Empirical Scaling Analyzer: An Automated System for Empirical Analysis of Performance Scaling

Yasha Pushak [a], Zongxu Mu [a] and Holger H. Hoos [b,a,*]

[a] *Department of Computer Science, The University of British Columbia, BC, Canada*

[b] *LIACS, Universiteit Leiden, The Netherlands*

E-mails: ypushak@cs.ubc.ca, zongxumu@cs.ubc.ca, hh@liacs.nl

Abstract. The time complexity of algorithms, *i.e.*, the scaling of the time required for solving a problem instance as a function of instance size, is of key interest in theoretical computer science and practical applications. In this work, we present a fully automated tool – Empirical Scaling Analyzer (ESA) – for performing sophisticated and detailed empirical scaling analyses. The methodological approach underlying ESA is based on a combination of automatic function fitting and bootstrap sampling; previous versions of the methodology have been used in prior work to characterize the empirical scaling behaviour of several prominent, high-performance SAT and TSP solvers. ESA is applicable to any algorithm or system, as long as running time data can be collected on sets of problem instances of various sizes. We present results from rigorous stress-testing to critically assess ESA on scenarios with challenging characteristics. We also give an overview of empirical scaling results obtained using ESA.

Keywords: Empirical scaling analysis, Running time scaling

1. Introduction

In theoretical computer science, time complexity is one of the most prominent concepts arising in the analysis of problems and algorithms. The time complexity of an algorithm describes the time required for solving a problem instance as a function of instance size and is traditionally studied by means of theoretical analysis. For instance, the Boolean satisfiability problem (SAT) and the travelling salesman problem (TSP) are two prominent $\mathcal{NP}$-hard problems, for which the best algorithms currently known have exponential time complexity in the worst case. However, worst-case behaviour may be encountered rarely or never at all in practical situations. Therefore, empirical analysis of time complexity has seen increasing interest, because it permits predicting the running times of high-performance algorithms in practice and may also provide useful insights into their behaviour [1–3].

Very few methods exist for performing empirical running time scaling analysis that handle noise and the stochastic behaviour of algorithms in a principled, statistical way [4]. Common practice among empirically oriented algorithm researchers is to perform relatively small numbers of algorithm runs while varying problem instance size, with some of the more advanced methods taking means over independent runs of the algorithm on the same input instances to reduce the variance in observations [5]. In some cases, the mean over tens or hundreds of runs on problems of the same size are plotted for varying instance sizes, and these points are compared against each other for two competing algorithms to show that one out-performs the other. In slightly more advanced work, standard least squares regression and curve-fitting procedures are used to fit and subsequently visualize trend lines [6]. An improvement to this practice was introduced by McGeoch *et al.* [7], who described and evaluated several prototype methods for fitting and bounding empirical running time data with polynomial models.

Somewhat related to our work are methods designed to perform algorithm profiling that can automatically extract notions of problem instance size and algorithm running time; however, even these rely on the simple methods we have described above [6, 8, 9]. Ultimately, the goal of performing empirical running time scaling

*Corresponding author. E-mail: hh@liacs.nl.

analysis is to obtain estimates or bounds on how well we can expect an algorithm to perform for larger problem instance sizes than those used to perform the analysis. However, neither the work by McGeoch *et al.* [7] nor simple curve-fitting procedures address the question of how much faith we should have in the accuracy of extrapolations obtained from empirical models of performance scaling.

This article summarizes and extends an ongoing line of research [4, 10–15] on the empirical analysis of performance scaling that addresses two previously ignored or poorly handled challenges: the variability of running time across inputs of the same size and the accuracy of extrapolations obtained from scaling models. This is accomplished by introducing a bootstrap sampling procedure that handles the variability in running time in a statistically meaningful way, and by assessing extrapolation accuracy using a set of challenge instances withheld during the model fitting process. We extend this methodology and introduce it in the form of a fully automated tool: the Empirical Scaling Analyzer (ESA). ESA takes an input file providing running time data for an algorithm (referred to as target algorithm hereafter), as well as other optional files to configure ESA. ESA is not limited to fitting and assessing a single scaling model, but can deal with multiple models simultaneously – in other words, once data collection is finished, a user can collate all running time data into a file, feed it into ESA and obtain results from the scaling analysis using several parametric models. The results are presented in a technical report, which contains easy-to-read tables and figures for the scaling of the target algorithm. ESA also automatically interprets the results and assesses whether a model describes the running time data well, using a decision model newly developed here.

The advanced statistical scaling analysis technique underlying ESA has previously been applied to state-of-the-art local search algorithms for Euclidean TSP instances [12], and a prototype of ESA was used to study the empirical scaling of high-performance SAT solvers [4], which were later extended to two classes of 4-SAT instances [13]. Earlier versions of ESA have also been used to perform empirical analysis of two inexact TSP solvers and to investigate the impact of automated algorithm configuration on their empirical running time scaling [13, 14]. ESA has also been used to extend the analysis of these cutting-edge inexact TSP algorithms to compare their scaling with that of a state-of-the-art exact TSP algorithm [15].

We believe that ESA will prove to be useful for other researchers who want to study the empirical time complexity of other algorithms. ESA is available as an easy-to-use on-line service[1] and can also be downloaded and installed locally as a command-line tool with additional functionality (for an overview on how to use ESA, see Section 3).

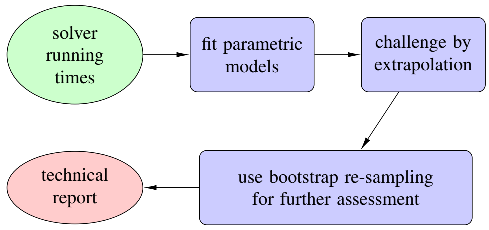


Fig. 1. Empirical scaling analysis approach underlying ESA.

Our work presented in the following makes two main contributions: we present ESA (see Section 3), a fully-automated implementation of an advanced empirical running time scaling analysis methodology (see Section 2 for a summary of the methodology and our improvements to it); and we summarize the results of performing rigorous experiments with ESA on challenging scenarios (see Sections 5 and 6). Improvements to ESA over an earlier, preliminary version include a nested bootstrap sampling procedure for randomized target algorithms and a novel method for automatically assessing the quality of fitted models (described in Section 2). To design challenging benchmarking tests for ESA, we also introduce a novel method for artificially generating realistic running time data (see Section 4). We further discuss several successful applications of the methodology underlying ESA in previous work (see Section 7), demonstrating its power and ability to provide meaningful insights in a diverse set of applications. Finally, we provide some general conclusions and briefly discuss future extensions to ESA (see Section 8).

2. Methodology

The methodology underlying ESA was first proposed by Hoos [10]; however, for completeness we resummarize this methodology here and highlight several new additions and minor improvements that we have made. At a high level, this methodology is illustrated in Figure 1. In more detail, it works as follows:

[1] www.cs.ubc.ca/labs/beta/Projects/ESA

(1) **Splitting the data.** The input set of instances and their corresponding running time and instance size data are split into two sets: a support set and a challenge set. These two sets are chosen such that all of the instances in the support set are smaller than the instances in the challenge set.

(2) **Fitting parametric models.** A pre-defined set of candidate scaling models are each fitted to the support set using the Levenberg-Marquardt algorithm, a prominent numerical optimization procedure.
If there is only a single running time available for each problem instance, then we use the same procedure as Mu & Hoos [4]. To be precise, we first calculate summary statistics, *e.g.*, median running times, for each instance size n based on k_n given instances, and then use these n data points to fit the scaling models.
For randomized algorithms, running time will vary between independent runs on the same instance. In this situation, we have multiple instances per given size and multiple running times per instance. We therefore use a nested approach that first calculates an inner summary statistic for each individual instance, *e.g.*, median running time over independent runs. Next, we calculate the n outer summary statistics for each instance size using the values of these inner statistics.

(3) **Extrapolation Test.** The resulting fitted scaling models are evaluated using the previously unseen running time data from the challenge set.

(4) **Bootstrap Analysis.** The real power of ESA comes from its statistical analysis, which is used to further quantify the confidence we can have in the scaling models and their predictions on the challenge set.

(a) **Creating bootstrap samples.** Using bootstrap sampling, we create b replicates of the support and challenge sets, respectively. If there is only a single running time available for each problem instance, we use the same procedure as Mu & Hoos [4]. In particular, for each support and challenge instance size n with k_n observed running times, we create one of the b bootstrap replicates by re-sampling k_n instances uniformly at random, with replacement.
If we have multiple running times per instance, then we use a novel, nested bootstrap sampling procedure to incorporate this information. In detail, this procedure works as follows: We first use an inner bootstrap sampling procedure to create b_{inner} bootstrap replicates of the running times for each individual instance, where for each instance I with l_I independent runs, we create one of the b_{inner} bootstrap replicates by re-sampling l_I running times uniformly at random, with replacement. Next, we use an outer bootstrap sampling procedure to create b_{outer} bootstrap replicates of the support and challenge sets. In particular, for each support and challenge instance size n with k_n instances (each of which contains b_{inner} bootstrap replicates), we create one of the b_{outer} bootstrap replicates by sampling uniformly at random one of the b_{inner} bootstrap replicates for each of k_n randomly chosen instances.

(b) **Fitting bootstrapped models.** We fit each of the candidate scaling models to each of the b_{outer} bootstrap replicates of the support set. This is done in precisely the same manner as Step 2.

(c) **Extrapolation test.** We evaluate the consistency of each candidate scaling model with the running time data from the challenge set. To do this, we calculate bootstrap percentile confidence intervals for a given confidence level α, *i.e.*, using the $(1-\alpha)/2$ and $1-(1-\alpha)/2$ quantiles of the empirical distribution of the bootstrapped statistics. We calculate these intervals for the running time statistics of each instance size in the challenge set (where these statistics are calculated in the same way as done in Step 2). Similarly, for each candidate scaling model, we calculate predictions for challenge instance size n using each of the b_{outer} fitted scaling models. Then, for each of these sets of predictions for each instance size, we again calculate bootstrap percentile confidence intervals. Finally, these intervals are used to determine whether or not a parametric model should be rejected at confidence level α (see below for more details).

