User Guide
ESA v1.1

Zongxu Mu and Yasha Pushak
Department of Computer Science
University of British Columbia
{zongxumu,ypushak}@cs.ubc.ca

27th July 2017

In this document, we introduce an automated tool – the Empirical Scaling Analyzer (ESA) – that can perform core elements of the analysis described in [2]. To use ESA, a user needs to prepare an input file of running time data of an algorithm (referred to as target algorithm hereafter), as well as other optional files, including a configuration file, a file specifying the parametric models to be fitted, a LaTeX template and a gnuplot template. Details of these input files will be given in Section 1. Note that 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 put all running time data into a file, feed it into ESA and obtain the results from the scaling analysis using several parametric models. Results are presented in a technical report, which contains easy-to-read tables and figures for the scaling of the target algorithm. The details of the output report are described in Section 2.

We believe the tool is useful for other researchers who want to study the empirical time complexity of algorithms, and can thus promote the use of such analysis for other problems and algorithms. The tool is available as an easy-to-use online service at `www.cs.ubc.ca/labs/beta/Projects/ESA/esa-online.html` and as a command-line tool with additional functionality (see Section 5 on how to run ESA). Here, we describe all features available in the command-line version.

1 Input

To perform scaling analysis, ESA requires input data containing the sizes of the instances studied and the running times the target algorithm requires for solving these instances. The input running time data need to follow the following formatting rules:

- the input file contains lines of details of the instances, one instance per line;
- in each line, the following pieces of information are provided in order and are separated by ",":

```
# instance name, size, datum (running time), datum, datum
portgen-500-1000.tsp, 500, 2.3, 2.54, 2.09
portgen-500-100.tsp, 500, 2.58, 2.28, 2.71
portgen-500-101.tsp, 500, 2.36, 2.44, 2.53
portgen-500-102.tsp, 500, 2.51, 2.21, 2.08
portgen-500-103.tsp, 500, 2.63, 2.65, 3.01
portgen-500-104.tsp, 500, 2.84, 2.98, 2.26
portgen-500-105.tsp, 500, 2.62, 2.24, 2.05
portgen-500-106.tsp, 500, 1.91, 2.45, 3
...
portgen-600-1000.tsp, 600, 3.4, 3.54, 4.01
...
portgen-4500-10.tsp, 4500, 727.68, inf, 801.32
portgen-4500-11.tsp, 4500, inf, inf, inf
...
#instances, 4000, 100
#instances, 4500, 100
```

Figure 1: Excerpt of an input file for ESA, where deleted lines are represented by "...". The last two columns in this file are optional, and would not be used for deterministic algorithms, since they produce consistent running times across independent runs.

- instance name (e.g., file name) and other optional information (this field is for the user's reference only; ESA does not use this field in the scaling analysis);
- instance size (as an integer);
- running time required to solve the instance, which itself may be a statistic for multiple runs of the target algorithm solving the instance and may be "inf" for time-out or crashed runs; or, a list of running times each separated by ",", in the case of multiple target algorithm runs per instance for randomized algorithms.

Besides, the user may specify the number of instances for some sizes. If there are not enough entries for one size, ESA will treat the missing entries as instances with unknown running time. An example for such data is described in [1], in the context of analyzing the scaling behaviour of EAX and LKH, where the running times of some instances are unknown because no optimal solution has been found in previous runs of Concorde. An excerpt of an input file for ESA is in Figure 1.

ESA also takes as input a configuration file, containing details on the target algorithm, the instance distribution and various parameters for ESA. The file contains lines of configurations, one configuration per line. Each configuration follows the "`name : value`" format. An example of a configuration file is shown in Figure 2. A complete list of the options that can be set in the configuration file can be found in Appendix A.

```
fileName : runtimes.csv
algName : WalkSAT/SKC
instName : random 3-SAT instances at phase transition
modelFileName : models.txt
numTrainingData : 7
alpha : 95
numBootstrapSamples : 1000
statistic : median
latexTemplate : template-AutoScaling.tex
modelPlotTemplate : template_plotModels.plt
residuePlotTemplate : template_plotResidues.plt
```

Figure 2: Example of a configuration file for ESA.

There are a number of other files that a user may supply (if not supplied, ESA will use the default file(s) distributed with the code), including:

- a file specifying the models to be fit;
- a LaTeX template specifying the content and format of the output report;
- gnuplot templates specifying the format of the plots.

The first of these is needed, because ESA supports customized models, as long as the models are supported by python (including the math and the numpy packages) and gnuplot. This file contains lines of models, one model per line. Each contains the following items, separated by ",":

- Model name (e.g., Exponential);
- Number of parameters (e.g., 2);
- LaTeX expression of the model;
- Python expression of the model;
- Gnuplot expression of the model;
- Default values of the parameters, separated by ",".

