed instances reveals that the number of failed instances have a strong leverage on the PQR(p, 10)-scores. For instance, in case of the VLSI benchmark, each failed instance counts 2 000s (penalty score of 36 000s / 18 instances within the TSP set) towards the PQR(p, 10)-score. In case of the entire data, every failed instance increases the PQR(p, 10)-score by roughly 20s (= 36 000s / 1 845 instances).

5.2 Impact of the Penalty-Factor

We now investigate the influence of the penalty factors, which are used for penalizing the failed runs. For this purpose, we fix the considered quantile to the default value of 50% and then compare the PQR-scores depending on the varied penalty factors. The corresponding numbers are listed in the left half of Table 3. Noticeably, the PQR-scores for EAX+restart are *always constant* within a single TSP set, except for the TSPLIB as it is the only instance set for which

Table 3: Influence of penalty factors $f \in \{1, 2, 5, 10, 20, 50, 100\}$ on the PQR(0.5, f)- and PERT(f)-scores of the solvers (left and right, respectively). For the former approach the scores reveal that the penalty factors influence the results only when (aggregated) failed instances exist among the considered TSP instances. In contrast to that, the scores based on the PERT-approach are already contaminated by failed (single) runs.

TSP Set	Quantile	Aggregation by means of the 50%-quantile						Aggregation by means of the PERT					
		EAX	EAX+rest.	LKH	LKH+rest.	MAOS	∅	EAX	EAX+rest.	LKH	LKH+rest.	MAOS	∅
RUE	1	952.9	**21.2**	163.7	51.8	653.8	10.7	3711.7	**28.1**	401.1	85.2	2714.1	15.8
	2	1894.9	**21.2**	271.7	63.8	1289.8	10.7	7410.0	**28.7**	708.8	118.6	5405.7	15.8
	5	4720.9	**21.2**	595.7	99.8	3197.8	10.7	18504.9	**30.7**	1632.0	218.6	13480.2	15.8
	10	9430.9	**21.2**	1135.7	159.8	6377.8	10.7	36996.3	**34.1**	3170.7	385.2	26937.8	15.8
	20	18850.9	**21.2**	2215.7	279.8	12737.8	10.7	73979.2	**40.7**	6248.1	718.6	53853.1	15.8
	50	47110.9	**21.2**	5455.7	639.8	31817.8	10.7	184927.7	**60.7**	15480.2	1718.6	134598.8	15.8
	100	94210.9	**21.2**	10855.7	1239.8	63617.8	10.7	369842.0	**94.1**	30867.1	3385.2	269175.0	15.8
VLSI	1	404.2	**6.4**	408.3	47.2	411.4	4.4	1382.9	**7.5**	1273.6	96.6	1251.3	5.8
	2	804.2	**6.4**	808.3	47.2	811.4	4.4	2760.7	**7.5**	2395.8	118.9	2489.4	5.8
	5	2004.2	**6.4**	2008.3	47.2	2011.4	4.4	6894.0	**7.5**	5762.5	185.5	6203.7	5.8
	10	4004.2	**6.4**	4008.3	47.2	4011.4	4.4	13782.9	**7.5**	11373.6	296.6	12394.2	5.8
	20	8004.2	**6.4**	8008.3	47.2	8011.4	4.4	27560.7	**7.5**	22595.8	518.9	24775.1	5.8
	50	20004.2	**6.4**	20008.3	47.2	20011.4	4.4	68894.0	**7.5**	56262.5	1185.5	61918.0	5.8
	100	40004.2	**6.4**	40008.3	47.2	40011.4	4.4	137782.9	**7.5**	112373.6	2296.6	123822.7	5.8
TSPLIB	1	821.4	**176.8**	387.5	207.1	336.8	10.8	4172.3	**1654.3**	1950.1	1705.6	2373.3	16.0
	2	1639.6	**340.4**	714.8	370.7	664.1	10.8	8340.5	**3290.7**	3791.0	3341.9	4736.3	16.0
