Applied Soft Computing Journal 88 (2020) 105901

Contents lists available at ScienceDirect

Applied Soft Computing Journal

journal homepage: www.elsevier.com/locate/asoc

A multi-objective perspective on performance assessment and automated selection of single-objective optimization algorithms

Jakob Bossek *, Pascal Kerschke, Heike Trautmann

Information Systems and Statistics, University of Münster, Leonardo-Campus 3, 48149 Münster, Germany

ARTICLE INFO

Article history:
Received 11 April 2019
Received in revised form 9 October 2019
Accepted 29 October 2019
Available online 2 November 2019

Keywords:
Algorithm selection
Multi-objective optimization
Performance measurement
Combinatorial optimization
Traveling Salesperson Problem

ABSTRACT

We build upon a recently proposed multi-objective view onto performance measurement of single-objective stochastic solvers. The trade-off between the fraction of failed runs and the mean runtime of successful runs – both to be minimized – is directly analyzed based on a study on algorithm selection of inexact state-of-the-art solvers for the famous Traveling Salesperson Problem (TSP). Moreover, we adopt the hypervolume indicator (HV) commonly used in multi-objective optimization for simultaneously assessing both conflicting objectives and investigate relations to commonly used performance indicators, both theoretically and empirically. Next to Penalized Average Runtime (PAR) and Penalized Quantile Runtime (PQR), the HV measure is used as a core concept within the construction of per-instance algorithm selection models offering interesting insights into complementary behavior of inexact TSP solvers.

1. Introduction

Predicting the best suited solver for an instance of an optimization problem at hand out of a candidate solver portfolio is a challenging task and commonly denoted as the algorithm selection problem [1,2]. Apart from automatically and ideally cheaply computable features characterizing the problem instances, algorithm selection models require a systematic and representative benchmark study of the respective candidate solvers [3,4]. However, performance measurement of solvers is not totally straightforward and the benchmark results naturally depend on the kind(s) of performance indicator(s) (PI) used.

In combinatorial optimization, the Penalized Average Runtime (PAR, e.g., [5]) or the Penalized Quantile Runtime (PQR, [6,7]) are widely used, while continuous black-box optimization benchmarks such as BBOB [8] per default are based on the Expected Running Time (ERT, [9]). All these indicators build upon assessing success of solver runs, although the definition of success usually depends on the given task. In case of the Traveling Salesperson Problem (TSP), where a set of n locations – usually denoted as cities – and pairwise distances between the latter are given, the objective is to find the shortest round-trip through all considered cities. Thus, a TSP solver might be considered successful, if it solved a given TSP instance up to optimality within a predefined time limit. In contrast, success in single-objective continuous optimization by default denotes approximating the optimum up to a predefined precision limit. For solvers of stochastic nature, apart from the focused fraction of (un)successful repeated solver runs on an instance, the second central aspect considered usually is the average quality of successful runs measured by running time, or number of function evaluations, until the success criterion is met. Thus, performance measurement of stochastic solvers inherently examines solver robustness by simultaneously addressing both solver success and runtime. However, the single-objective indicators such as PAR or ERT offer a scalarized version which incorporates both perspectives but does not allow to directly analyze the trade-off between average performance and robustness.

In this paper, we handle the fraction of unsuccessful runs and the mean running time of successful runs separately and take a multi-objective view onto them following our proposal in [10]. As both objectives should be minimized simultaneously, the algorithm selection is turned into a multi-objective decision problem, allowing to assess the trade-off behavior of the considered solvers on different aggregation levels, e.g., (i) individually per instance, or (ii) per instance set, or even (iii) aggregated across the whole benchmark. Specifically, we will see that the state-of-the-art solver EAX [11] on TSP, enhanced by a restart strategy [12], shows high performance regarding both objectives and thus often dominates competing solvers, in the Pareto sense [13].

We will suggest the unary dominated hypervolume indicator (HV, [14]) as a suitable measure reflecting solver performance in the respective bi-objective space which directly operates on

* Corresponding author.
E-mail addresses: bossek@wi.uni-muenster.de (J. Bossek), kerschke@uni-muenster.de (P. Kerschke), trautmann@wi.uni-muenster.de (H. Trautmann).

https://doi.org/10.1016/j.asoc.2019.105901
1568-4946/

the objectives of interest related to the idea of automated multi-objective configuration presented in [15]. And although we do not claim that our proposed approach always results in superior performances[1] compared to established indicators such as PAR10, it provides a complementary alternative to measure and assess algorithm performances. Moreover, the underlying objectives can easily be replaced — making HV a flexible and generalizable alternative to common performance indicators.

Further, commonly used indicators are not parameter-free – see, e.g., the penalty factor used in the PAR score – however they are often treated as such [7]. Although the HV indicator also possesses a configurable parameter – its reference point – it can be set in a straightforward manner to the maximum per objective, compared to the rather arbitrary selection of the penalty factor.

Our contributions are as follows: Building upon our previous work [10], we (1) specifically investigate theoretical dependencies and relationships of the HV measure to common performance indicators such as PAR10, (2) point our advantages and disadvantages of common measures, and (3) use HV as the underlying performance measure for constructing an algorithm selection model on inexact TSP solvers building upon a recent study conducted in [6] based on the PQR measure. It will be shown, that the proposed HV measure is a linear transformation of the PAR score offering an alternative perspective on algorithm performance. Furthermore, the model presented here not only illustrates an intuitive measure for the natural multi-objective view at solver performances. In fact, the model also yields qualitatively comparable results while simultaneously incorporating a lower number of features. In addition, our multi-objective view provides more insights into the strengths and weaknesses of the portfolio's solvers. The multi-objective HV concept should therefore complement the commonly applied and accepted set of performance indicators. Note that it – in contrast to the established performance measures – easily generalizes to other kinds and numbers of objective combinations apart from average running time and fraction of unsuccessful runs.

Section 2 provides details regarding common performance measures together with corresponding configuration challenges and Section 3 profoundly introduces the multi-objective perspective on solver performance. Our case study on automated algorithm selection on TSP is presented in Section 4. Conclusions are drawn in Section 5 complemented by an outlook on future research perspectives.