ESA generates text-based interpretations for the results of the (bootstrap-based) scaling analysis, which are included in the form of a discussion in the automatically generated technical report. This is done by assessing how well a model fits the given challenge data, based on the percentage of challenge instance sizes for which the model predicts the corresponding running times reasonably accurately. If a model produces good predictions for most challenge sizes, then the model should be accepted as a good fit. Technically, we define a very good prediction, or a *strongly consistent prediction*, as one for which the bootstrap confidence interval for the model performance prediction contains the confidence interval for the observed challenge statistic, and we define a good prediction, or a *weakly consistent prediction*, as one for which the confidence intervals for the predicted and observed performances are overlapping.

Our interpretation procedure especially emphasizes the challenge points for larger instance sizes, as those provide more information regarding whether the model predictions scale with instance size. Each statement is determined using the percentage of model predictions that are strongly and weakly consistent with the observations, so the procedure is best viewed as a heuristic grounded in a statistical method. We designed the procedure and carefully hand-picked the minimum percentage of the predictions required to be strongly consistent, weakly consistent, *etc.* based on extensive experiments with several use cases (both real and artificial), to produce statements similar to those that would be made by an expert upon viewing a plot of the fitted models. However, we note that these statements (*e.g.*, "the model tends to over-estimate the challenge data") do not technically correspond to statistical tests, since a hypothetico-deductive method (such as the one used by ESA) can only be used to *reject* a hypothesis, rather than to *accept* it. Nevertheless, these statements still provide valuable, easy-to-interpret insights into the characterization of the scaling of the algorithm, and thereby enhance the overall usefulness of ESA. The detailed criteria for the various statements included in our interpretation are as follows:

- **very good fit**: the model predicts very well for most of the challenge sizes; more precisely, $\geqslant 90\%$ of the predictions for challenge sizes are strongly consistent, or $\geqslant 90\%$ of the predictions for the larger half of the challenge sizes are strongly consistent and $\geqslant 90\%$ of all of the predictions for all challenge sizes are weakly consistent;
- **fair fit**: the model predicts well for most of the challenge sizes; more precisely, $\geqslant 90\%$ of the predictions for challenge sizes or $\geqslant 90\%$ of predictions for the larger half of the challenge sizes are weakly consistent;
- **tends to over-/under-estimate**: the model predictions are over-/under-estimates or are weakly consistent with observed running time data for most of the challenge instance sizes; more precisely, $> 10\%$ of the confidence intervals for predictions on challenge instance sizes are disjoint from the confidence intervals for observed running time data and $\geqslant 90\%$ of the prediction intervals are above/below or are consistent with the observation intervals;
- **over-/under-estimate**: the model predictions are over-/under-estimates of a large percentage of the challenge sizes; more precisely, $\geqslant 70\%$ of the confidence intervals for predictions on all challenge instance sizes or $\geqslant 70\%$ of those on the larger half of the challenge sizes are above/below the observation intervals.

These criteria are combined into the fully automated interpretation procedure illustrated in Figure 2. Note that when medians (or other statistics) are not definitely known (due to instances with unknown running times), we create bootstrap confidence intervals for the medians that combine both the uncertainty from the variability in running times and the uncertainty from unknown running times, and we compare these intervals against those for the predicted running times. To be more precise, we determine confidence intervals for statistics that combine both sources of variability using an optimistic-pessimistic strategy, whereby we treat an unknown running time as zero when we calculate the lower bound of the confidence interval and as infinity in the upper bound of the confidence interval. In this way, we can guarantee that the confidence intervals must contain the desired statistic of the sample, regardless of what values the unknown running times might have taken if they had been observed. Then, we say a confidence interval I_o for observed running time on a given challenge instance size is below the corresponding confidence interval I_p for predicted running time, if the upper bound of I_o is smaller than the lower bound of I_p.

3. Running ESA

ESA implements the methodology described in Section 2 in Python 2.7, also making use of gnuplot and LaTeX to automatically generate and compile an easy-to-read technical report (in the form of a PDF file) containing detailed empirical scaling analysis results presented in tables and figures and their interpretation using our new procedure outlined in Section 2. ESA can be used in two ways: either as a simple web-based sys-

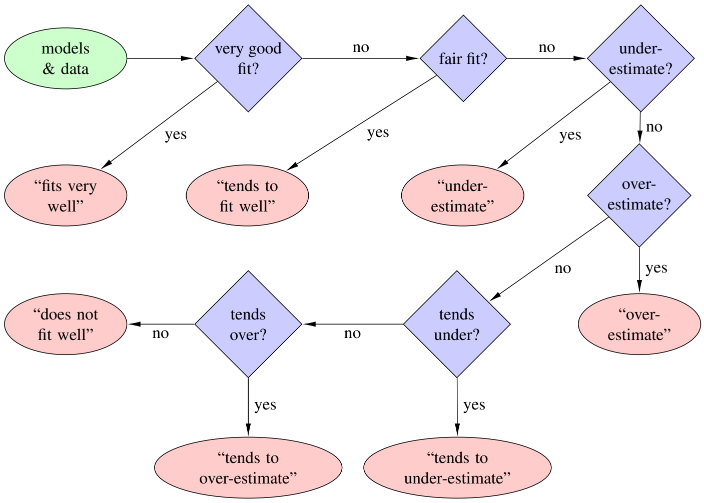


Fig. 2. Procedure used by ESA for automatic interpretation of scaling analysis results. Details on the conditions are provided in the main text.

tem or as a command-line tool[2]. While the web-based system provides easy access to most of ESA's features, the command-line tool provides some additional functionality. For many datasets, ESA can be run in 10 minutes or less; however, the running time of ESA depends primarily on how easily the scaling models can be fitted to the training set and on the number of bootstrap samples. Larger training sets, an increased number of bootstrap samples and scaling models that poorly fit the data (hence requiring more time to fit), all tend to increase the running time of ESA.

To perform scaling analysis, ESA requires input data containing the sizes of the instances studied and the running times of the target algorithm on those instances. The user may also specify the number of instances for some sizes; if there are fewer entries for a given instance size than specified explicitly, ESA will treat the missing entries as instances with unknown running times. An example for such data is found in a recent study by Dubois *et al.* [12], in the context of analyzing the scaling behaviour of two inexact TSP algorithms, where the running times of some instances were unknown due to unknown optimal tour lengths. An excerpt of an input file for ESA is shown in Figure 3. In this example, multiple running times are provided for each instance, each of which corresponds to an independent run of the target algorithm on the specified instance. The user is required to include at least one column with running times; however, any number of additional columns may be appended to the file to add additional independent runs per instance.

[2]Both are available at www.cs.ubc.ca/labs/beta/Projects/ESA.

```
#instance, size, datum (running time), datum, ...
500-1.tsp, 500, 1.60276, 1.54476, ...
500-2.tsp, 500, 1.52777, 1.42378, ...
500-3.tsp, 500, 1.41978, 1.53777, ...
...
500-1000.tsp, 500, 1.72774, 1.72074, ..
600-1.tsp, 600, 3.45747, 3.2595, ...
600-2.tsp, 600, 1.92, 2.35964, ...
...
4500-96.tsp, 4500, 1132.75, 2436.47, ...
4500-97.tsp, 4500, 227.771, 1027.32, ...
4500-99.tsp, 4500, 399.643, 188.184, ...
#instances,4000,100
#instances,4500,100
```

Fig. 3. Example input file for ESA, where "..." represents omitted lines analogous to those shown. Data shown is from EAX [16] on RUE instances.

In many applications, a user may wish to substitute an instance feature other than size that is known to affect instance difficulty. In such a scenario, it is also important for the user to ensure that all features other than the one being varied are held constant, or that the other feature values are independently and identically distributed over the instance set, so as to avoid invalidating the results of the statistical analysis due to compounding factors.

ESA also takes as input a configuration file, containing details on the target algorithm (algName), the instance distribution (instName), the number of support instance sizes (numTrainingData), *etc.* Each line of this file specifies one parameter setting in the format of "`name : value`".

There are a number of other files that a user may supply, including: a file specifying the models to be fitted, a LaTeX template specifying the content and format of the output report, and gnuplot templates specifying

Table 1

Support data summary for EAX on RUE instances

n	500	600	$\cdots$	1500
# instances	1000	1000	$\cdots$	1000
# running times	1000	1000	$\cdots$	1000
mean	1.901	2.951	$\cdots$	29.15
coefficient of variation	0.4569	1.05	$\cdots$	6.151
Q(0.1)	1.495	2.111	$\cdots$	10.84
Q(0.25)	1.601	2.253	$\cdots$	11.58
median	1.73	2.516	$\cdots$	13.21
Q(0.75)	1.917	2.774	$\cdots$	24.97
Q(0.9)	2.08	3.181	$\cdots$	34.69

the format of the plots appearing in the report. The first of these is needed, because ESA supports customized models, as long as the models are supported by python (including the math and numpy packages) and gnuplot. Each line of this file specifies one parametric scaling model, including the model name (*e.g.*, Exponential), the number of parameters (*e.g.*, 2), LaTeX, python and gnuplot expressions for the model, as well as default values for the fitting parameters. In the mathematical expressions for the models, x represents the instance size, while model parameters are written as @@a@@, @@b@@, *etc.*

ESA comes with a default LaTeX template for the report containing the results of the scaling analysis. This template can be customized easily by anyone with working knowledge of LaTeX. Dynamic elements are enclosed in "@@" in the template; *e.g.*, the target algorithm name specified in the configuration file is referenced as "@@algName@@". Users of ESA can also modify the formatting of the plots used for graphically presenting scaling analysis results, by editing the default template gnuplot script, *e.g.*, to obtain log-log or semi-log plots.

Here, we illustrate some examples of ESA output from our analysis on EAX [16], a state-of-the-art inexact TSP solver based on an evolutionary algorithm with a special edge assembly crossover operator [17]. The tables and figures include:

- Two tables showing statistics of running times for support and challenge data, respectively, to summarize the data set. An example of the support data summary is illustrated in Table 1.
- A table presenting fitted models and corresponding root mean squared error (RMSE) values, which can be used to easily see which model best fits the data according to the challenge RMSE (which is highlighted in boldface), as illustrated in Table 2.