	5	4094.2	**831.3**	1696.6	861.6	1645.9	10.8	20845.1	**8199.8**	9313.8	8251.0	11825.3	16.0
	10	8185.1	**1649.5**	3333.0	1679.8	3282.3	10.8	41686.0	**16381.6**	18518.3	16432.8	23640.2	16.0
	20	16366.9	**3285.9**	6605.7	3316.2	6555.0	10.8	83367.8	**32745.2**	36927.4	32796.5	47270.1	16.0
	50	40912.4	**8195.0**	16423.9	8225.2	16373.2	10.8	208413.3	**81836.1**	92154.7	81887.4	118159.7	16.0
	100	81821.4	**16376.8**	32787.5	16407.1	32736.8	10.8	416822.3	**163654.3**	184200.1	163705.6	236309.1	16.0
National	1	4.9	**5.1**	29.0	16.5	729.5	4.1	744.9	**6.6**	66.6	21.5	1299.4	4.8
	2	4.9	**5.1**	29.0	16.5	1449.5	4.1	1484.9	**6.6**	66.6	21.5	2588.0	4.8
	5	4.9	**5.1**	29.0	16.5	3609.5	4.1	3704.9	**6.6**	66.6	21.5	6453.7	4.8
	10	4.9	**5.1**	29.0	16.5	7209.5	4.1	7404.9	**6.6**	66.6	21.5	12896.6	4.8
	20	4.9	**5.1**	29.0	16.5	14409.5	4.1	14804.9	**6.6**	66.6	21.5	25782.3	4.8
	50	4.9	**5.1**	29.0	16.5	36009.5	4.1	37004.9	**6.6**	66.6	21.5	64439.4	4.8
	100	4.9	**5.1**	29.0	16.5	72009.5	4.1	74004.9	**6.6**	66.6	21.5	128868.0	4.8
Netgen	1	469.2	**13.0**	218.6	161.0	564.4	10.3	1794.6	**16.6**	598.4	430.5	2574.3	13.5
	2	931.2	**13.0**	326.6	239.0	1116.4	10.3	3580.0	**16.6**	1038.7	730.3	5131.8	13.5
	5	2317.2	**13.0**	650.6	473.0	2772.4	10.3	8936.2	**16.6**	2359.5	1629.7	12804.1	13.5
	10	4627.2	**13.0**	1190.6	863.0	5532.4	10.3	17863.3	**16.6**	4560.8	3128.8	25591.4	13.5
	20	9247.2	**13.0**	2270.6	1643.0	11052.4	10.3	35717.3	**16.6**	8963.4	6126.9	51165.9	13.5
	50	23107.2	**13.0**	5510.6	3983.0	27612.4	10.3	89279.4	**16.6**	22171.3	15121.1	127889.5	13.5
	100	46207.2	**13.0**	10910.6	7883.0	55212.4	10.3	178549.7	**16.6**	44184.4	30111.6	255762.1	13.5
Morphed	1	690.1	**16.7**	257.0	168.1	571.3	11.4	2709.7	**20.9**	817.7	339.4	2811.9	16.3
	2	1374.1	**16.7**	389.0	222.1	1129.3	11.4	5406.3	**20.9**	1462.3	534.7	5607.2	16.3
	5	3426.1	**16.7**	785.0	384.1	2803.3	11.4	13496.4	**20.9**	3395.9	1120.3	13993.1	16.3
	10	6846.1	**16.7**	1445.0	654.1	5593.3	11.4	26979.9	**20.9**	6618.7	2096.4	27969.7	16.3
	20	13686.1	**16.7**	2765.0	1194.1	11173.3	11.4	53946.8	**20.9**	13064.1	4048.5	55922.8	16.3
	50	34206.1	**16.7**	6725.0	2814.1	27913.3	11.4	134847.5	**20.9**	32400.6	9904.9	139782.0	16.3
	100	68406.1	**16.7**	13325.0	5514.1	55813.3	11.4	269682.0	**20.9**	64628.0	19665.7	279547.5	16.3
Total	1	700.7	**18.7**	216.6	126.8	591.9	10.7	2737.1	**41.1**	626.8	299.4	2678.3	15.1
	2	1393.3	**20.7**	337.5	175.6	1169.5	10.7	5462.6	**60.9**	1112.6	491.0	5338.1	15.1
	5	3471.4	**26.5**	700.5	321.9	2902.2	10.7	13638.9	**120.0**	2569.9	1065.7	13317.4	15.1
	10	6934.8	**36.3**	1305.3	565.9	5790.0	10.7	27266.1	**218.7**	4998.7	2023.4	26616.3	15.1
	20	13861.6	**55.8**	2515.1	1053.7	11565.6	10.7	54520.4	**416.0**	9856.3	3938.9	53214.0	15.1
	50	34642.1	**114.4**	6144.4	2517.1	28892.4	10.7	136283.5	**1007.8**	24429.1	9685.5	133007.3	15.1
	100	69276.3	**211.9**	12193.1	4956.1	57770.5	10.7	272555.3	**1994.3**	48717.1	19263.1	265996.1	15.1