2. Performance measurement

In this section we briefly review concepts of measuring performances. Prior to this, we introduce some mathematical vocabulary: let $\mathcal{A} = \{A_1, \ldots, A_{n(\mathcal{A})}\}$ be a set of stochastic algorithms and $\mathcal{I} = \{I_1, \ldots, I_{n(\mathcal{I})}\}$ a set of problem instances. Since the considered algorithms are stochastic, each algorithm $A \in \mathcal{A}$ is executed $m > 1$ times on instance $I \in \mathcal{I}$, which results in empirical running times $r_1^{A,I}, \ldots, r_m^{A,I}$. Given a cutoff-time T we consider the ith run successful if $r_i^{A,I} \leq T$ and unsuccessful otherwise. In order to establish a total order of algorithms $\mathcal{A}$ on an instance set $\mathcal{I}$ usually two aggregation steps are performed: First an instance-wise aggregation produces a scalar performance measure PM (A, I) for each pair $(A, I) \in \mathcal{A} \times \mathcal{I}$. In the second step, the instance-wise performances are aggregated by a summary statistic, e.g., the arithmetic mean or the median, to produce a single scalar performance value of A on the entire set $\mathcal{I}$ (see Fig. 1 for an illustration). In the following we focus on the instance-wise aggregation and review classical performance measures. In the remainder of this paper we use the arithmetic mean for the second aggregation step.[2]

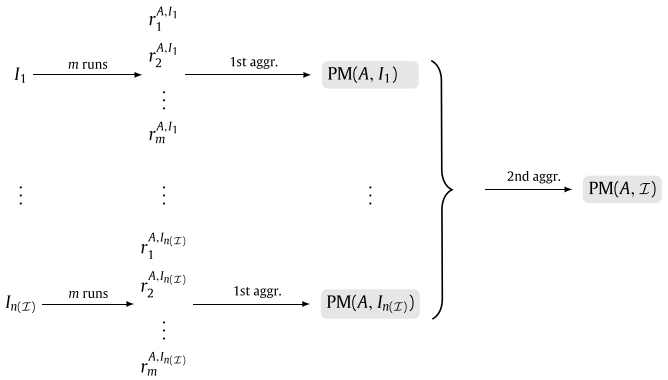

Fig. 1. Schema of two-step aggregation of running times of an algorithm A on an instance set $\mathcal{I}$: (1) aggregate on per-instance basis, (2) aggregate the aggregations.

2.1. Classical performance measurement

In the algorithm selection scenario scalar performance measures are used. That is, m runs of an algorithm on an instance are aggregated into a single quantitative value balancing goals like the fraction of successful runs and the mean (or median) running time. In the context of combinatorial optimization the so-called *Penalized Average Runtime* (PAR, [5]) is the most frequently adopted performance measure. It penalizes expired runs, i.e., runs that did not find the (known) global optimum within a maximal running time/cutoff-time T, with a score of $f \cdot T$ and afterwards averages per pair of algorithm A and instance I. Here, $f > 0$ is the penalty factor, which is commonly set to 10 (PAR10) or 2 (PAR2) [7]. For sake of completeness we state the mathematical equation:

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

For the PAR-score, the summary statistic of choice is the arithmetic mean, which is tremendously sensitive to outliers. Imagine a solver which manages to locate the global optimum in the majority of runs very fast – e.g., within seconds – but fails once — reaching the cutoff of, e.g., one hour. This single failed run shifts the PAR-score to higher values which is even leveraged further by choosing a high penalty factor. Replacing the mean by the statistically robust p-quantile leads to the *Penalized Quantile Runtime* (PQR)

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

Here, an algorithm is assumed to be unsuccessful, or failed, if less than $p \cdot 100\%$, $p \in [0, 1]$, of the m runs reached the optimum within the cutoff-time T. Otherwise, the algorithm is considered successful and the measure corresponds to the p-quantile (q_p) of the runtimes $r_i^{A,I}, i = 1, \ldots, m$, which is typically set to $p =$

[1] Aside from the considered performance measure, the quality of algorithm selectors is affected by numerous other factors such as the selected features, hyperparameter configuration, folds used for the model assessment, or even the stochasticity of the algorithm itself.

[2] Note that this decision likewise may affect the algorithm selection process. However, investigating the impact of this scalarization step is beyond the scope of this work.

0.5. In continuous optimization, another prominent performance measure is the so-called *Expected Running Time* (ERT), as proposed in [9]. As we do not consider the ERT within this study, we will not provide any details, but instead refer the interested reader to [4,9]. In the remainder of this paper we use the wide-spread less formal short-term notations PAR10 and PQR10 whenever the cutoff-time T is known from the context.

Obviously, common performance indicators are not parameter-free (consider, e.g., the penalty factor of the PAR-score). Finding a suitable configuration of these parameters is not trivial, but usually this challenge is not even considered due to widely used standard settings such as $f = 10$ in PAR10 or by choosing the arithmetic mean per default for aggregation purposes instead of quantiles. In [7] we used TSP benchmark data of inexact solvers taken from [6] to illustrate the effects and interplay of varied parameter settings. While the penalty factor only affects the influence of failed instances, it of course heavily alters the leverage of those instances compared to the impact of successful runs, especially in case aggregation over different runs is conducted by the arithmetic mean which is heavily influenced by outlier values. Thus, the choice of the penalty factor can have a substantial impact on instance-based automated algorithm selection models. In the PQR setting the level of the quantile used for the aggregation of runtimes across solver runs determines the degree of desired solver robustness. With increasing quantile level the percentage of required successful runs also increases. Naturally, very robust solvers such as the restart version of EAX, denoted EAX+Restart, which is capable of solving almost all instances of the considered instances set to optimality, are less sensitive to the choice of the quantile.

The proposed multi-objective perspective on solver performance (see Section 3) offers a promising and suitable alternative to traditional indicators incorporating multi-objective optimization concepts that allow to set corresponding parameters in a straightforward and meaningful manner.

3. Multi-objective perspective

$\text{PAR}_{f,T}$ and $\text{PQR}_{p,f,T}$ implicitly address the two goals of maximizing probability of success $p_s^{A,I} \in [0, 1]$ and minimizing the mean running time $r_s^{A,I}$ of successful runs of a solver A on instance I. Note that in the following we simply write p_s and r_s respectively to omit notational overhead. However, due to the construction of the measures along with high penalties for few failed runs certain goals are preferred, resulting in an information loss. Recently, [10] proposed to focus on the objectives separately — prior to scalarization. More precisely, the authors focused on the bi-objective view of simultaneously minimizing the probability of failure $p_f = 1 - p_s$ (maximizing p_s, respectively) and minimizing the running time of successful runs r_s yielding the *bi-objective performance vector*[3]

$$\text{MO}_T(A, I) := \left(r_s, p_f\right) \in [0, T] \times [0, 1]. \tag{1}$$

Note, that p_f (denoted as PF) may be estimated by the fraction of failed runs and r_s (denoted as ET) by the mean or median of running times of successful runs. Scalarization is performed at this stage summarizing solver behavior per instance by explicitly focusing on dedicated aspects of the solver behavior – e.g., robustness in case of p_f – and hence less information loss compared to directly calculating single-objective indicators, such as PAR, has to be accepted.