Table 2

Individual scaling models fitted to EAX on RUE Instances

	Model		RMSE (support)	RMSE (challenge)
EAX	Exp. Model	1.0511×1.0017^{x}	1.1903	$[674.26, 864.85]$
	RootExp. Model	$\mathbf{0.15777 \times 1.1215^{\sqrt{x}}}$	**0.22553**	$[\mathbf{24.586}, \mathbf{183.39}]$
	Poly. Model	$1.409 \times 10^{-5} \times x^{1.8788}$	0.11865	$[122.17, 300.04]$

Table 3

Confidence intervals for scaling model parameters from Table 2

Solver	Model	Confidence interval of a	Confidence interval of b
EAX	Exp.	$[1.0177, 1.079]$	$[1.0017, 1.0017]$
	RootExp.	$[0.14835, 0.16531]$	$[1.1198, 1.1236]$
	Poly.	$[1.1187 \times 10^{-5}, 1.6711 \times 10^{-5}]$	$[1.8542, 1.9122]$

Table 4

Bootstrap confidence intervals for RMSE of EAX scaling models

Solver	Model	Support RMSE		Challenge RMSE	
		Median	Confidence Interval	Median	Confidence Interval
EAX	Exp.	0.45758	$[0.40345, 0.51484]$	780.07	$[417.88, 980.96]$
	RootExp.	**0.20977**	$[\mathbf{0.16623}, \mathbf{0.28264}]$	**82.313**	$[\mathbf{11.721}, \mathbf{468.78}]$
	Poly.	0.21901	$[0.15746, 0.28357]$	200.36	$[95.001, 574.23]$

- A figure showing running times, fitted models and corresponding bootstrap confidence intervals for each model, which provides a very useful and easy to understand visualization of the analysis performed, as illustrated in Figure 4.
- A figure showing the residues of the fitted models, which helps the user easily identify trends, as illustrated in Figure 5.
- A table of bootstrap confidence intervals for all model parameters, which allows a user to assess the uncertainty in the fitted models and perhaps reject (or fail to reject) hypotheses about whether or not empirical observations match theoretical expectations about an algorithm's scaling. An example is shown in Table 3.
- A table of medians and bootstrap confidence intervals for the support and challenge RMSE of each model, which provides more information about how well the models fit the data than Table 2 by leveraging the bootstrap analysis, as illustrated in Table 4.
- Two tables, for each model, of bootstrap confidence intervals for observed and predicted running times, one for support data and the other for challenge data. These tables allow the user to easily identify which model predictions are weakly or strongly consistent with the observa-

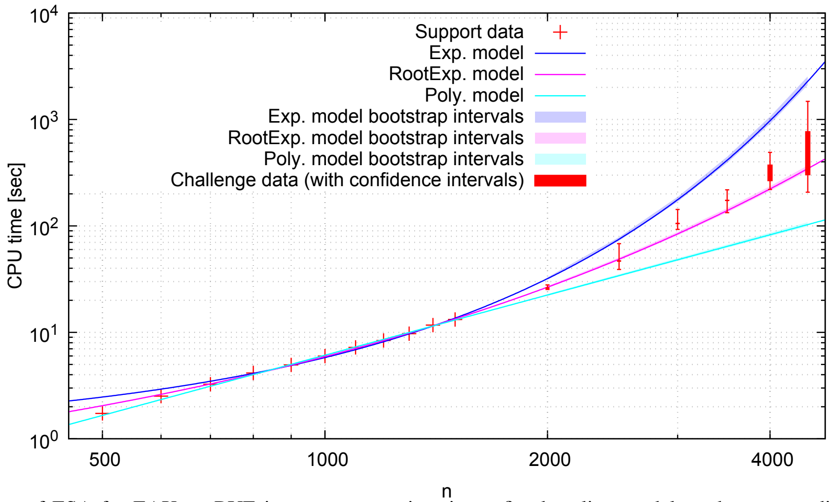


Fig. 4. Example output of ESA for EAX on RUE instances – running times, fitted scaling models and corresponding bootstrap confidence intervals.

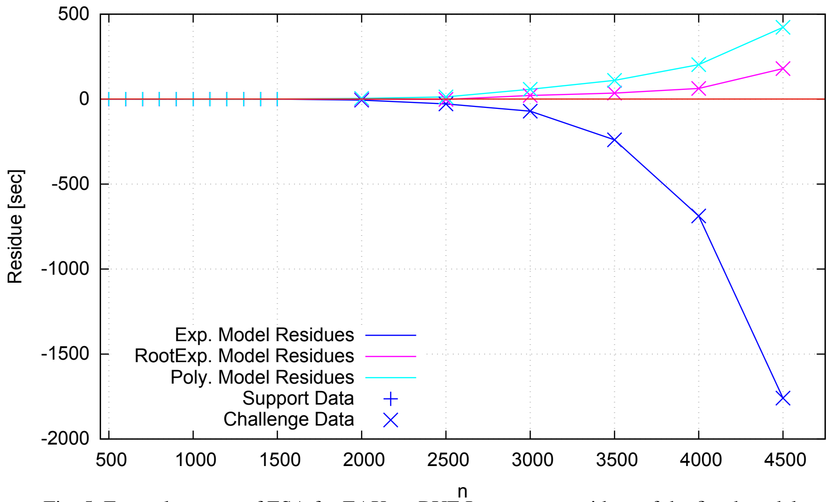


Fig. 5. Example output of ESA for EAX on RUE Instances – residues of the fitted models.

tions through boldface and asterisks. An example for challenge data is illustrated in Table 5.

A snapshot of the report generated by ESA using the default LaTeX template is shown in Figure 6, the full report can be found online [3].

4. Benchmark sets

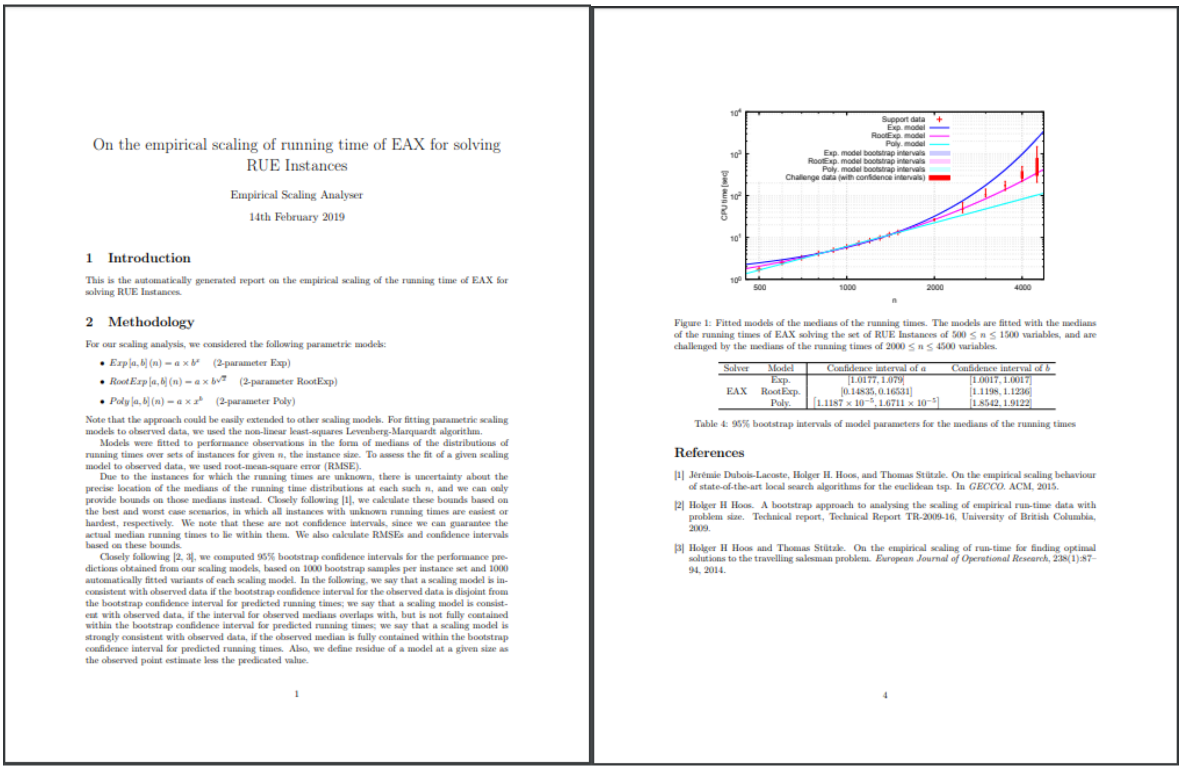
In order to assess the quality of the results obtained by ESA, it is important to study "real" application scenarios (*i.e.*, running time data sets obtained by running an algorithm on a set of instances) as well as "artificial" scenarios, where we have complete control over the properties of the running time data set and are thus able to verify that ESA produces the correct output. To this end, we have developed a novel technique for producing approximately realistic running time data sets with known scaling properties. We perform a rigorous analysis of ESA's performance on such artificial data sets and derive advice and best-practices in Section 5 and we perform additional experiments investigating ESA's performance on artificial data sets with competing, lower-order terms in Section 6. Finally, in Section 7, we present a summary of successful applications of ESA's methodology on real-world data sets.

[3] www.cs.ubc.ca/labs/beta/Projects/ESA/samples/scaling_EAX.pdf

Table 5

Confidence intervals for observed and predicted challenge data

Solver	n	Prediction confidence intervals RootExp. model	Observed median run-time Point estimates	Observed median run-time Confidence intervals
EAX	2000	**[26.06, 27.26]**	26.19	[25.3, 27.92]
	2500	**[47.37, 50.44]**	46.82	[39.03, 68.04]
	…	…	…	…
	4500	**[327.5, 369.2]**	[303.3, 766.8]	[207.5, 1482]



On the empirical scaling of running time of EAX for solving RUE Instances

Empirical Scaling Analyser

14th February 2019

1 Introduction

This is the automatically generated report on the empirical scaling of the running time of EAX for solving RUE Instances.