EAX+restart failed on one instance (Table 2) – all other instances were solved successfully.

Table 3 also reveals that the impact of the penalty factor is close to linear. That is, if one compares the PQR(0.5, 1)- and PQR(0.5, 10)-scores, they will roughly differ by factor 10. The same holds true for a comparison of PQR(0.5, 10) to PQR(0.5, 100).

Investigating the penalized performance scores of EAX+restart across the entire benchmark (TSP set = Total), a linear relationship between the penalty factors and the PQR-scores can be observed. Neglecting any penalty-effects, i.e., looking at PQR(1, q), the impact of a single failed instance is approximately $2s$ (= 3 600s / 1 845 instances). This impact grows to $4s$ (for penalty factor 2), $10s$ ($f = 5$), $20s$ ($f = 10$), etc. and thus perfectly aligns with the observed increase in (penalized) performances. In consequence, one

can conclude that the number of failed instances, as well as the penalty factor linearly influence the respective PQR-scores.

These linear relationships also become visible in Figure 2, which displays the growth in (penalized) performances depending on the size of the penalty factors for all but the Morphed instance set (which provide qualitatively similar results as Netgen). For better comparability and visibility of the results, all PQR-scores were scaled by the respective performance of EAX+restart. As a result, the curves of EAX+restart (dashed blue lines) constantly exhibit a value of 1.0, whereas all other curves are *linearly* growing with increasing penalty factors. Again, the only exception to these linear relationships are the results of the TSPLIB as in that scenario the

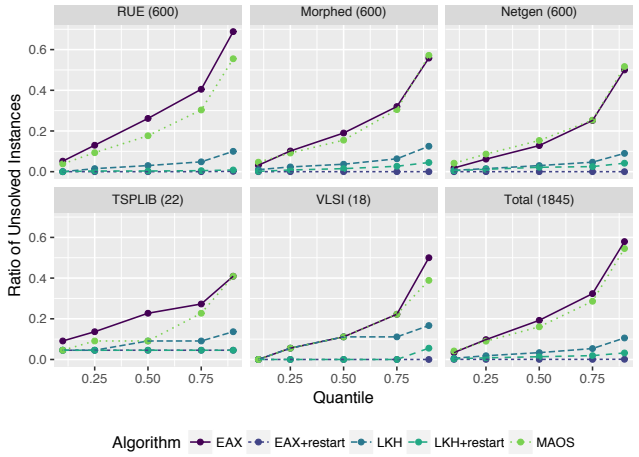

Figure 1: Ratio of unsolved TSP instances per solver and TSP set, and subject to five different quantiles q_p with $p \in \{0.10, 0.25, 0.50, 0.75, 0.90\}$. While EAX+restart and LKH+restart solve most instances constantly (i.e., robust and independent of the quantile), EAX and MAOS perform poorly when using the 90%-quantile for aggregating the runs.