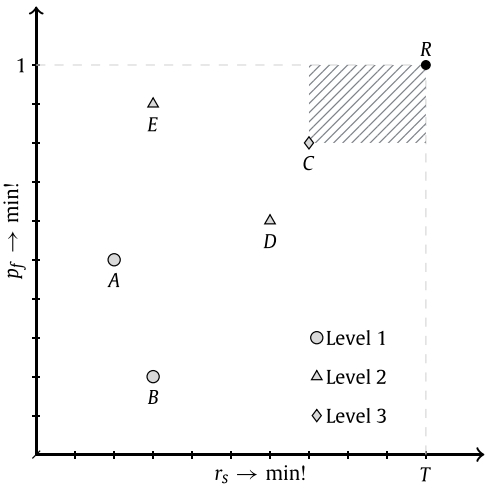

Fig. 2. Illustration of Pareto dominance, non-domination levels and the HV_T measure.

Now what is a "good" solver w.r.t. MO_T? Ideally, such a solver will show low running times (low r_s value) and high robustness (low p_f value). For algorithms $A, B \in \mathcal{A}$, *A (Pareto-)dominates B*, if A is better in at least one objective and not worse in the other. However, the case of A being more robust but slower on average and B behaving complementary, i.e., solving problems fast, but locating the optimum less often than A, is not unlikely. Thus, in terms of multi-objective optimization, A and B are incomparable. Fig. 2 illustrates this perspective in the bi-objective space. Here, A and B are incomparable, but A dominates, e.g., C. Clearly, we prefer algorithms which are located on low non-domination levels; level 1 algorithms being the most preferred ones (see the non-filled circles in Fig. 2).[4] Moreover, we may adopt performance indicators from multi-objective optimization to rank algorithms. We propose to use the *dominated hypervolume* (HV, [13,14]) of an algorithm,[5] which is defined as the space bounded by its performance (r_s, p_f) in objective space and a reference point R that is implicitly given by $(T, 1)$, i.e.,

$$\text{HV}_T(A, I) = \left(T - r_s\right) \cdot \left(1 - p_f\right),$$

to rank the algorithms — illustrated as a shaded region for algorithm C in Fig. 2. This seems to be the "natural" choice for a scalarization of the bi-objective space. By this means algorithms exceeding the cutoff-time T in all runs do not positively impact performance, whereas all successful runs (with a runtime r_s of less than T) exert a positive impact on the overall performance. Ishibuchi et al. [16] showed that the choice of the reference point has a strong impact on the result of hypervolume calculation. However, in our scenario the calculation is (a) based on a single objective vector only and (2) is robust with respect to the reference point choice as preliminary experiments revealed. Note that HV has the beneficial property that if an algorithm A dominates another algorithm B on instance I, the HV values are ranked accordingly, i.e., $\text{HV}(A, I) > \text{HV}(B, I)$. HV values per algorithm can be aggregated either per instance or instance set (see Section 2) inducing a total order of algorithms which paves the way for classical AS techniques.

We want to stress here that we do not claim the HV-measure to be the holy grail of performance measurement. Rather, we aim

[3] It should be noted that the proposed bi-objective perspective has a straightforward generalization to $p > 2$ performance measures $\text{PM}_i(A, I)$, $1 \leq i \leq p$, e.g., by additionally incorporating the variance of running times of successful runs.

[4] Level 1 algorithms are the non-dominated algorithms. Level $k > 1$ algorithms are non-dominated among all algorithms which are not on levels lower than k.

[5] Note that usually the hypervolume is calculated given a set of points, but in our scenario a single point is the basis.

to establish a bridge between classical performance measurement and a multi-objective performance viewpoint. As we shall see in Section 3.1 each performance measure is relevant, but certainly has advantages and potential drawbacks. Moreover, for a subset of performance measures functional relationships exist.

3.1. Relations to classical measures

Two of the aforementioned measures, i.e., $\text{PAR}_{f,T}$ and HV_T, are functions of the (mean) running times of successful runs and the failure or success rate, respectively. It is thus a valid assumption that there is a functional relationship between those measures. Let $n_s = \sum_{i=1}^{m} \mathbb{1}(r_i \leq T)$ be the number of successfully finished runs where $\mathbb{1}$ denotes the indicator function which evaluates to 1 if the condition inside the braces is true and 0 otherwise. Given m independent runs we can estimate the running time of successful runs via $\frac{1}{n_s} \sum_{i=1}^{m} \mathbb{1}\{r_i \leq T\} \cdot r_i$. Next, we derive some interesting relationships.

Theorem 1. *Let $T > 0$ be a time-limit and $f \geq 1$ a penalty factor. Then the following equation holds:*

$$PAR_{f,T}(A, I) = f \cdot T - p_s \cdot (f - 1) \cdot T - HV_T(A, I).$$

In order to prove Theorem 1 the following Lemma is helpful.

Lemma 2. *Let $T > 0$ be a time-limit and $f \geq 1$ a penalty factor:*

$$HV(A, I) = p_s \cdot T - \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}\{r_i \leq T\} \cdot r_i.$$

Proof. We apply basic algebraic transformations:

$$\begin{aligned}
\text{HV}_T(A, I) &= \left(T - \frac{1}{n_s} \sum_{i=1}^{m} \mathbb{1}\{r_i \leq T\} \cdot r_i \right) \cdot \underbrace{(1 - p_f)}_{=p_s} \\
&= p_s \cdot T - p_s \cdot \underbrace{\frac{m}{n_s}}_{=\frac{1}{p_s}} \cdot \frac{1}{m} \cdot \sum_{i=1}^{m} \mathbb{1}\{r_i \leq T\} \cdot r_i \\
&= p_s \cdot T - \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}\{r_i \leq T\} \cdot r_i \quad \square
\end{aligned}$$

Now, we are ready to prove Theorem 1:

Proof of Theorem 1.

$$\begin{aligned}
\text{PAR}_{f,T}(A, I) &= \frac{1}{m} \cdot \sum_{i=1}^{m} \left[\mathbb{1}\{r_i \leq T\} \cdot r_i + \mathbb{1}\{r_i > T\} \cdot f \cdot T \right] \\
&= \frac{1}{m} \cdot \sum_{i=1}^{m} \mathbb{1}\{r_i \leq T\} \cdot r_i + \underbrace{\frac{\sum_{i=1}^{m} \mathbb{1}\{r_i > T\}}{m}}_{=p_f = (1 - p_s)} \cdot f \cdot T \\
&= \frac{1}{m} \cdot \sum_{i=1}^{m} \mathbb{1}\{r_i \leq T\} \cdot r_i + (1 - p_s) \cdot f \cdot T \\
&= f \cdot T - p_s \cdot (f - 1) \cdot T \underbrace{- p_s \cdot T + \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}\{r_i \leq T\} \cdot r_i}_{= -\text{HV}_T(A,I) \quad \text{(see Lemma 2)}} \\
&= f \cdot T - p_s \cdot (f - 1) \cdot T - \text{HV}_T(A, I). \quad \square
\end{aligned}$$

Corollary 3. $PAR_{1,T}(A, I) = T - HV_T(A, I)$.