2 Methodology

For our scaling analysis, we considered the following parametric models:

- $Exp\,[a,b]\,(n) = a \times b^x$ (2-parameter Exp)
- $RootExp\,[a,b]\,(n) = a \times b^{\sqrt{x}}$ (2-parameter RootExp)
- $Poly\,[a,b]\,(n) = a \times x^b$ (2-parameter Poly)

Note that the approach could be easily extended to other scaling models. For fitting parametric scaling models to observed data, we used the non-linear least-squares Levenberg-Marquardt algorithm.

Models were fitted to performance observations in the form of medians of the distributions of running times over sets of instances for given n, the instance size. To assess the fit of a given scaling model to observed data, we used root-mean-square error (RMSE).

Due to the instances for which the running times are unknown, there is uncertainty about the precise location of the medians of the running time distributions at each such n, and we can only provide bounds on those medians instead. Closely following [1], we calculate these bounds based on the best and worst case scenarios, in which all instances with unknown running times are easiest or hardest, respectively. We note that these are not confidence intervals, since we can guarantee the actual median running times to lie within them. We also calculate RMSEs and confidence intervals based on these bounds.

Closely following [2, 3], we computed 95% bootstrap confidence intervals for the performance predictions obtained from our scaling models, based on 1000 bootstrap samples per instance set and 1000 automatically fitted variants of each scaling model. In the following, we say that a scaling model is inconsistent with observed data if the bootstrap confidence interval for the observed data is disjoint from the bootstrap confidence interval for predicted running times; we say that a scaling model is consistent with observed data, if the interval for observed medians overlaps with, but is not fully contained within the bootstrap confidence interval for predicted running times; we say that a scaling model is strongly consistent with observed data, if the observed median is fully contained within the bootstrap confidence interval for predicted running times. Also, we define residue of a model at a given size as the observed point estimate less the predicated value.

1

Figure 1: Fitted models of the medians of the running times. The models are fitted with the medians of the running times of EAX solving the set of RUE Instances of $500 \leq n \leq 1500$ variables, and are challenged by the medians of the running times of $2000 \leq n \leq 4500$ variables.

Solver	Model	Confidence interval of a	Confidence interval of b
EAX	Exp.	[1.0177, 1.079]	[1.0017, 1.0017]
	RootExp.	[0.14835, 0.16531]	[1.1198, 1.1236]
	Poly.	$[1.1187 \times 10^{-5}, 1.6711 \times 10^{-5}]$	[1.8542, 1.9122]

Table 4: 95% bootstrap intervals of model parameters for the medians of the running times

References

[1] Jérémie Dubois-Lacoste, Holger H. Hoos, and Thomas Stützle. On the empirical scaling behaviour of state-of-the-art local search algorithms for the euclidean tsp. In *GECCO*. ACM, 2015.

[2] Holger H Hoos. A bootstrap approach to analysing the scaling of empirical run-time data with problem size. Technical report, Technical Report TR-2009-16, University of British Columbia, 2009.

[3] Holger H Hoos and Thomas Stützle. On the empirical scaling of run-time for finding optimal solutions to the travelling salesman problem. *European Journal of Operational Research*, 238(1):87–94, 2014.

4

Fig. 6. A snapshot of the 7-page technical report generated by ESA.

Any artificially generated data set of running times should display to the greatest possible degree characteristics similar to those of realistic application scenarios. Towards this end, we simulated a randomized algorithm with three distinct sources of variability in running time: variance due to differences between problem instances (of the same size); variance in running times between multiple independent runs on the same problem instance; and changes in running times as a function of instance size. To simulate the variance from independent runs on a single instance, we draw samples from an exponential distribution, parameterized to have a median running time equal to the desired running time for the instance. In theory, many families of distributions could be used to model the variability in running times, and each would likely produce slightly different results. However, since we are particularly interested in being able to apply the lessons learned from this analysis to scenarios with $\mathcal{NP}$-hard problems, we chose an exponential distribution, which is known to closely resemble behaviour observed for a range of prominent stochastic local search algorithms (see, *e.g.*, [18]).

Similarly, to determine the median running time for a given instance, we draw a sample from an exponential distribution with a prescribed median. While we expect that the distribution in running times between instances will vary between applications, we chose an exponential distribution because we have observed high variability and heavy tails in (median) running times across instances of the same size for several scenarios we studied previously (*i.e.*, three complete and three incomplete SAT solvers for Random 3-SAT phase transition instances [4] and two state-of-the-art, inexact TSP solvers on RUE instances [14]). Finally, to determine the median running time for a given instance size, we use a given scaling model mapping instance size to median running time.

As an example, assume we want to generate running times for a randomized algorithm with quadratic scaling on an instance of size $n = 1\,000$. First, we would pick the running time scaling model, e.g., $10^{-6} \cdot n^2$, and use it to compute the median running time for instance size 1 000 – in this case $10^{-6} \cdot 1\,000^2 = 1$ (CPU second). Second, we draw a sample from an exponential distribution with median 1. In this case, assume we draw a value of 0.83291 (CPU seconds); this means that 0.83291 is the median running time for that particular instance. Finally, if we want to simulate 3 independent runs of the algorithm on this instance, we would draw 3 samples from an exponential distribution with median 0.83291.

5. Stress testing

There are many potential factors that could cause ESA to report misleading or incorrect results: for example, it could erroneously accept an incorrect scaling model because of misleading lower-order terms, or because the confidence intervals for the model predictions are so large that any model fits the data. Clearly, the latter case is more benign than the first, but it is not always clear how to resolve the problem. To better understand the robustness and limitations of ESA, we conducted a series of carefully designed *stress-testing experiments*. Specifically, by varying properties of artificially generated benchmark sets of running time (see Section 4), we studied the performance of ESA for a range of challenging situations, *i.e.*,

- decreasing the number of instances per instance size;
- decreasing the number of support instance sizes;
- reducing the number of independent runs per instance;
- increasing the extrapolation distance; and
- decreasing the number of bootstrap samples ESA uses.

We discuss the additional challenges imposed by the presence of competing, lower order terms in an algorithm's running time scaling in Section 6.

We generated two data sets, using a polynomial and an exponential scaling model. For the polynomial model, we used $2.58 \cdot 10^{-10} \cdot n^{3.37}$, which tended to fit some real running time data obtained from a TSP solver in preliminary experiments. We fitted the exponential model to data from the polynomial model, to make the two models as similar as possible. We generated running times for 21 instance sizes: 500, 600, ..., 1 900, 2 000, 2 500, ..., 4 500, and we used 16 support instance sizes, 5 challenge instance sizes, 500 instances per size and 10 independent runs per instance. Since the instance sets are sampled from a probability distribution, we created a very large data set with 10 000 instances per size and 100 independent runs per instance. This allowed us to perform 1 000 independent runs of ESA on various sub samples of the original data sets with the desired properties, *e.g.*, 100 instances per support instance size. For all of our experiments, we set ESA's parameters to their default values, using 1 000 bootstrap samples (unless otherwise indicated) and an alpha value of 95, and we studied the median running time of the per-instance medians.

In the following, we provide only a high-level summary of our findings, and distill from these results generic advice on best-practices for using ESA. For a substantially more detailed discussion and presentation of the results, please see our online, supplementary material[4].

What happens when we decrease the number of instances per instance size? We studied ESA's performance with 10, 20, 50, 100, 200, 500 and 1 000 instances per instance size. We found that ESA can identify that the correct model fits the data even with a very small number of instances per size; however, this is mostly because the size of the bootstrap confidence intervals for the fitted model predictions grows much more quickly than the size of the confidence intervals for the observed challenge statistics, *i.e.*, all of the fitted models fit the data very well for small instance sets. For example, for the polynomial model on the polynomial data set and for 10 support instances per size, the median confidence interval size for predictions on challenge instance size 4 500 was 247.2, where we measure the size of a confidence interval as the upper bound divided by the lower bound. That is to say, the upper bound of the 95% confidence interval was 247.2 times larger than the lower bound, for the median size of the confidence intervals determined from our 1 000 runs of ESA. On the other hand, the

[4] www.cs.ubc.ca/labs/beta/Projects/ESA

median confidence interval size for observed running times on instance size 4 500 was only 6.1, *i.e.*, the interval for performance predictions was 40.2 times larger than that for observations. In comparison, when using 1 000 instances per instance size, the interval for predictions was only 1.4 times larger than that for observations. We observed qualitatively similar results for the other models and on the exponential data set. Given how much more quickly the confidence intervals for predictions grow than those for observed performance as we decrease the number of instances per instance size, it is unsurprising that ESA determines that all of the fitted models fit the challenge data very well.

What happens when we decrease the number of support instance sizes? To avoid conflating changes to the number of support instance sizes with changes to other properties of the data set (*e.g.*, the extrapolation distance or the range covered by the support sizes), we fixed the location of all of the challenge instance sizes and the largest support instance size, and we forced the smallest support instance size to always be either 500 or 600. Then, to control the number of support instance sizes we varied their density. For example, when using 8 support instance sizes instead of 16, we used every second support instance size from our overall settings of $n = 500, 600, ..., 2000$.

To our surprise, we observed that ESA reported far less false positives in this case than when we reduced the number of instances per instance size, relative to the total number of support instances available. For example, the square-root exponential model was reported to tend to fit the data 6.4% of the time for the polynomial data set and 14.7% of the time for the exponential data set when given only 3 support instance sizes. In comparison, when given 16 support instance sizes, but only 100 instances per size – a roughly comparable total number of support instances – the square-root exponential model tended to fit the data to 51.4% and 51.0% of the time, for the polynomial and exponential data sets respectively. Specifically, when given 1 600 instances spread between 16 support instance sizes, ESA reported false positives 8.0 times more often for the polynomial data set and 3.5 times more often for the exponential data set, than when given 1 500 instances spread between 3 instance sizes.