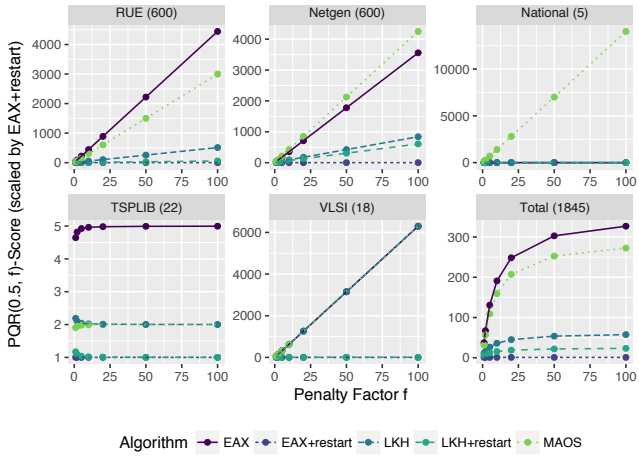

Figure 2: Impact of the penalty factors $f \in \{1, 2, 5, 10, 20, 50, 100\}$ on the PQR-scores. All performances were a priori scaled by the corresponding ones of EAX+restart.

results of EAX+restart are also influenced by a failed instance. When comparing e.g., the performances of EAX+restart and LKH+restart on the TSPLIB, both solvers have an identical amount of failed instances (1) within that TSP set and therefore, the standardization of (penalized) performances basically displays the differences in the solvers' runtimes on all remaining 21 successfully solved instances.

5.3 Joint Impact of Quantile and Penalty Factor

So far, we have analyzed the effects of the quantiles (see Section 5.1) and penalty factors (Section 5.2), separately. Now we have a closer look at their combined influence. We computed the performances per optimization algorithm and TSP set for each pair of penalty factor and quantile, standardized the performances by the respective values of EAX+restart and visualized the resulting ratios as heatmaps in Figure 3. Due to the large magnitudes of the ratios, their values are visualized in log-scale for better recognizability. Within each of the single heatmaps, the previously discussed values are highlighted: the impact of the quantiles based on a fixed penalty factor of 10 (Section 5.1) are framed by white dashed lines, and the influence of the penalty factors, given an aggregation by means of the median (Section 5.2) are highlighted by white dotted lines.

The heatmaps reveal the following: First, the second row of heatmaps (i.e., the ones for EAX+restart) are entirely purple. This is plausible as all PQR values were scaled by the performances of EAX+restart and thus, all ratios within those heatmaps have a value of 1. Second, for some data sets (such as National or VLSI), the quantiles have a larger impact on the PQR scores than the penalty factors. For instance, the first three columns of EAX on the National instances are completely of purple color, indicating that the performance of EAX is quite comparable to the one of EAX+restart on these instances. The tiles of the following two columns are colored between green and yellow, indicating that the performance of EAX strongly decreases compared to EAX+restart. However, for the penalty factor a similar trend can be observed: the larger the penalty factor, the brighter are the corresponding colors (see for instance the heatmap of LKH on the Netgen data).

The heatmaps do not only reveal linear trends per penalty factor or quantile (separately), but one can also observe "diagonal" trends in the coloring from the bottom left (small quantile and penalty factor) to the top right (large quantile and penalty factor) of a heatmap. For instance, EAX and MAOS show such a pattern on Netgen or VLSI. Thus, such solvers perform much more competitive to EAX+restart in case of solver-friendly aggregations (by means of small quantiles) and no or low penalization of the failed instances than for large quantiles and high penalty scores.

Interestingly, the results on the RUE data reveal U-shaped patterns (for all solvers) implying that for extreme quantiles (10% and 90%), the solvers perform more similar to EAX+restart (on this benchmark) than for the intermediate quantiles (25% to 75%).