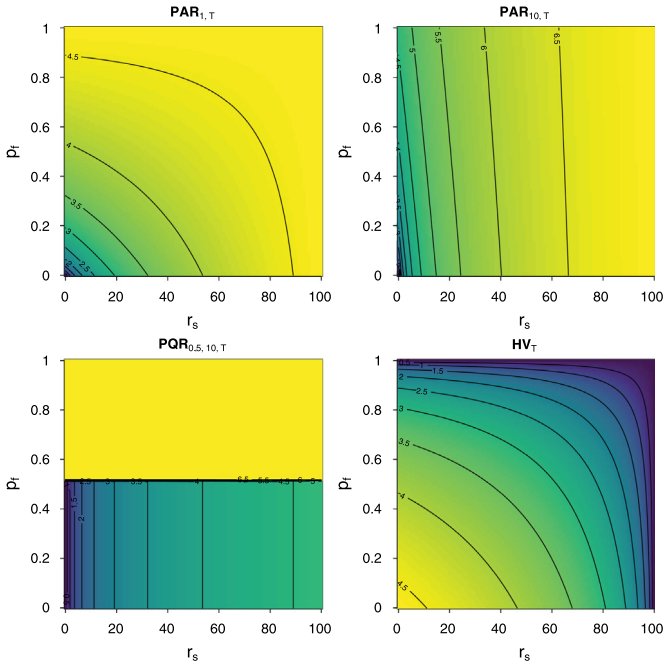

Fig. 3. Visualization of scalar performance measures as functions of the running time of successful runs r_s and the fraction of failed runs p_f (brighter colors indicate higher values). The total number of runs was set to $m = 100$ to obtain a fine-grained resolution.

Corollary 4. *Let $A, B \in \mathcal{A}$ and $I \in \mathcal{I}$. Then*

$$PAR_{1,T}(A, I) \leq PAR_{1,T}(B, I) \Leftrightarrow HV_T(A, I) \geq HV_T(B, I).$$

Proof. Follows directly from Corollary 3:

$$\begin{aligned}
& \text{PAR}_{1,T}(A, I) \leq \text{PAR}_{1,T}(B, I) \\
\Leftrightarrow \quad & T - \text{HV}_T(A, I) \leq T - \text{HV}_T(B, I) \\
\Leftrightarrow \quad & \text{HV}_T(A, I) \geq \text{HV}_T(B, I). \quad \square
\end{aligned}$$

Corollary 3 is of particular interest. In fact, the HV-measure is a linear transformation of the classical PAR-score with no penalization at all. The theoretical results are supported by the visualizations depicted in Fig. 3. Here, we see different performance measures covered in Sections 2 and 3 as functions of $p_f \in [0, 1]$ and $r_s \in [0, T]$ with cutoff-time $T = 100$. The derived relationship between $\text{PAR}_{1,T}$ and HV_T becomes obvious at first glance. In addition, the plots reveal different advantages and disadvantages of the measures (see Table 1). Let us focus on PAR-score and HV-score here. The former allows to adjust the penalization of failed runs. However, an algorithm A with many fast runs and few timeouts may be scored worse than another algorithm B which is on average slower but never fails. In contrast, the HV-score is applicable to more than two objectives and puts equal weight on both objectives. However, in certain situations the latter property may be undesired (see Table 1). Note that calculating weighted Hypervolume [17] could be a promising solution and generalization of the proposed approach.

4. Automated algorithm selection: A case study on TSP

Following the theoretical investigations of the different performance measures, we will now compare their strengths and weaknesses by means of an empirical case study based on the TSP.

Table 1
Overview of advantages and disadvantages of the covered measures. The last column illustrates examples for which undesirable effects may occur.

Measure	Advantages	Disadvantages	Negative example
$\text{PAR}_{f>1,T}$	Simple and established measure. Penalty for impact of failed runs adjustable.	May lead to biased results regarding the gap to SBS, if there are few instances where the SBS fails.	$m-1$ very fast runs and 1 failed run is scored worse than m slower runs.
$\text{PQR}_{p,f,T}$	Less outlier sensitive than $\text{PAR}_{f,T}$. Level of sensitivity and penalty for failed runs adjustable.	Information loss (runtimes of $m-\lceil mq+1\rceil$ runs not considered at all).	$m-\lceil mq+1\rceil$ failed runs, remaining runs very fast.
HV_T	Intuitive measure in the multi-objective space. No parameters necessary. Directly applicable to >2 objectives.	Both objectives equally weighted.	$\frac{m}{2}$ failed runs and $\frac{m}{2}$ successful runs with $r_s=\varepsilon$ close to 0 has equal score as no failed runs and $r_s\approx\frac{T}{2}$

4.1. Experimental setup

In order to ensure comparability of our results with the current state of the art in (inexact) TSP solving, we analyze our proposed performance metrics based on the setup presented in [6]. We thus trained competitive algorithm selectors by applying sophisticated machine learning techniques to a combination of five heuristic state-of-the-art optimization algorithms, six representative benchmark sets and three – within the literature well established and widely accepted – sets of TSP features.

TSP solvers. Our portfolio of optimization algorithms is composed of five inexact TSP solvers. EAX [11] is an evolutionary algorithm using an edge assembly crossover operator for recombining edges of parent solutions, along with sophisticated diversity preservation techniques. LKH [18] (version 2.0.7) is Helsgaun's variant of the Lin–Kernighan heuristic, a local search algorithm which heuristically performs complex edge exchanges. We further consider their corresponding restart variants, EAX/LKH+Restart [12], which trigger a restart once the corresponding original algorithm prematurely terminates. The multi-agent optimization algorithm (MAOS, [19]) completes the portfolio. All solvers are given a cutoff-time T of one hour (=3600 s) to find the optimal solution and perform $m=10$ independent runs for statistical soundness of subsequent investigations.

TSP instances. As performances strongly depend on the considered problem instances, we aimed for a balanced test suite of TSP benchmark problems and thus combined instances from real-world and artificial problem sets. The former are taken from the three well-known TSP sets VLSI, National and TSPLIB, whereas the latter – random uniform Euclidean (RUE), clustered (Netgen) and mixed (Morphed) problems – are created with problem generators (see [6] for details on the instance generation process). In the end, our benchmark comprised a total of 1844 TSP instances — 21 TSPLIB, 18 VLSI and 5 National instances, as well as 600 instances for each of the three artificial test beds. All instances contained between 500 and 2000 cities each.