However, while this may indicate a good option for saving on computational expenses, we advise extreme caution when analyzing results with ESA that use very small numbers of support instance sizes. In particular, when ESA does report a false positive, it does so with confidence intervals for model predictions that are smaller than in the case with less instances per instance size; *e.g.*, for instance size 4 500 the median interval size for the square-root exponential model performance predictions on the polynomial data set was 3.2 when using 1 500 instances spread between 3 instance sizes, compared to 7.4 for 1 600 instances spread between 16 support instance sizes. As a result, a user may be lead to incorrectly assume that ESA's best-fit model accurately captures the true scaling. In real scenarios, we expect there to be an added challenge for ESA: coping with the effects of lower order scaling terms, which would likely significantly increase the probability that ESA will incorrectly classify the scaling when only a few support instances sizes are used. Furthermore, as we discuss in Section 6, the best safeguard of which we are aware against making incorrect assumptions due to lower order terms consists of looking at the degree to which the model fits both the support and challenge data. When a very small number of support instance sizes are available, this type of safeguard becomes impossible, because there is not enough data to observe systematic deviations in the fitted models compared to the running times observed on the smallest support instance sizes.

What happens when we reduce the number of independent runs per instance? For randomized target algorithms, or in situations where significant noise in the execution environment is present, it is common to perform multiple independent runs of the algorithm on each instance and then take the median running time for each instance in order to obtain stable performance estimates [5]. We studied a set of 6 values for the number of independent runs per instance: 1, 2, 5, 10, 20 and 50.

We observed a similar trend as when we reduced the number of instances per instance size; however, the decrease in confidence interval size and the increase in false positives are more benign in this case. In particular, consider the decrease from 10 runs per instance to 1 run per instance, compared to the difference from using 500 instances per instance size to 50 instances per size. In both cases, we are decreasing the total number of algorithm runs by a factor of 10. However, the response in the size of the bootstrap intervals, and hence in ESA's interpretation of the model fit, is drastically different in the two cases. In particular, for 1 run per instance, the median size of the bootstrap interval for the running time predicted by the polynomial model on the polynomial data set for instance size 4 500 is 2.6, compared to 9.9 for 50 instances per instance size. The percentage of false positives for the square-root exponential model on the polynomial data set is only 4.1% with 1 run per instance compared to 74.1% with 50 instances per instance size – *i.e.*, there are 18.1 times as many false positives for the square-root exponential model on the polynomial data set when reducing the number of instances per size by a factor of 10 than when reducing the number of runs per instance by a factor of 10.

Based on this striking difference, it is clear that if the time required to collect all of the running time data is constrained, then the best option is to use very few independent runs per instance and more instances per instance size, assuming these instances are available. This result aligns with our expectations, since using multiple instances captures variability due to both randomness in the algorithm (and/or noise in the execution environment) as well as differences between instances, whereas performing multiple runs per instance captures a strict subset of the total variability. This insight also underlies a theoretical result by Birattari [19], which shows that using only a single run per instance with many instances is the optimal choice when estimating the mean performance of a randomised algorithm over a distribution of multiple problem instances. Nevertheless, in the event that additional instances are unavailable for study, the quality of the resulting analysis can still be improved by using the nested bootstrap procedure presented in this work to properly quantify the variance due to additional independent runs per instance.

What happens when we increase the extrapolation distance? To isolate the effect of varying the extrapolation distance, we used only a single challenge instance size for these experiments. We also used only 11 support instance sizes, 500, 600, ..., 2 000, instead of the 16 used in the previous experiments, in order to work with 5 different challenge instance sizes: 2 500, 3 000, ..., 4 500.

Overall, these results line up well with our intuition: the farther the extrapolation, the higher the probability that ESA will correctly identify the true scaling and reject incorrect scaling hypotheses. Consider, for example, the exponential data set. The exponential model was reported to tend to fit or to fit the data very well in at least 99.8% of runs for each location of the challenge instance size. However, as the challenge instance size was moved from 2 500 to 4 500, the percentage of times it was reported to fit the data very well increased from 16.7% to 88.1%. At the same time, the square-root exponential model was reported to tend to fit the data 94.8% of the time with challenge size 2 500, but only 0.1% of the time with challenge size 4 500. We obtained analogous results for the polynomial data set. While this may seem somewhat unsurprising, it does indicate that the separation of the models fitted by ESA grows more quickly than the size of the intervals for the model predictions, otherwise ESA's ability to distinguish between the models would not increase. As a result, increasing the extrapolation distance is one of the best ways to obtain more reliable and statistically significant results with ESA. Of course, this comes at the cost of the target algorithm runs themselves requiring more running time.

What happens when we decrease the number of bootstrap samples used by ESA? We tried seven values for the number of bootstrap samples used by ESA with approximately logarithmic spacing: 20, 50, 100, 200, 500, 1 000 and 2 000. We found that modifying this parameter had a mostly negligible effect on ESA's performance, which is somewhat surprising, especially in the context of very small numbers of bootstrap samples. Overall, the largest effect that we observed was a change in ESA's running time, which is roughly linear with the number of bootstrap samples. We believe that we would have observed more significant effects on ESA's performance had we also used less support data, so we still recommend to use at least 1 000 bootstrap samples, since the cost is typically small relative to performing additional algorithm runs. On the other hand, we observed no significant benefit to increasing the number of bootstrap samples beyond 1 000.

6. Lower-order terms

One possible source of difficulty for ESA occurs when a given target algorithm shows scaling of running time characterised by a function that includes lower-order terms in addition to the asymptotically dominant term. For example, an algorithm may show exponential asymptotic scaling of running time with instance size; however, for small instance sizes, the scaling may appear to be polynomial, because the running times are dominated by polynomial costs incurred by initializing data structures.

To investigate these effects, we used running time data sets generated with two polynomial terms with degrees 2 and 5. For the degree-2 polynomial term, we used the coefficient $9.6 \cdot 10^{-7}$, and for the degree-5 polynomial term, we considered three coefficients: $4.8 \cdot 10^{-15}$, $4.8 \cdot 10^{-16}$ and $4.8 \cdot 10^{-17}$. These values were chosen so that the transition occurs near the low, middle and high end of our support instance sizes, respectively. Our data sets contained 11 independent runs per instance with 1 000 instances for each of the 21 instance sizes 500, 600, ..., 1 900, 2 000, 2 500, ..., 4 500.

In addition, we ran ESA three times using three different values for the number of support instance sizes: 8, 12 and 16; in each case, all remaining instance sizes were used as challenge data. These experiments produced a total of nine different ESA reports, which we examined in detail. We also provided ESA with a four-parameter, two-term polynomial model of the form $a \cdot n^b + c \cdot n^d$, to determine if the scaling behaviour could be correctly identified, and accurate scaling models could be produced.

What happens when the transition is early? When the transition between dominant terms occurs within the lower range of small support instance sizes, the fit of the single-term model is able to capture the asymptotic scaling relatively accurately, *e.g.*, see Figure 7, where a polynomial model of degree 4.97 fits the challenge data very well. In comparison, the two-term polynomial model provides a better fit for the small instance sizes, but yields larger bootstrap intervals for the model predictions. For this model, ESA fit a degree of 2.00 for one of the polynomial terms and a degree of 5.14 for the other, and reported that this model also fit the data very well. Considering the large number of parameters in the more flexible two-term polynomial model, this is not surprising.

These results are positive, but we note that ESA experiences some difficulties fitting the 4 parameter model. In particular, higher-quality initial parameter values are needed for the two term model than for single-term models. Furthermore, the confidence intervals for the degrees of each term in the four-parameter model are relatively large and overlapping, at $b \in [1.41, 4.87]$ and $d \in [4.77, 6.26]$. We believe that this is caused by outliers in the data for large support instances sizes that lead to some model over-fitting within ESA.

What happens when the transition occurs near the middle of the support range? As the transition between the two terms moves closer to the large end of the support instance sizes, the quality of the ESA report starts to degrade. Overall, ESA is still able to do quite well, as long as the location of the transition is completely covered by the support instance sizes, as seen in Figure 8, where the bootstrap intervals for the predictions obtained from both types of polynomial models capture the observed challenge data. The single-term model is reported to "fit the data very well", although it does not quite capture the true degree of the asymptotic scaling with a reported confidence interval of $[4.00, 4.74]$. On the other hand, the two-term model, which is only reported to "tend to fit the data", does capture the true asymptotic scaling, with the confidence intervals of $b \in [1.50, 2.30]$ and $d \in [4.78, 7.06]$.

In this case, we also see that ESA had less trouble distinguishing between the two polynomial terms when fitting the two-term polynomial model, as evidenced by the disjoint bootstrap intervals for b and d. However, we also see that the size of the bootstrap intervals for the two-term model predictions has increased significantly (see Figure 8). We believe this is caused by the fact that there is only a small number of instance sizes past the midpoint of the transition, and hence less data to help ESA recover from the effects of outliers. When we decrease the number of support instance sizes (data not shown) we find that the confidence intervals for the model predictions are similar in size for the two-term polynomial model; however, the single-term model is unable to accurately capture the asymptotic scaling.