5.4 Comparison of PERT(f) and PQR(0.5, f)

Performances of combinatorial solvers are usually measured in actual runtimes (i.e., they are measured in time units) and afterwards aggregated by PAR- or PQR-scores. In contrast to that, performances of solvers of continuous optimization problems are usually

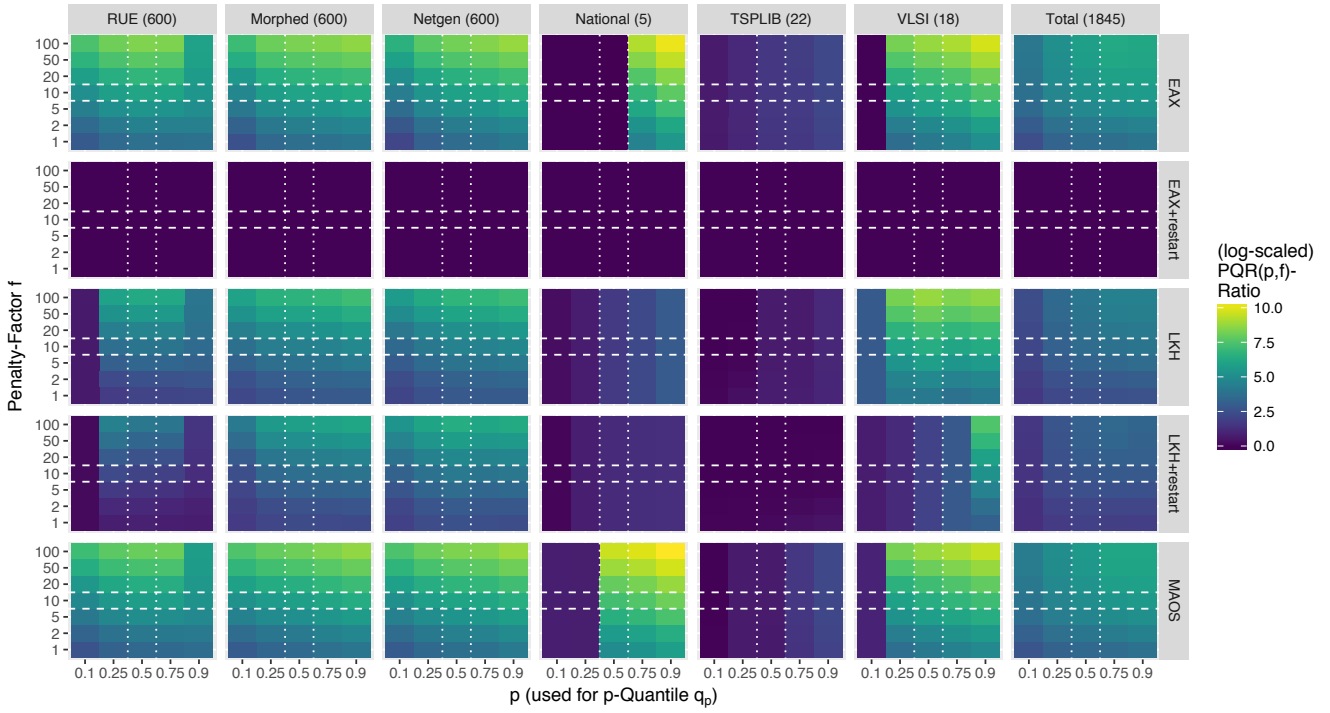

Figure 3: Heatmap showing the joint impact of penalty factors and quantiles. The performances were standardized by the performances of EAX+restart and for better visibility, the coloring of the heatmap is based on the log-scaled values of the performance ratios. The heatmaps are created per optimization algorithm (rows) and pair of TSP set (columns).

measured by means of function evaluations and afterwards evaluated using the ERT. The reasoning for using function evaluations rather than actual runtimes is that 1) for real-world problems a single evaluation can be quite costly, and 2) runtimes mainly depend on the computational resources and thus are incomparable across different machines; even on the same machine, the runtimes for several runs of identical experiments will likely not be identical.

However, the definition of the ERT does not restrict its applicability to *continuous* optimization problems; one simply needs the solver's runtime (actual runtime, function evaluations, etc.) and runstatus (successful / unsuccessful) per run. Hence, one could also assess the performances of the TSP solvers by means of the ERT.