Feature sets. During the last decade, several research groups developed features for characterizing TSP problems with [20–22] and [23] providing the most popular and promising concepts. Here, we focus on the latter three feature sets as they can be understood as extended versions of the characteristics from [20]. The 68 *TSPmeta features* [21] compute statistics based on the distribution of the edge costs, the convex hull of the problem's cities, the distances among the nearest neighbors, as well characteristics of a minimum spanning tree of the problem. The 50 *UBC features* [22] are based on minimum spanning trees again, the problem's distance matrix, statistics of multiple local search runs, and information extracted from the branch-and-cut tree produced by a two-second run of Concorde [24] — the best known exact TSP solver. The third feature set, i.e., the 287 *Pihera features* [23], builds upon the TSPmeta and UBC feature sets and extends them by further convex hull, local search and nearest-neighbor graph based features. Note that all feature sets are partially redundant as they share multiple features.

Machine learning techniques for algorithm selection. For finding competitive algorithm selectors, which predict the best optimizer w. r. t. our proposed performance measures, multiple classification models have been trained using the R-package `mlr` [25]. We used kernel-based support vector machines (ksvm), recursive partitioning and regression trees (rpart), gradient boosting (xgboost) and random forests. For reasons of simplicity, (almost) all machine learning algorithms have been executed in their default settings. The only exception to this is the SVM's inverse kernel width parameter `sigma`, which has been configured *a priori* due to its strong leverage on the SVM's performance.

In order to ensure reliable results, the performance of all models was assessed with a 10-fold cross-validation. Given the rather high noise and/or redundancy among the features, we additionally considered feature selection strategies during the training phase. More precisely, we tried a greedy sequential floating forward–backward selection (sffs), as well as a stochastic (10 + 5) genetic algorithm feature selection strategy with a maximum of 100 iterations. Further details on both approaches can be found in [6].

Both feature selection strategies attempt to find feature subsets that result in optimal performance among the considered feature sets w.r.t. HV and PAR10, respectively. For the computation of the HV, we straightforwardly set the reference point to (3600, 1.0) as this corresponds to the natural upper bounds of the two underlying objectives (see Section 3): a maximal runtime of one hour (= 3600 s) and a fraction of failed runs of at most 1 (100% respectively).

Regardless of their promising results in previous studies [4,6], both selection strategies rely on imperfect principles — greedy and stochastic, respectively. In consequence, they cannot guarantee to find the best possible feature combination. Still, using either one of them usually strongly improves a selector's performance – in comparison to training it on either none or all features.

4.2. Evaluation and discussion of the experimental results

In a first step, the performances of the five considered TSP solvers (EAX, EAX+Restart, LKH, LKH+Restart and MAOS) are visually compared from a multi-objective perspective by looking at the mean running time of successful runs r_s and failure probability p_f simultaneously (see Fig. 4). Noticeably, all solvers without an additional restart mechanism (EAX, LKH and MAOS) frequently failed to solve an instance, whereas EAX+Restart and LKH+Restart almost always succeeded in finding an optimal tour within the given time budget.

Following the setup described in Section 4.1, numerous algorithm selectors have been trained and afterwards the best classifier based on HV and PAR10, respectively, was chosen. The selector trained on PAR10 (denoted AS-PAR10) is a support vector machine using a total of 11 nearest-neighbor-graph features,

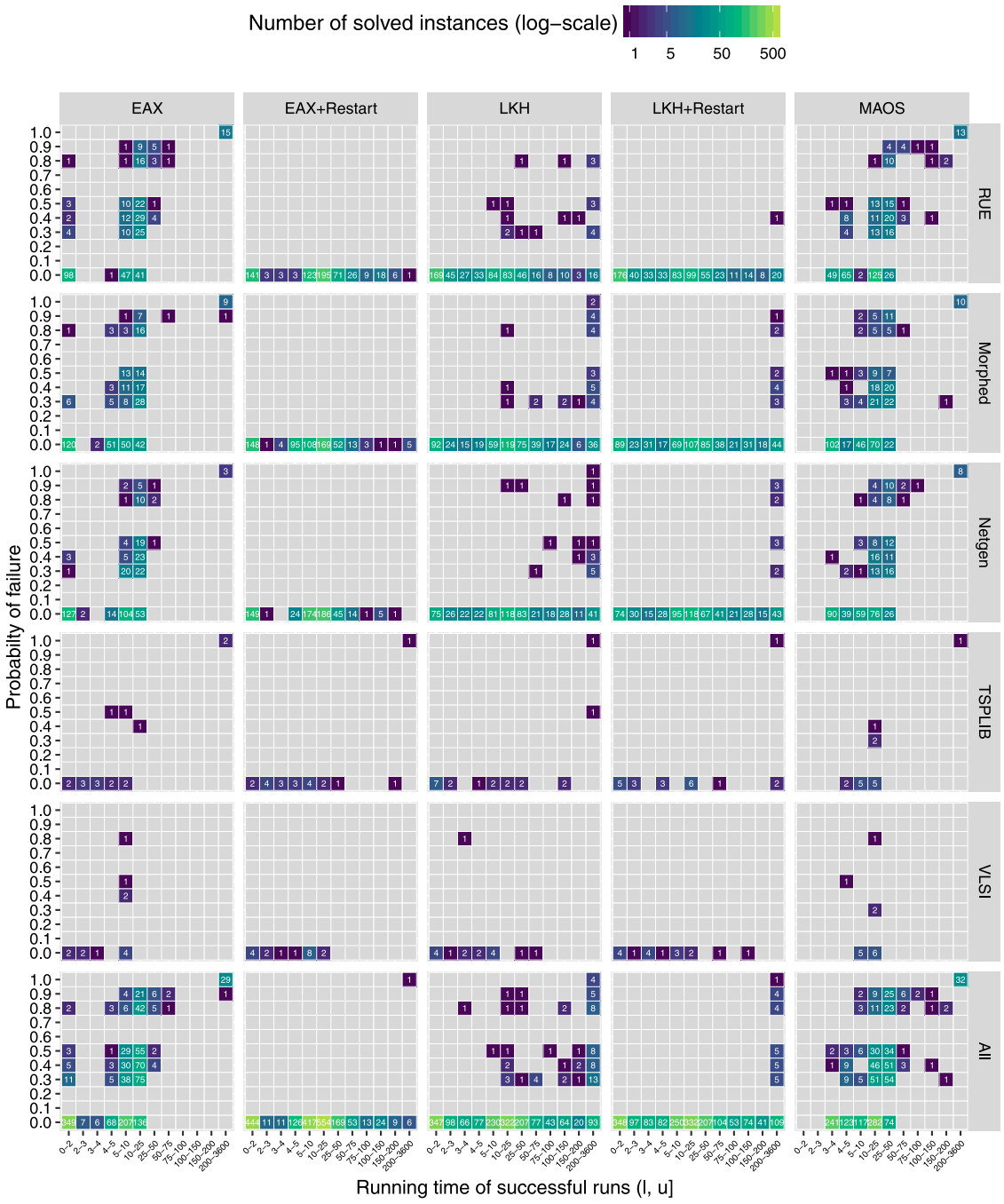

Fig. 4. Heatmaps illustrating the performances of the heuristic TSP solvers (columns) per instance set (rows). Here, the continuous scale of successful running times (x-axis) has been manually discretized into meaningful groups. Moreover, as a result of the ten runs that each TSP solver performs per instance, the probability of failure (y-axis) automatically follows a discrete scale $\{0.0, 0.1, \ldots, 1.0\}$.

which were originally proposed by [23] and have now been selected by the greedy feature selection approach. While the second selector(AS-HV) is also based on a support vector machine, it relies on a completely different set of features: four features from [22], describing the cost matrix and minimum spanning tree of a TSP instance, which were chosen by the stochastic feature selection strategy.