What happens when the transition is late? The worst-case scenario for ESA occurs when the transition in dominance between two competing terms occurs for instance sizes close to or beyond the largest support instance size, so that the true asymptotic scaling is heavily obscured on the given running time data. This can be seen in Figure 9, where the square-root exponential model appears to fit the data very well. However, ESA also identifies that the two-term polynomial model fits the data very well, and thereby does not dismiss the correct scaling model. In practice, the safest course of action in such cases is to collect more running time data – in particular, for larger challenge instance sizes – and to run ESA again. In cases where this is impossible, a pragmatic user would be inclined to choose the square-root exponential model as the one that is the best fit, while keeping in mind that it may be an over-estimate for the true running time scaling. In our example, we can see limited support that the square-root exponential model is an over-estimate by examining the smallest support instance sizes, for which the curvature of the square-root exponential model is just beginning to pull the model above the observed running times. Similar situations may occur in other scenarios where the best-fit model may not accurately capture the asymptotic scaling due to the effects of lower-order terms. One useful method for detecting this is to examine the residual plots generated by ESA, and in partic-

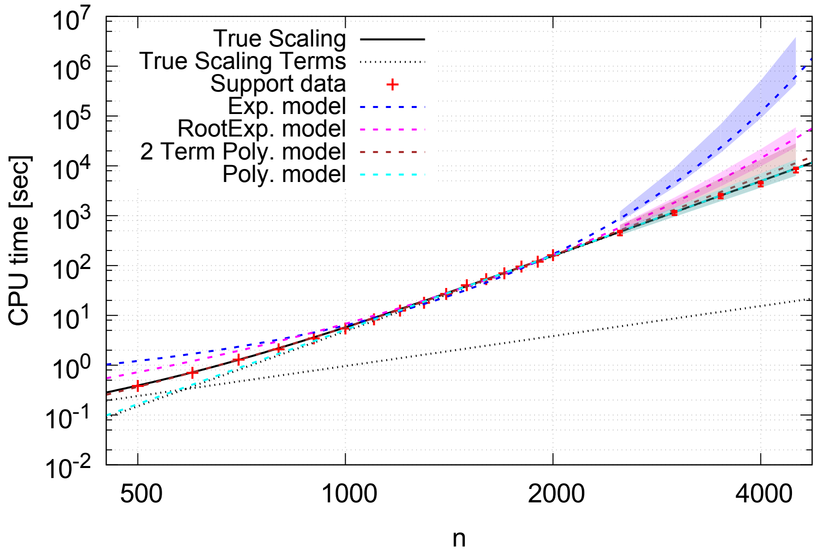


Fig. 7. **Early transition example:** generated using $4.8 \cdot 10^{-15} \cdot x^5 + 9.6 \cdot 10^{-7} \cdot x^2$ with 16 support instance sizes.

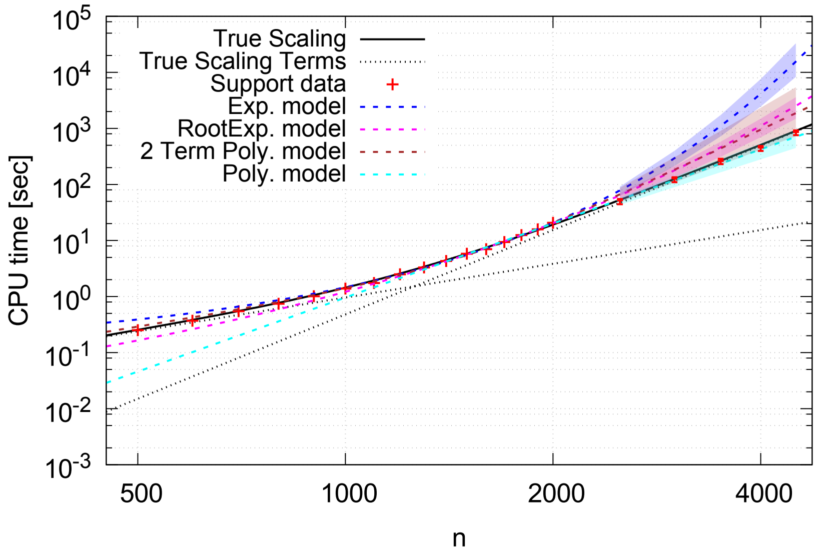


Fig. 8. **Mid transition example:** generated using $4.8 \cdot 10^{-16} \cdot x^5 + 9.6 \cdot 10^{-7} \cdot x^2$ with 16 support instance sizes.

ular to zoom in on the smallest support instance sizes to look for systematic deviations or trends indicating that the model may not accurately predict the data for smaller instance sizes (and hence may also not fit the data for larger instance sizes).

We also observed that the size of the bootstrap intervals for predicted running times continues to increase for the four-parameter model. This is even more clear for the case with only 12 support instance sizes (data not shown), where the bootstrap intervals for predicted running times are very large ($4.4 \cdot 10^4$ for instance size 4 500). When running ESA on a variant of this scenario with only 8 support instance sizes, we found that the Levenberg-Marquardt algorithm simply was unable to fit the four-parameter model to the data – even when the default fitting parameters were set to the true values for the running time scaling, the implementation of the Levenberg-Mardquart algorithm used in ESA simply

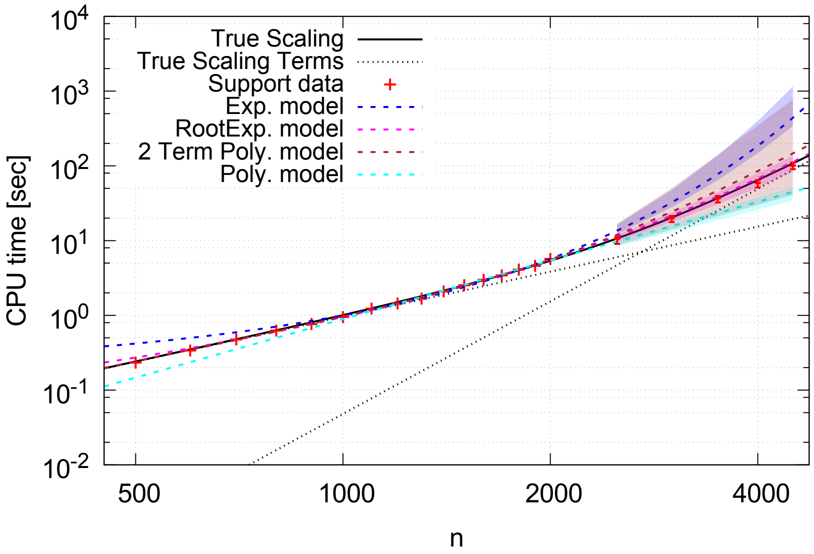


Fig. 9. **Late transition example:** generated using $4.8 \cdot 10^{-17} \cdot x^5 + 9.6 \cdot 10^{-7} \cdot x^2$ with 16 support instance sizes.

crashed.

What happens if we have prior knowledge about the lower-order terms? Practical applications of ESA to algorithms with competing terms may have known scaling for start-up costs; *e.g.*, the initialization of a data structure may be known to have quadratic scaling. Hence we may be inclined to use a scaling model that captures this prior knowledge, such as a three-parameter, two term polynomial model of the form $a \cdot n^b + c \cdot n^2$.

We ran ESA again on each of our 9 scenarios; however, this time we used the three-parameter, two-term polynomial model $a \cdot n^b + c \cdot n^2$. Overall, the results produced by ESA did not change much. In a few cases, the fit of the 3-parameter model was slightly better for the small instance sizes; however, it appeared to remain unchanged for the challenge instance sizes. We also observed that the bootstrap intervals for predicted running times were slightly smaller and located slightly higher in most of the scenarios. The only exception to this observation was for the function $4.8 \cdot 10^{-15} \cdot x^5 + 9.6 \cdot 10^{-7} \cdot x^2$ when ESA was run with 8 support instance sizes (a mid-range transition scenario). We found that the bootstrap confidence intervals for predictions obtained from the four-parameter model were very large (comparable to those in Figure 9); however, the three-parameter model produced substantially smaller confidence intervals (comparable to those in Figure 8). Unfortunately, the Levenberg-Mardquart algorithm was still unable to run successfully on some of the 1 000 bootstrap samples of the $4.8 \cdot 10^{-17} \cdot x^5 + 9.6 \cdot 10^{-7} \cdot x^2$ data set when only 8 support instance sizes were used.

7. Successful applications

During the development of ESA, earlier versions were used in several projects to analyze the empirical scaling of high-performance algorithms for several widely studied $\mathcal{NP}$-hard problems. These early applications provided interesting results, as well as valuable insights that guided the development of the version of ESA presented here. In the following, we outline the findings obtained from these earlier applications.

The methodology underlying ESA was first used to study the empirical scaling of the running time of Concorde, a prominent TSP solver. Concorde represents the long-standing state of the art in exact TSP solving; it incorporates mechanisms based on over 50 years of research on the TSP and has been used to solve the largest non-trivial TSP instances for which provably optimal solutions are known [20, 21]. Using the methodology underlying ESA, Hoos & Stützle fitted an exponential model of the form $a \cdot b^n$ with $b \approx 1.003$ to the running times observed for Concorde on one of the most widely studied types of TSP instances – so-called random uniform Euclidean (RUE) instances [11]. Challenged on larger instance sizes, this model was rejected with 95% confidence in favour of a square-root exponential model of the form $a \cdot b^{\sqrt{n}}$, with $b \approx 1.24$. This clearly indicates that solving this widely studied class of TSP instances, which up to this point were believed to be challenging, is substantially easier than expected based on theoretical worst-case complexity results.

In another application of the ESA methodology, Dubois *et al.* studied the scaling for the state-of-the-art inexact TSP solvers, EAX [16] and LKH [22] on the same set of RUE instances [12]. They found evidence that the running time of EAX shows square-root exponential scaling, with small support for polynomial scaling with degree $b \approx 1.95$. The scaling they observed for LKH 1.3 was only consistent with the square-root exponential model. For LKH 2, they found that there was also some evidence that the scaling was between a square-root exponential model and a polynomial model with degree $b \approx 2.9$. However, all three algorithms showed better scaling of the running time required for finding optimal solutions than Concorde, with bases $b \approx 1.12, 1.20$ and 1.19 for the square-root exponential models fitted for EAX, LKH 1.3 and LKH 2, respectively. Later, these results were improved using much longer runs of Concorde to find the optimal solutions for nearly all of the instances used in these experiments (see Section 6.3 in [13]). ESA indicated that the performance of EAX was inconsistent with a polynomial model, but was still mostly consistent with a square-root exponential model, while for LKH 2, the observed scaling falls between a square-root exponential and a polynomial model of degree $b \approx 2.9$. These studies demonstrate how the methodology underlying ESA can reveal substantial differences in empirical performance scaling between different state-of-the-art algorithms, and between different versions of the same algorithm. Furthermore, they indicate qualitative differences in the empirical complexity of state-of-the-art exact and inexact TSP solvers.