A benefit of using the ERT instead of the previously analyzed PQR(0.5, 10)-approach is that every single run counts – and thereby impacts the solver's performance on that instance. While aggregating all runs of an instance by means of the median (or any other quantile) provides a very robust measure – it basically uses the runtimes of only one or two runs – the magnitudes of all the other runs are completely ignored. In contrast to that, the ERT, which was defined as $\mathrm{ERT}_{A,I} = (\sum_{j=1}^{s} r_{i_j}^{A,I} + (m-s)\cdot T)/s$ (see Section 3), uses the information of *all* runtimes as its numerator is the sum of the runtimes (including cutoff time T for each failed run). As a consequence of this, the ERT is not robust. More precisely, it is very prone to changes within the runtimes, especially to more extreme observations such as failed runs. Even worse, every single unsuccessfully finished run penalizes the ERT twice: on the one hand, the numerator grows (each failed run costs at least the given cutoff time, which is larger than the runtime of any successfully completed run), and on the other hand the denominator decreases (as s is the sum of successful runs).

In addition to that, we also analyzed whether a penalized version of the ERT, i.e., $\mathrm{PERT}_{A,I}(f) = (\sum_{j=1}^{s} r_{i_j}^{A,I} + (m-s)\cdot f\cdot T)/s$, makes sense. The right half of Table 3 summarizes the PERT(f)-values. Obviously, the performances for penalty factor $f = 1$ correspond to the classical usage of ERT – i.e., without any additional penalization. As one can see for the values of EAX+restart, the values do not change if all runs were solved successfully. However, once a single run failed (as it occurs within the RUE data), the penalty factor influences the results. Furthermore, the PERT(f)-scores are larger than the ones from the previously described PQR(0.5, f)-approach. This finding is not very surprising as the PERT can be much larger, due to the denominator of the ERT computation. In case of a fixed cutoff time of 3 600s and $m = 10$ runs, the ERT exhibits values between 0s and up to 36 000s. More generally, the upper boundary of the PERT can be computed via cutoff time $T \times$ #runs $m \times$ penalty factor f.

Figure 4 visually compares the effects of the different penalty factors 1, 2, 5, 10, 20, 50 and 100 on the performances of EAX+restart (purple curve) and LKH+restart (green) for the two previously described approaches, i.e., median (solid line with a bullet for each observed value) and ERT (dashed line with triangles). Again, for better comparability of the results, the performances were first standardized by the ones of EAX+restart under the median approach. As discussed before, the ERT values (dashed lines) are larger than the corresponding ones of the median approach. Also, in case the

algorithms solved all runs of the entire benchmark (as it is the case for the *National* benchmark), the penalty factors have no influence on the performances and thus, all four curves are horizontal lines. Noticeably, the linear leverage of the penalty factors on the performances, which we already detected and discussed in Section 5.2 for the median-approach, also exists for the ERT approach (see e.g., the results of LKH+restart on RUE, Netgen or VLSI).

6 CONCLUSIONS

Parametrizations of the penalized quantile runtime as a generalization of penalized average runtime (PAR) and the expected running time (ERT) were systematically investigated on a comprehensive benchmark study of inexact TSP solvers.

It could be shown that the quantile used for aggregating runtimes across solver runs on an instance substantially influences the robustness requirement of a solver, i.e., larger quantiles require solvers to successfully complete a higher percentage of runs. Thus, the performance assessment of robust solvers, such as EAX+restart, which rarely fails on any instance, will be much less influenced by the choice of the quantile. In fact, no parametrization changed the result of EAX+restart being the single best solver within the focussed TSP benchmark study. Varying the penalty factor of course only influences failed runs but allows for altering the leverage they have regarding differences in algorithm performance between solvers which is important for instance-based automated algorithm selection selection techniques.

Transferring the concept of ERT to combinatorial optimization problems in terms of relying on runtimes rather than function evaluations as commonly used in continuous black-box optimization is possible. However, one should be aware that ERT is much more prone to the results of single runs than the more robust PQR-approach as results of all runs are included. Failed runs thus have a huge impact on the ERT performance – which increases even more in case a penalty factor is used.

Future studies will focus on theoretically investigating performance indicator properties as well as on introducing alternative and robust indicators using the insights of this study. This could also lead to taking a multi-objective perspective onto performance measurement, e.g., by directly investigating the trade-off between the fraction of failed runs and the average runtime of a solver [2]. Moreover, results will be discussed within the context of automated algorithm selection and configuration on TSP. Applications to the continuous domain are of interest as well.