Then, the two selectors were compared with the five heuristic TSP solvers, as well as with the PQR10-based selector proposed in [6] (AS-ECJ). As depicted in the scatterplots in the bi-objective space in Fig. 5 and quantitatively supported by Table 2, EAX+Restart clearly outperforms its four competitors and consequently is the Single Best Solver (SBS), i.e., the solver that performs best on average, from the portfolio — independent of the considered performance measure or TSP instance set.[6] In fact, for many TSP test suites it even resembles performances close to or even better than the ones of the three selectors and thus defines a challenging baseline for the latter. Moreover, EAX+Restart even achieved the best performance among all algorithms – including the selectors – and across all considered performance measures when applied to the instances from the clustered TSP sets (Netgen and Morphed).

[6] We omitted results for the National data set due to its small size.

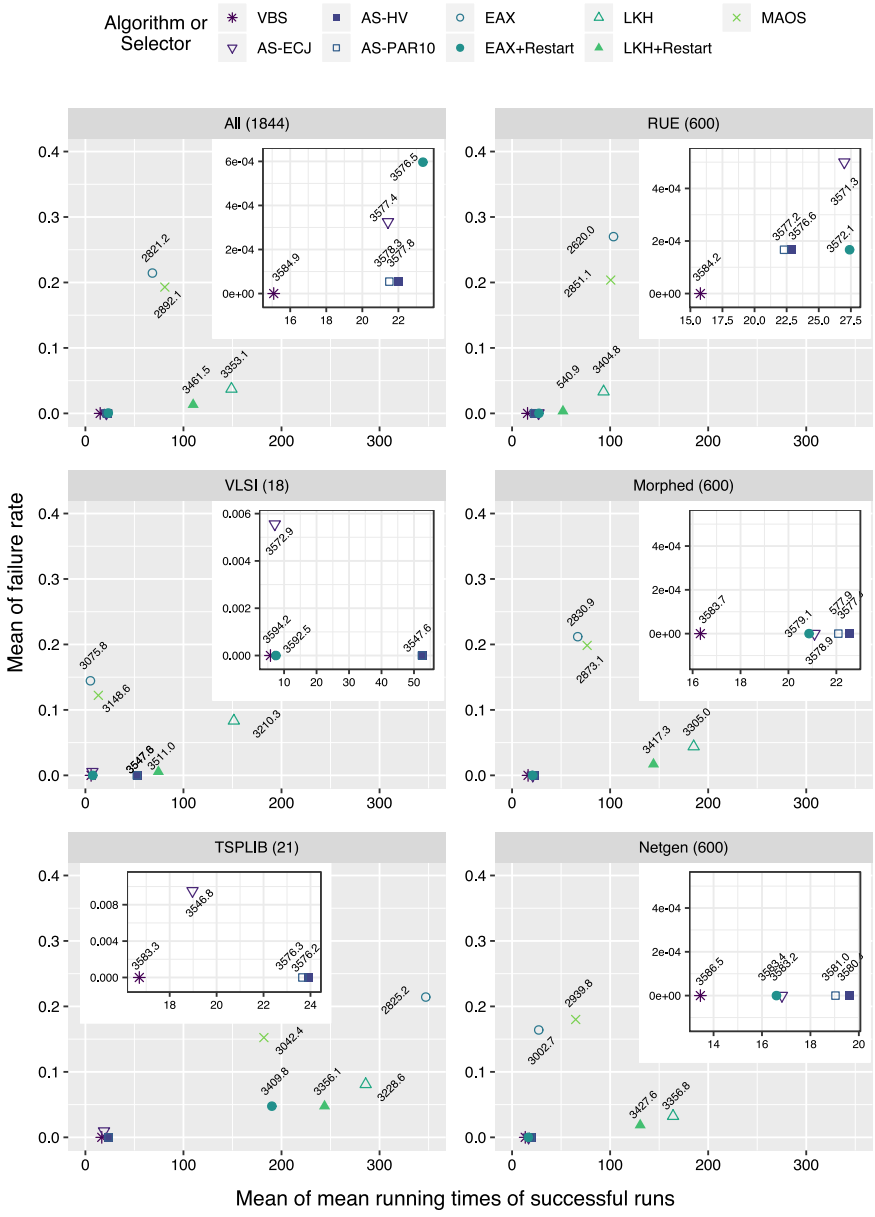

Fig. 5. Scatterplots of running times of successful runs (in s) vs. failure rates of all runs aggregated component-wise for each instance set. The white inset-plots show a zoom-in of the region close to (0, 0) to reveal minor details visually. The numbers next to the points display the corresponding HV values, which are based on the reference point (3600, 1.0). According to this indicator (and with the exception of the TSPLIB instances), EAX+Restart shows similar performance to all three selectors while outperforming the remaining four considered TSP solvers.

A commonly used measure for assessing the quality of an algorithm selector – independent of the particular task – is the so-called gap closure, which gives the percentage of the gap between the oracle-like *Virtual Best Solver*[7] (VBS) and the aforementioned SBS that is covered by the selector at hand [5,26]. While all three considered selectors are able to reduce their respective gaps, the magnitude of these ratios strongly varies depending on the considered objective: 15% (for AS-HV based on HV), 70% (AS-PAR10 based on PAR10) and 76% (AS-ECJ based on PQR10). Although the difference in gap-closure between HV and PAR10 looks rather large (15% vs. 70%), Fig. 5 indicates very similar performances between the two selectors. This finding also supports our thesis according to which the penalty factor has a strong impact on common performance measures such as PAR10 and PQR10 (see previous section). Furthermore, the similar performances of AS-HV and AS-PAR10 are also in line with our theoretical findings (outlined in Section 3) as HV simply is a linear transformation of the PAR1 score. Considering that PAR1 and all other PAR_f scores are identical to the mean running time in case of exclusively successful runs, HV also possesses a direct linear relation to PAR_f (incl. PAR10) in that particular scenario.