For all expressions of the models, x represents the size, and the parameters should be $a, b, \ldots$, and should be surrounded by "@@". For example, the specification in Figure 3, tells ESA to fit an exponential and a polynomial model of the form $a \cdot b^x$ and $a \cdot x^b$ respectively:

For the LaTeX template, ESA will use the default template, if no customized template is found. In the template, dynamic values should be surrounded by "@@". For instance, the name of the algorithm (which is defined in the configuration file) is a dynamic value. Wherever mentioned in the template file, the user should use "@@algName@@", and ESA will instantiate it to be the real name of the algorithm when

```
Exp,2,@@a@@\times @@b@@^{x},@@a@@*@@b@@**x,@@a@@*@@b@@
    **x,1e-4,1.01
Poly,2,@@a@@\times x^{@@b@@},@@a@@*x**@@b@@,@@a@@*x**
    @@b@@,1e-8,1
```

Figure 3: An example of model specification for ESA.

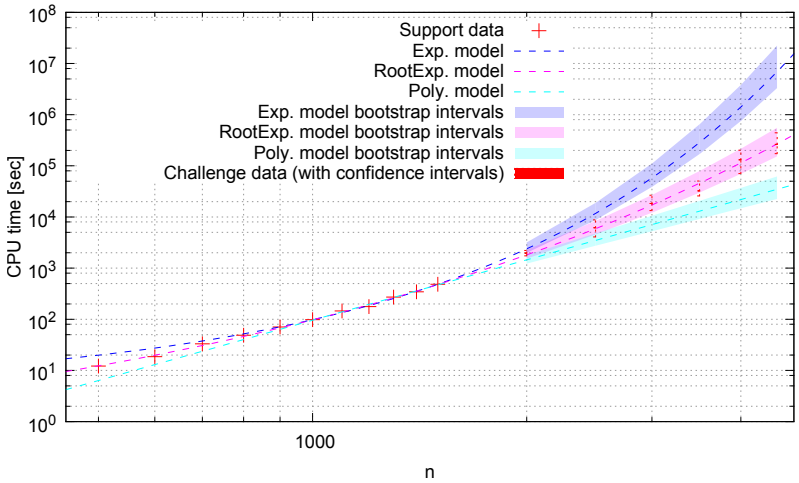


Figure 4: Example output of ESA – a figure of running times, fitted models and corresponding bootstrap confidence intervals of each model.

generating the report. Users can also specify the formats of the plots via the template gnuplot script. For instance, users may choose whether to use a log-log plot or a semi-log plot via the template gnuplot script. Default templates are available for download together with the source code (see Section 5 for details).

2 Output

ESA automatically generates a technical report containing detailed empirical scaling analysis results and interpretation. This report contains tables and figures that users can easily read, including:

- two tables of statistics of running times, one for support data and the other for challenge, as illustrated in Tables 1 & 2, respectively;
- a table of fitted models and corresponding RMSE values, as illustrated in Table 3;
- a figure of running times, fitted models and corresponding bootstrap confidence intervals of each model, as illustrated in Figure 4;

n	500	600	700	800
# instances	1000	1000	1000	1000
# running times	1000	1000	1000	1000
mean	19.33	31.55	56.8	89.35
coefficient of variation	1.164	1.418	1.303	1.455
Q(0.1)	0.43	2.21	6.73	10.62
Q(0.25)	3.75	8.75	15.3	22.31
median	12.22	18.64	33.15	48.95
Q(0.75)	25.01	37.64	68.29	102.5
Q(0.9)	47.33	70.35	130.4	195.5

n	900	1000	1100	1200
# instances	1000	1000	1000	1000
# running times	1000	1000	1000	1000
mean	139.7	201.2	314.6	385.4
coefficient of variation	1.734	1.759	1.851	1.713
Q(0.1)	17.23	23.28	28.8	38.76
Q(0.25)	32.72	43.23	60.84	78.66
median	70.64	98.56	145.7	177.2
Q(0.75)	142.7	216.1	341.3	409.2
Q(0.9)	302.7	429.4	693.2	846.3

n	1300	1400	1500
# instances	1000	1000	1000
# running times	1000	1000	1000
mean	548.7	749	1072
coefficient of variation	1.859	2.227	2.109
Q(0.1)	53.27	73.78	93.04
Q(0.25)	112.2	153.3	210.1
median	271.7	344.3	483.5
Q(0.75)	583.6	783.6	1136
Q(0.9)	1190	1517	2277

Table 1: Example output of ESA – statistics of running times for support data.