Finally, Mu *et al.* used ESA to investigate the impact of parameter settings and automated algorithm configuration on the performance scaling of the two inexact TSP algorithms [14]. For EAX, algorithm configuration helps improve the scaling, which can be further improved by adapting the population size with instance size. In particular, they achieved a ≈ 1.13-fold improvement in the median running time for EAX to solve RUE instances of size $n = 4\,500$ and found evidence for more substantial improvements on larger instances. For LKH, significant impact of parameter settings on performance scaling was observed; however, the state-of-the-art algorithm configurator SMAC [23] tends to overfit the running times for smaller instances and thus produces configurations for which LKH scales worse. These results indicate that parameter settings of heuristic, state-of-the-art algorithms for computationally challenging problems, such as the TSP, can impact the qualitative scaling behaviour (*i.e.*, lead to improvements in running times beyond constant factors). Unfortunately, they also reveal that current automated algorithm configuration methods may not be able to realise those improvements, since they do not sufficiently take into account performance scaling.

In a second line of work, the empirical scaling analysis approach underlying ESA has been used to study high-performance solvers for the propositional satisfiability problem (SAT). Specifically, Mu & Hoos studied three prominent, incomplete SAT solvers based on stochastic local search (SLS) [24]: BalancedZ [25], WalkSAT/SKC [26] and probSAT [27]. They also studied three prominent DPLL-based, complete solvers, kcnfs [28], march_hi [29] and march_br [30] (version SAT+UNSAT), on random phase-transition 3-SAT instances [4]. For each algorithm, they used ESA to investigate a polynomial model ($a \cdot n^b$) and an exponential model ($a \cdot b^n$) of scaling of running time with instance size. For the SLS-based algorithms, they analysed the scaling of median running times on the satisfiable instances and found that the observed scaling was consistent with a polynomial model with degree $b \approx 3$, whereas the exponential model was inconsistent with most of the observations for the challenge data. Looking at the confidence intervals for the polynomial model parameters for the three SLS-based solvers, they found no evidence that any algorithm scaled significantly better than any other. The DPLL-based, complete SAT solvers were analysed on the satisfiable and unsatisfiable instances; here, the median running time was found to be consistent with an exponential model with $b \approx 1.03$ and inconsistent with the polynomial model. This was confirmed to also be the case when considering only the satisfiable instances. These results clearly indicate that random-3-SAT at the solubility phase transition is hard for state-of-the-art complete SAT solvers, yet easy for cutting-edge SLS-based algorithms – a result that calls into doubt the suitability of this class of instances as a benchmark that captures the complexity of SAT.

Mu later expanded this analysis to two classes of random 4-SAT instances; this work, described in detail in Section 5.4 of [13], yielded several interesting results. On random phase-transition 4-SAT instances, an exponential model with $b \approx 1.02$ was found to be consistent with the observed performance data for BalancedZ, whereas a root-exponential model (which was fitted with $b \approx 2.8$ for WalksSAT/SKC and $b \approx 1.6$ for probSAT) was found to accurately describe the performance scaling of the other two SLS-based, incomplete SAT solvers. The DPLL-based, complete solvers continued to demonstrate scaling behaviour consistent with exponential models; however, the degree of the model ($b \approx 1.1$) was found to be significantly higher than in the case of 3-SAT instances. Mu also studied WalkSAT/SKC and kcnfs on a class of random 4-SAT instances below the phase transition that are believed to be intrinsically challenging and showed that both solvers scale significantly better than on phase-transition random instances, in that a polynomial model best describes the observed performance scaling. As in the case of random-3-SAT, these results challenge prior beliefs and assumptions that were based on combinations of theoretical complexity results and simpler forms of empirical performance assessment, and open interesting avenues for future investigation.

Of course, one may wonder how all these results compare to those produced by the improved version of ESA presented in this article. Our biggest change to the methodology underlying ESA is the addition of the nested bootstrap sampling procedure to handle multiple independent runs of a randomized algorithm on a given problem instance. However, considering the results from Section 5, we would not expect this modification to the method to substantially affect these earlier results. Unfortunately, the only remaining copies of the data from earlier studies contain per-instance median running times, so we were unable to test this hypothesis using direct comparisons of the earlier and most recent versions of ESA on the original data. We therefore re-ran EAX on the same TSP RUE instance set (using the improved optimality results by Mu [13]), but this time, we performed 11 independent runs per instance. On this data, the version of ESA described here yielded results that are qualitatively very similar to those reported by Mu *et al.*. [14]. We found that the sizes of the bootstrap confidence intervals for the observed running times on challenge instances increased by 4.2% on average, where the size of the interval is defined as its upper bound divided by its lower bound. Similarly, the size of the confidence intervals for the square-root exponential model that best describes the running times increased by 0.1% on average. This is unsurprising – we expect an increase in confidence interval size, because the resampling over multiple independent runs per instance allows us to capture additional variability in the scaling models due to randomization of the target algorithm, whereas previously, the statistical nature of the observed median running times was not taken into account. In addition, the relatively small size of the increase is consistent with our observations in Section 5.

The only other change we made to ESA's methodology was to include for the first time the decision model described in Section 2 used to automatically summarize the scaling results. However, while this enhancement augments the previous scaling analysis results and provides a well-defined, principled and rigorous way for assessing the scaling models, it does not change any of the previous results. As an example, the new procedure describes the exponential model predictions as being over-estimates for the observed scaling of the three SLS-based SAT solvers on the random 3-SAT phase transition instances and characterises the polynomial model as being a very good fit for the observations for WalkSAT/SKC and probSAT; however, because the polynomial model predictions are only weakly consistent with the observations for the largest two instance sizes for BalancedZ, it describes the polynomial scaling model for BalancedZ as being a fair fit.

8. Conclusions and future work

In this work, we introduced the empirical scaling analyzer (ESA), an automated tool for analyzing the empirical scaling of algorithm performance with input size. ESA can fit multiple models on running time data collected for a given algorithm across a series of inputs of varying sizes and generate results in the form of technical reports. These reports contain easy-to-read figures and tables, as well as automatically generated interpretations. We also presented new methodology to appropriately handle the variance between independent runs of a randomized algorithm and a novel method for automatically interpreting the scaling analysis results.

We presented a rigorous analysis of ESA's performance on several types of challenging scenarios. In many cases, if ESA's output appears unreliable (*e.g.*, the size of bootstrap confidence intervals for the model predictions is large for all of the fitted models), more data is needed – perhaps running times for more instance sizes, larger challenge instance sizes, more instances per size or more independent runs per instance. In particular, we found that increasing the number of instances per instance size is a more cost-effective means of increasing the power of the statistical analyses performed by ESA than increasing the number of independent runs per instance. In addition, increasing the extrapolation distance by using larger challenge instance sizes is one of the most reliable (albeit costly) means to increase ESA's ability to discriminate between different scaling models. Based on our extensive empirical analysis, we also caution against using small numbers of support instance sizes, since this can make it challenging or impossible to identify detrimental effects of lower-order terms or outliers on the results obtained from ESA. From our experience, we recommend to use around 11 support instance sizes, although the exact number required varies between application scenarios.

We found that ESA can correctly recognize the asymptotic performance scaling of a given algorithm when lower-order terms are present, provided that the transition between the two competing terms of the scaling model occurs towards the lower end of the support instance sizes used for the scaling analysis. However, increasing the number of parameters in a parametric model to capture the lower-order terms and the asymptotic scaling substantially increases the size of the bootstrap confidence intervals for the model performance predictions and can cause ESA to experience difficulties in fitting the models. If the effect of a lower-order term dominates that of the asymptotic scaling behaviour across all support instance sizes, ESA is likely to correctly recognize the true asymptotic scaling.

Overall, we have found that ESA is able to perform well in most scenarios. Unlike theoretical running time analysis, there is always the risk that a lower-order term is initially dominating the running time, and hence larger instance sizes are needed to accurately identify the true asymptotic scaling. Nevertheless, empirical scaling analysis plays a key role in characterizing and understanding the behaviour of high-performance algorithms for important problems. This is particularly true for scenarios where the observed performance of algorithms exceeds the expectations provided by a worst-case analysis, as well as in cases where theoretical assumptions about the expected behaviour of an algorithm may not hold for real-world instances. The methodology underlying ESA is widely applicable to problems and algorithms where running time data can be gathered for various instance sizes. ESA provides an easy and convenient way to apply empirical scaling analysis to algorithms of interest. Thus, we believe that ESA will prove to be a useful tool for researchers studying both the empirical and theoretical scaling behaviour of algorithms, and we hope that ESA will promote and enhance such studies.

There are several directions for future improvements of ESA. In particular, it would be interesting to automatically select models from a large family of functions based on input data. This could also facilitate fitting of models with lower-order terms. One possible approach towards this end involves repeated fitting of models, first on the original data, then on the residues, in order to obtain a model with several terms. It would also be interesting to use an automated machine learning method, such as Auto-WEKA [31] or auto-sklearn [32], to determine a scaling model; however, special care would need to be taken to preserve the statistical significance of the results through multiple testing correction (and perhaps through the use of a validation set with instances of intermediate size). Currently, one of ESA's biggest limiting factors is the requirement that instances be grouped by size. Since it is not always possible to collect instances grouped by size, it would be extremely useful for many practical application scenarios to develop new methodology for empirical scaling analysis that overcomes this limitation. Furthermore, we believe that many users of ESA may be motivated to find upper or lower bounds on the running times required to solve very large instances. To this end, future extensions of ESA could be developed that fit tight bounds on the running time scaling to provide users with such estimates.