In addition, the plots of Fig. 5 revealed two further noticeable results. First, the mean failure rates of AS-HV and AS-PAR10 are always less or equal to the mean failure rate of AS-ECJ. At the same time, the average running time of the successful runs was usually slightly shorter for the latter selector. Hence, we can conclude that there also exists a trade-off between solution quality and speed across the selectors. We further noticed that throughout the set of all 1844 problem instances, AS-PAR10 and AS-HV always achieved identical failure rates, but the average running times (across the successful runs) of AS-PAR10 were shorter than

[7] The VBS uses oracle-like knowledge to predict the best solver per instance and thus tries to avoid misclassifications.

Table 2
Summary statistics of multiple performance indicators grouped by TSP set and solver (or selector, respectively). The table lists the hypervolume (HV), PAR10- and PQR10-scores, as well as the probability of failure p_f and estimated mean running time of successful runs r_s. The best performances per TSP set and performance measure are highlighted.

TSP set	Algorithm	HV	PAR10	PQR10	p_f	r_s
All (1844)	AS-ECJ	3577.44	33.10	**16.75**	0.0003	**21.42**
	AS-HV	3577.80	23.95	17.42	**0.0001**	22.03
	AS-PAR10	**3578.34**	**23.42**	16.89	**0.0001**	21.50
	EAX	2821.22	7724.40	6938.57	0.2144	68.32
	EAX+Restart	3576.48	42.85	36.32	0.0006	23.36
	LKH	3353.13	1457.48	1306.05	0.0374	148.96
	LKH+Restart	3461.50	568.98	566.16	0.0133	109.90
	MAOS	2892.06	6959.52	5793.11	0.1930	80.99
RUE (600)	AS-ECJ	3571.31	44.89	20.65	0.0005	26.99
	AS-HV	3576.63	28.77	18.21	**0.0002**	22.87
	AS-PAR10	**3577.20**	**28.20**	**17.58**	**0.0002**	**22.30**
	EAX	2620.05	9727.95	9430.92	0.2700	103.45
	EAX+Restart	3572.10	33.30	21.21	**0.0002**	27.40
	LKH	3404.78	1269.82	1135.67	0.0332	93.33
	LKH+Restart	3540.88	167.12	159.79	0.0033	51.91
	MAOS	2851.12	7347.68	6377.80	0.2037	100.61
Morphed (600)	AS-ECJ	3578.91	21.09	16.93	**0.0000**	21.09
	AS-HV	3577.45	22.55	17.70	**0.0000**	22.55
	AS-PAR10	3577.92	22.08	17.25	**0.0000**	22.08
	EAX	2830.93	7632.47	6846.10	0.2118	66.96
	EAX+Restart	**3579.14**	**20.86**	**16.72**	**0.0000**	**20.86**
	LKH	3304.98	1720.62	1444.99	0.0440	185.20
	LKH+Restart	3417.27	733.53	654.08	0.0170	144.23
	MAOS	2873.10	7158.30	5593.25	0.1985	76.57
Netgen (600)	AS-ECJ	3583.18	16.82	13.22	**0.0000**	16.82
	AS-HV	3580.38	19.62	15.47	**0.0000**	19.62
	AS-PAR10	3580.96	19.04	14.93	**0.0000**	19.04
	EAX	3002.66	5910.94	4627.24	0.1640	27.21
	EAX+Restart	**3583.40**	**16.60**	**12.99**	**0.0000**	**16.60**
	LKH	3356.76	1290.84	1190.55	0.0323	164.18
	LKH+Restart	3427.58	777.22	862.97	0.0187	130.67
	MAOS	2939.77	6492.23	5532.35	0.1800	64.88
TSPLIB (21)	AS-ECJ	3546.78	361.79	**13.68**	0.0095	**18.97**
	AS-HV	3576.08	23.92	20.87	**0.0000**	23.92
	AS-PAR10	**3576.35**	**23.65**	20.60	**0.0000**	23.65
	EAX	2825.16	7717.70	8574.76	0.2143	347.13
	EAX+Restart	3409.84	1733.01	1727.97	0.0476	190.16
	LKH	3228.56	2994.30	3491.68	0.0810	285.83
	LKH+Restart	3356.10	1786.75	1759.76	0.0476	243.90
	MAOS	3042.37	5494.77	3438.39	0.1524	182.03
VLSI (18)	AS-ECJ	3572.90	207.10	**5.36**	0.0056	7.14
	AS-HV	3547.32	52.68	42.38	**0.0000**	52.68
	AS-PAR10	3547.56	52.44	42.14	**0.0000**	52.44
	EAX	3075.84	5204.16	4004.18	0.1444	**5.11**
	EAX+Restart	**3592.50**	**7.50**	6.35	**0.0000**	7.50
	LKH	3210.30	3089.70	4008.29	0.0833	151.40
	LKH+Restart	3511.01	268.99	47.23	0.0056	74.41
	MAOS	3148.64	4411.36	4011.35	0.1222	13.21

the corresponding ones of AS-HV. However, the respective time differences are more or less negligible and could instead simply result from the imperfect feature selection procedures.

Despite the imperfect feature selection procedures, each of the three discussed selectors is able to leverage the performance of the SBS. This claim is also supported by a series of pairwise Wilcoxon tests based on bootstrap samples of the selector and solver performances as, e.g., previously proposed in [6]. That is, we created 1000 bootstrap samples of the 1844 instances and computed for each of the samples the aggregated performance (based on the HV measure) for each of the three selectors (AS-PAR10, AS-JV and AS-ECJ), as well as the SBS (= EAX+Restart). For each pair of the four considered algorithms (including the selectors), we then conducted paired, one-sided Wilcoxon tests based on a significance level of $\alpha = 5\%$. According to our test results, EAX+Restart is inferior to AS-ECJ, which in turn was outperformed by AS-HV. Across all tests, AS-PAR10 showed superior performance to its three contenders.

At last, we want to emphasize that our findings are completely based on this particular TSP case study. Consequently, they could look different when given a different task/scenario.

5. Conclusions

In this paper we propose a multi-objective view onto performance measurement assessing the trade-off between failure rate and average running time of successful runs for single-objective stochastic solvers on optimization problem instances. The algorithm selection problem is treated as a multi-objective decision problem allowing for deriving interesting insights into solver behavior, and the dominated hypervolume (HV) of solver results in the bi-objective space functions as a scalar and – in our

scenario – parameter-free indicator of overall performance measurement. Theoretical insights are derived in terms of analytical relationships to the PAR-indicator, specifically for PAR1.