n	2000	2500	3000
# instances	1000	100	100
# running times	1000	100	100
mean	5402	∞	∞
coefficient of variation	2.624	N/A	N/A
Q(0.1)	307	671.5	3538
Q(0.25)	765.5	1694	8188
median	1969	6149	1.84×10^4
Q(0.75)	5207	1.766×10^4	4.118×10^4
Q(0.9)	1.137×10^4	4.611×10^4	9.048×10^4

n	3500	4000	4500
# instances	100	100	100
# running times	100	100	100
mean	∞	∞	∞
coefficient of variation	N/A	N/A	N/A
Q(0.1)	6060	2.096×10^4	3.041×10^4
Q(0.25)	1.226×10^4	4.125×10^4	1.22×10^5
median	3.246×10^4	1.312×10^5	2.633×10^5
Q(0.75)	9.717×10^4	3.938×10^5	∞
Q(0.9)	2.76×10^5	∞	∞

Table 2: Example output of ESA – statistics of running times for challenge data.

		Model	RMSE (support)	RMSE (challenge)
Concorde	Exp. Model	4.0388×1.0032^{x}	7.7847	2.7852×10^{6}
	RootExp. Model	$\mathbf{0.083457} \times \mathbf{1.2503}^{\sqrt{\mathbf{x}}}$	**7.0439**	**9169.4**
	Poly. Model	$1.6989 \times 10^{-10} \times x^{3.9176}$	9.9327	1.038×10^{5}

Table 3: Example output of ESA – fitted models and corresponding RMSE values.

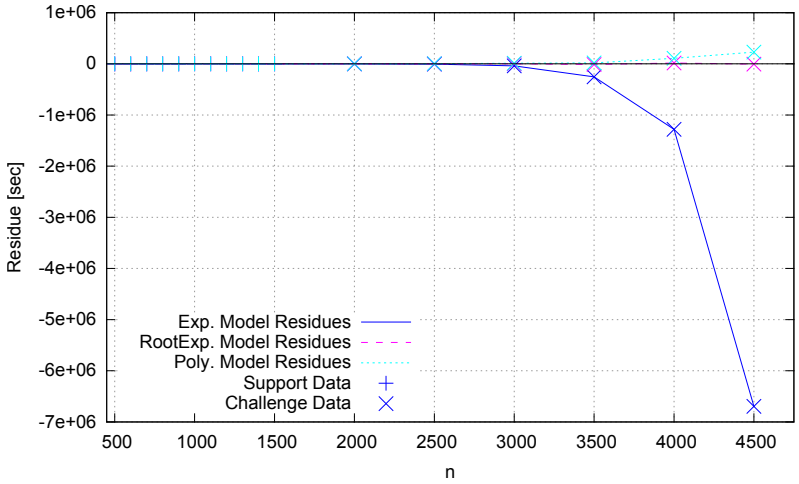


Figure 5: Example output of ESA – a figure of residues of the fitted models.

- a figure of residues of the fitted models, as illustrated in Figure 5;
- a table of bootstrap confidence intervals for all model parameters, as illustrated in Table 4;
- a table of medians and bootstrap confidence intervals for the model support and challenge RMSE, as illustrated in Table 5;
- two tables of bootstrap confidence intervals for observed and predicted running times, one for support data and the other for challenge, as illustrated in Tables 6 & 7.

A snapshot of the reported generated by ESA using the default LaTeX template is shown in Figure 6.

Solver	Model	Confidence interval of a	Confidence interval of b
Concorde	Exp.	$[2.6108, 5.2975]$	$[1.003, 1.0036]$
	RootExp.	$[0.037056, 0.15111]$	$[1.2287, 1.2793]$
	Poly.	$[6.1872 \times 10^{-12}, 1.7351 \times 10^{-9}]$	$[3.5859, 4.3713]$

Table 4: Example output of ESA – bootstrap confidence intervals for all model parameters.

Solver	Model	Support RMSE		Challenge RMSE	
		Median	Confidence Interval	Median	Confidence Interval
LKH	Exp.	0.24191	[0.62545, 1.9637]	10333	[4483.9, 30248]
	RootExp.	0.28221	[0.51392, 1.9043]	798.71	[297.93, 2239.3]
	Poly.	**0.41045**	[**0.59351**, **2.1016**]	**102.41**	[**36.443**, **288.48**]

Table 5: Example output of ESA – bootstrap confidence intervals and medians for the model RMSEs.