ESA's implementation is currently restricted to analyzing a single feature, describing instance size or difficulty, at a time. However, applications where two or more features affect instance difficulty arise commonly. In principle, with relatively minor modifications to the methodology, ESA could be extended to study such scenarios. However, this would give rise to more complex classes of scaling models; as we have seen in Section 6, this can pose challenges for ESA. Therefore, we expect that substantial additional work may be required to achieve a usable, robust extension of ESA that can deal with multiple instance features. Nevertheless, such an improvement could be immensely useful, as it would also enable users to easily reason about the relative and absolute impact of various instance features on the running time of an algorithm. In principle, this type of comparative feature importance analysis could be performed by applying the existing version of ESA multiple times with different instance features. However, special attention would be required when generating or collecting instance sets. Ideally, only one parameter controlling instance difficulty should be varied at a time when using ESA; otherwise, interaction effects may invalidate the results. If it is impractical to obtain one such instance set for each feature studied, it might be possible to vary all of the features independently and identically across the instance set; however, this would substantially increase the effective variability in the running time of the algorithm, and may make it practically impossible to obtain statistically significant results.

Another interesting avenue of study is the performance of ESA when used to analyze the scaling of polynomial-time algorithms. Preliminary results indicate that such algorithms tend to show significantly less statistical variation in their running times than the heuristic, $\mathcal{NP}$-hard algorithms that have been the primary subject of our study so far. Very small bootstrap intervals can provide new challenges for ESA that will need to be overcome in future extensions, since in such cases, all of the fitted models tend to be rejected. The introduction of tight upper and lower bounds into the methodology underlying ESA may provide a way to overcome this challenge.

The current version of ESA is limited to given, static datasets. However, in practice, users may find after running ESA that there is insufficient data to yield statistically meaningful results. In this case, a user would likely want to collect more running time data and repeat the analysis. Future versions of ESA could be extended to automatically alert the user that more data would be beneficial. Alternatively, ESA could be modified to perform additional runs of the algorithm to collect more data automatically. Similarly, a variant of ESA could be developed that automatically interleaves the collection of running time data with scaling analysis, until there is sufficient evidence to reject with 95% confidence all but one of the candidate scaling models, thereby minimizing the amount of running time data that needs to be collected.

In addition, we are currently working on uses of ESA in the development of automated algorithm configuration procedures for better scaling behaviour. Such procedures could make automated configuration even more applicable to real-world situations, as problem instances of practical interest can take a long time to solve. Automated configuration usually requires many runs of the given target algorithm with different parameter settings, which can make it impractical to run a configuration procedure directly on large, challenging instances. Previous work on the problem of automatically configuring algorithms for improved performance scaling has focused on generic protocols for using existing configurators [33, 34]. An alternative consists of incorporating empirical scaling analysis, as performed by ESA, more directly into algorithm configuration. Unfortunately, the current version of ESA requires running time data for many problem instances of different sizes, which can take a long time to produce. Thus, it will be important to design a way to reduce the time, possibly by leveraging previously fitted models, *e.g.*, by integrating Bayesian methods into empirical scaling analysis, with a previous model acting as the prior for model fitting. This could lead to an enhanced version of ESA that could then be integrated into a future configuration procedure.

Acknowledgements

Yasha Pushak was supported by an NSERC Vanier Scholarship. Holger Hoos acknowledges funding through an NSERC Discovery Grant, CFI JLEF funding and startup funding from Universiteit Leiden.

References

[1] D. Kunkle, Empirical Complexities of Longest Common Subsequence Algorithms, Technical Report, Computer Science Department, Rochester Institute of Technology, NY, USA, 2002.

[2] K. Subramani and D. Desovski, On the empirical efficiency of the vertex contraction algorithm for detecting negative cost cycles in networks, in: *Proceedings of the 5th International Conference on Computational Science (ICCS 2005)*, Springer, 2005, pp. 180–187.

[3] R. Aguirre-Hernández, H.H. Hoos and A. Condon, Computational RNA secondary structure design: Empirical complexity and improved methods, *BMC Bioinformatics* **8**(1) (2007), 34.

[4] Z. Mu and H.H. Hoos, On the empirical time complexity of random 3-SAT at the phase transition, in: *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI 2015)*, 2015, pp. 367–373.

[5] Q. Sun, Sampling-based prediction of algorithm runtime, PhD thesis, The University of Waikato, 2009.

[6] D. Zaparanuks and M. Hauswirth, Algorithmic profiling, *ACM SIGPLAN Notices* **47**(6) (2012), 67–76.

[7] C. McGeoch, P. Sanders, R. Fleischer, P.R. Cohen and D. Precup, Using finite experiments to study asymptotic performance, in: *Experimental Algorithmics*, Springer, 2002, pp. 93–126.

[8] E. Coppa, C. Demetrescu and I. Finocchi, Input-sensitive profiling, *ACM SIGPLAN Notices* **47**(6) (2012), 89–98.

[9] E. Coppa, C. Demetrescu and I. Finocchi, Input-sensitive profiling, *IEEE Transactions on Software Engineering* **40**(12) (2014), 1185–1205.

[10] H.H. Hoos, A bootstrap approach to analysing the scaling of empirical run-time data with problem size, Technical Report, TR-2009-16, Department of Computer Science, University of British Columbia, 2009.

[11] H.H. Hoos and T. Stützle, On the empirical scaling of run-time for finding optimal solutions to the travelling salesman problem, *European Journal of Operational Research* **238**(1) (2014), 87–94.

[12] J. Dubois-Lacoste, H.H. Hoos and T. Stützle, On the Empirical Scaling Behaviour of State-of-the-art Local Search Algorithms for the Euclidean TSP, in: *Proceedings of the 17th International Genetic and Evolutionary Computation Conference (GECCO 2015)*, 2015, pp. 377–384.

[13] Z. Mu, Analysing the empirical time complexity of high-performance algorithms for SAT and TSP, Master's thesis, University of British Columbia, Vancouver, Canada, 2015.

[14] Z. Mu, H.H. Hoos and T. Stützle, The Impact of Automated Algorithm Configuration on the Scaling Behaviour of State-of-the-Art Inexact TSP Solvers, in: *Proceedings of the 10th International Conference on Learning and Intelligent Optimization (LION 2016)*, LNCS, Vol. 10079, 2016, pp. 157–172.

[15] Z. Mu, J. Dubois-Lacoste, H.H. Hoos and T. Stützle, On the empirical scaling of running time for finding optimal solutions to the TSP, *Journal of Heuristics* (2017), 1–20.

[16] Y. Nagata and S. Kobayashi, A Powerful Genetic Algorithm Using Edge Assembly Crossover for the Traveling Salesman Problem, *Institute for Operations Research and the Management Sciences Journal on Computing (INFORMS JOC)* **25**(2) (2013), 346–363.

[17] Y. Nagata, Edge Assembly Crossover: A High-power Genetic Algorithm for the Traveling Salesman Problem, in: *Proceedings of the 7th International Conference on Genetic Algorithms (ICGA 1997)*, 1997, pp. 450–457.

[18] H.H. Hoos and T. Stützle, Towards a Characterisation of the Behaviour of Stochastic Local Search Algorithms for SAT, *Artificial Intelligence* **112**(1–2) (1999), 213–232, ISSN 0004-3702.

[19] M. Birattari, On the Estimation of the Expected Performance of a Metaheuristic on a Class of Instances, Technical Report, Technical Report TR/IRIDIA/2004-01, IRIDIA, Université Libre de Bruxelles, 2004.

[20] D.L. Applegate, R.E. Bixby, V. Chvátal and W.J. Cook, *The Traveling Salesman Problem: A Computational Study*, Princeton University Press, 2006.

[21] D.L. Applegate, R.E. Bixby, V. Chvátal and W.J. Cook, The traveling salesman problem, concorde TSP solver (2012), Last visited on 2019-02-05. http://www.tsp.gatech.edu/concorde.

[22] K. Helsgaun, An effective implementation of the Lin–Kernighan traveling salesman heuristic, *European Journal of Operational Research* **126**(1) (2000), 106–130.

[23] F. Hutter, H.H. Hoos and K. Leyton-Brown, Sequential model-based optimization for general algorithm configuration, in: *Proceedings of the 5th International Conference on Learning and Intelligent Optimization (LION 2011)*, Springer, 2011, pp. 507–523.

[24] H.H. Hoos and T. Stützle, *Stochastic Local Search: Foundations & Applications*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005. ISBN 1558608729.

[25] C. Li, C. Huang and R. Xu, Balance between intensification and diversification: a unity of opposites, in: *Proceedings of the 2014 SAT Competition*, 2014, pp. 10–11.

[26] B. Selman, H.A. Kautz and B. Cohen, Noise strategies for improving local search, in: *Proceedings of the 12th International Conference on Artificial Intelligence (AAAI 1994)*, 1994, pp. 337–343.

[27] A. Balint and U. Schöning, probSAT and pprobSAT, in: *Proceedings of the 2014 SAT Competition*, 2014, p. 63.

[28] G. Dequen and O. Dubois, Kcnfs: An efficient solver for random k-SAT formulae, in: *Proceedings of The 7th International Conference on Theory and Applications of Satisfiability Testing (SAT 2004)*, Springer, 2004, pp. 486–501.

[29] M.J. Heule and H. van Marren, march_hi: Solver description, in: *Proceedings of the 2009 SAT Competition*, 2009, pp. 23–24.

[30] M.J. Heule, march_br, in: *Proceedings of 2013 SAT Competition*, 2013, p. 53.

[31] C. Thornton, F. Hutter, H.H. Hoos and K. Leyton-Brown, Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms, in: *Proceedings of the 19th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2013)*, 2013, pp. 847–855.

[32] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum and F. Hutter, Efficient and Robust Automated Machine Learning, in: *Proceedings of the 28h International Conference on Advances in Neural Information Processing Systems (NeurIPS 2015)*, Curran Associates, Inc., 2015, pp. 2962–2970. http://papers.nips.cc/paper/5872-efficient-and-robust-automated-machine-learning.pdf.

[33] J. Styles, H.H. Hoos and M. Müller, Automatically Configuring Algorithms for Scaling Performance, in: *Proceedings of the 6th International Conference on Learning and Intelligent Optimization (LION 2012)*, 2012, pp. 205–219.

[34] J. Styles and H.H. Hoos, Ordered Racing Protocols for Automatically Configuring Algorithms for Scaling Performance, in: *Proceedings of the 15th International Conference on Genetic and Evolutionary Computation (GECCO 2013)*, 2013, pp. 551–558.