A recent benchmark and automated algorithm selection study for state-of-the-art inexact TSP solvers is used for illustrating the potential of the introduced concept and for comparison to common approaches such as PAR10 or PQR10. By this means EAX+Restart is identified as almost exclusively (Pareto) dominating other solvers showing high degree of robustness with respect to few failed runs and low running times. Selection models specifically constructed w.r.t. the HV measure show qualitatively comparable results to classical PAR10 and dominating behavior in terms of failure rate regarding PQR. Moreover, the HV measure directly operates in the bi-objective space and the selector only requires a small amount of TSP features. Therefore, the HV measure proves to suitably complement the existing state-of-the-art performance indicators.

The proposed concept generalizes to higher dimensions, which – in combination with other performance criteria – offers various perspectives for future studies. Also, we will further investigate our proposed approach based on other scenarios from the ASlib [5] (http://www.coseal.net/aslib/), which has been out of scope for the current work. Moreover, we will tackle the domain of single-objective continuous black-box optimization as well together with theoretical and systematic empirical comparisons to ERT performance measurement.

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.asoc.2019.105901.

Acknowledgments

The authors acknowledge support from the *European Research Center for Information Systems (ERCIS)* and the DAAD PPP, Germany project No. 57314626.

References

[1] J.R. Rice, The algorithm selection problem, Adv. Comput. 15 (1976) 65–118.
[2] L. Kotthoff, Algorithm selection for combinatorial search problems: A survey, AI Mag. 35 (3) (2014) 48–60.
[3] P. Kerschke, H.H. Hoos, F. Neumann, H. Trautmann, Automated algorithm selection: Survey and perspectives, Evol. Comput. (ECJ) 27 (1) (2019) 3–45.
[4] P. Kerschke, H. Trautmann, Automated algorithm selection on continuous black-box problems by combining exploratory landscape analysis and machine learning, Evol. Comput. (ECJ) 27 (1) (2019) 99–127.
[5] B. Bischl, P. Kerschke, L. Kotthoff, T.M. Lindauer, Y. Malitsky, A. Fréchette, H.H. Hoos, F. Hutter, K. Leyton-Brown, K. Tierney, J. Vanschoren, Aslib: A benchmark library for algorithm selection, Artif. Intell. J. (AIJ) 237 (2016) 41–58.
[6] P. Kerschke, L. Kotthoff, J. Bossek, H.H. Hoos, H. Trautmann, Leveraging TSP solver complementarity through machine learning, Evol. Comput. (ECJ) 26 (4) (2018) 597–620.
[7] P. Kerschke, J. Bossek, H. Trautmann, Parameterization of state-of-the-art performance indicators: A robustness study based on inexact TSP solvers, in: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '18) Companion, Kyoto, Japan, 2018, pp. 1737-1744.
[8] N. Hansen, S. Finck, R. Ros, A. Auger, Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions, Tech. Rep. RR-6829, INRIA, 2009.
[9] N. Hansen, A. Auger, S. Finck, R. Ros, Real-Parameter Black-Box Optimization Benchmarking 2009: Experimental Setup, Tech. Rep. RR-6828, INRIA, 2009.
[10] J. Bossek, H. Trautmann, Multi-objective performance measurement: Alternatives to PAR10 and expected running time, in: R. Battiti, M. Brunato, I. Kotsireas, P. Pardalos (Eds.), Proceedings of the 12th International Conference on Learning and Intelligent Optimization (LION), in: Lecture Notes in Computer Science, vol. 11353, Springer International Publishing, Kalamata, Greece, 2019, pp. 215–219.
[11] Y. Nagata, S. Kobayashi, A powerful genetic algorithm using edge assembly crossover for the traveling salesman problem, INFORMS J. Comput. 25 (2) (2013) 346–363.
[12] J. Dubois-Lacoste, H.H. Hoos, T. 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), ACM, New York, NY, USA, 2015, pp. 377–384.
[13] C.A. Coello Coello, G.B. Lamont, D.A. van Veldhuizen, Evolutionary Algorithms for Solving Multi-Objective Problems, Springer, 2007.
[14] E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach, IEEE Trans. Evol. Comput. 3 (4) (1999) 257–271.
[15] A. Blot, H.H. Hoos, L. Jourdan, M.-É. Kessaci-Marmion, H. Trautmann, MO-ParamILS: A multi-objective automatic algorithm configuration framework, in: Proceedings of the 10th International Conference on Learning and Intelligent Optimization (LION), in: Lecture Notes in Computer Science (LNCS), vol. 10079, Springer, Ischia, Italy, 2016, pp. 32–47.
[16] H. Ishibuchi, R. Imada, Y. Setoguchi, Y. Nojima, How to specify a reference point in hypervolume calculation for fair performance comparison, Evol. Comput. (ECJ) 26 (3) (2018) 411–440.
[17] D. Brockhoff, J. Bader, L. Thiele, E. Zitzler, Directed multiobjective optimization based on the weighted hypervolume indicator, J. Multi-Criteria Decis. Anal. 20 (5–6) (2013) 291–317.
[18] K. Helsgaun, General k-opt submoves for the lin-kernighan TSP heuristic, Math. Program. Comput. 1 (2–3) (2009) 119–163.
[19] X.-F. Xie, J. Liu, Multiagent optimization system for solving the traveling salesman problem (TSP), IEEE Trans. Syst. Man Cybern. B Cybern. 39 (2) (2009) 489–502.
[20] K.A. Smith-Miles, J. van Hemert, Discovering the suitability of optimisation algorithms by learning from evolved instances, Ann. Math. Artif. Intell. 61 (2) (2011) 87–104.
[21] O. Mersmann, B. Bischl, H. Trautmann, M. Wagner, J. Bossek, F. Neumann, A novel feature-based approach to characterize algorithm performance for the traveling salesperson problem, Ann. Math. Artif. Intell. 69 (2013) 151–182.
[22] F. Hutter, L. Xu, H.H. Hoos, K. Leyton-Brown, Algorithm runtime prediction: Methods & evaluation, Artif. Intell. J. (AIJ) 206 (2014) 79–111.
[23] J. Pihera, N. Musliu, Application of machine learning to algorithm selection for TSP, in: Proceedings of the IEEE 26th International Conference on Tools with Artificial Intelligence (ICTAI), IEEE, 2014.
[24] D.L. Applegate, R.E. Bixby, V. Chvátal, W.J. Cook, The Traveling Salesman Problem: A Computational Study, Princeton University Press, Princeton, NJ, USA, 2007.
[25] B. Bischl, M. Lang, L. Kotthoff, J. Schiffner, J. Richter, E. Studerus, G. Casalicchio, Z.M. Jones, Mlr: Machine Learning in R, J. Mach. Learn. Res. (JMLR) (2016).
[26] T.M. Lindauer, H.H. Hoos, F. Hutter, T. Schaub, AutoFolio: An automatically configured algorithm selector (Extended Abstract), in: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 2017, pp. 5025–5029.