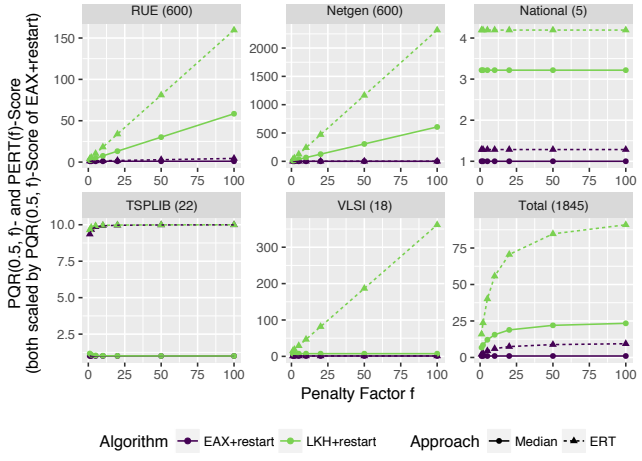

Figure 4: Comparison of the influences of the penalty factors on the performance scores, shown for two approaches – the 'classical' median-PQR-approach (solid line) and the ERT (dashed) – and two optimization algorithms, EAX+restart (purple) and LKH+restart (green).

ACKNOWLEDGMENTS

The authors acknowledge support from the *European Research Center for Information Systems* (ERCIS, https://www.ercis.org/) and the DAAD PPP project No. 57314626.

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. 2016. ASlib: A Benchmark Library for Algorithm Selection. *Artificial Intelligence Journal (AIJ)* 237 (2016), 41 – 58. https://www.sciencedirect.com/science/article/pii/S0004370216300388

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

[3] Jérémie Dubois-Lacoste, Holger H. Hoos, and Thomas Stützle. 2015. 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)*. ACM, 377 – 384. https://doi.org/10.1145/2739480.2754747

[4] Ágoston E. Eiben and James E. Smith. 2015. *Introduction to Evolutionary Computing*. Springer. https://doi.org/10.1007/978-3-662-44874-8

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

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

[7] Richard M. Karp. 1972. *Reducibility among Combinatorial Problems*. Springer, Boston, MA, USA, 85 – 103. https://doi.org/10.1007/978-1-4684-2001-2_9

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

[9] Lars Kotthoff, Pascal Kerschke, Holger H. Hoos, and Heike Trautmann. 2015. Improving the State of the Art in Inexact TSP Solving Using Per-Instance Algorithm Selection. In *Proceedings of 9th International Conference on Learning and Intelligent Optimization*. Springer, 202 – 217. https://doi.org/10.1007/978-3-319-19084-6_18

[10] Lars Kotthoff. 2014. Algorithm Selection for Combinatorial Search Problems: A Survey. *AI Magazine* 35, 3 (2014), 48 – 60. https://doi.org/10.1609/aimag.v35i3.2460

[11] Yuichi Nagata and Shigenobu Kobayashi. 2013. A Powerful Genetic Algorithm Using Edge Assembly Crossover for the Traveling Salesman Problem. *INFORMS Journal on Computing* 25, 2 (2013), 346 – 363. https://doi.org/10.1287/ijoc.1120.0506

[12] John Rischard Rice. 1976. The Algorithm Selection Problem. *Advances in Computers* 15 (1976), 65 – 118. https://doi.org/10.1016/S0065-2458(08)60520-3

[13] Danilo Sanches, L. Darrell Whitley, and Renato Tinós. 2017. Building a Better Heuristic for the Traveling Salesman Problem: Combining Edge Assembly Crossover and Partition Crossover. In *Proceedings of the 19th Annual Conference on Genetic and Evolutionary Computation (GECCO)*. ACM, 329 – 336. https://doi.org/10.1145/3071178.3071305

[14] David Hilton Wolpert and William G. Macready. 1997. No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation (TEVC)* 1, 1 (1997), 67 – 82. https://doi.org/10.1109/4235.585893

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