Solver	n	Predicted confidence intervals	Observed median run-time	
		Exp. model	Point estimates	Confidence intervals
Concorde	500	[15.43, 23.47]	12.22	[11.01, 13.33]
	600	[22, 31.52]	18.64	[16.99, 20.16]
	700	[**31.38**, **42.57**]#	33.15	[30.38, 35.9]
	800	[**44.68**, **57.59**]#	48.95	[44.05, 53.15]
	900	[**63.49**, **77.63**]*	70.64	[64.34, 77.28]
	1000	[**90.18**, **104.8**]#	98.56	[90.16, 107.8]
	1100	[**127.6**, **141.9**]	145.7	[130.9, 158.6]
	1200	[**178.6**, **193**]	177.2	[165.8, 196.1]
	1300	[**244.7**, **266.8**]	271.7	[244.9, 298.3]
	1400	[**332.7**, **375.7**]#	344.3	[318.9, 383.3]
	1500	[**448.8**, **534.7**]#	483.5	[432.8, 532.9]

Solver	n	Predicted confidence intervals	Observed median run-time	
		RootExp. model	Point estimates	Confidence intervals
Concorde	500	[**9.176**, **15.31**]*	12.22	[11.01, 13.33]
	600	[**15.56**, **23.81**]*	18.64	[16.99, 20.16]
	700	[**25.21**, **35.83**]#	33.15	[30.38, 35.9]
	800	[**39.43**, **52.46**]#	48.95	[44.05, 53.15]
	900	[**60.23**, **74.99**]#	70.64	[64.34, 77.28]
	1000	[**89.56**, **105.2**]#	98.56	[90.16, 107.8]
	1100	[**130.4**, **145.2**]	145.7	[130.9, 158.6]
	1200	[**184.4**, **199.1**]	177.2	[165.8, 196.1]
	1300	[**252**, **274**]#	271.7	[244.9, 298.3]
	1400	[**336.6**, **380.4**]#	344.3	[318.9, 383.3]
	1500	[**442**, **522.8**]#	483.5	[432.8, 532.9]

Solver	n	Predicted confidence intervals	Observed median run-time	
		Poly. model	Point estimates	Confidence intervals
Concorde	500	[4.206, 8.58]	12.22	[11.01, 13.33]
	600	[9.398, 16.47]	18.64	[16.99, 20.16]
	700	[18.41, 28.62]	33.15	[30.38, 35.9]
	800	[**32.91**, **46.48**]	48.95	[44.05, 53.15]
	900	[**55.04**, **71.24**]#	70.64	[64.34, 77.28]
	1000	[**86.98**, **104.3**]#	98.56	[90.16, 107.8]
	1100	[**131.4**, **147.6**]#	145.7	[130.9, 158.6]
	1200	[**188.5**, **204.4**]	177.2	[165.8, 196.1]
	1300	[**257.9**, **280.1**]#	271.7	[244.9, 298.3]
	1400	[**339.3**, **384.2**]#	344.3	[318.9, 383.3]
	1500	[**435.4**, **516.3**]#	483.5	[432.8, 532.9]

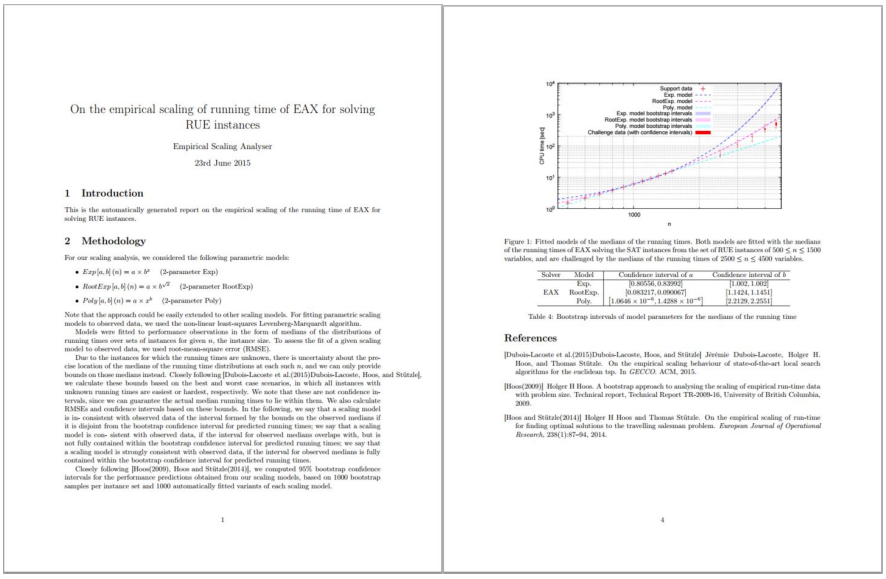
Table 6: Example output of ESA – bootstrap confidence intervals for observed and predicted running times for support data.

Solver	n	Predicted confidence intervals Exp. model	Observed median run-time Point estimates	Observed median run-time Confidence intervals
Concorde	2000	$[\mathbf{1988}, \mathbf{3179}]$	1969	$[1739, 2222]$
	2500	$[\mathbf{8718}, \mathbf{1.884 \times 10^4}]$	6149	$[4084, 8812]$
	3000	$[3.853 \times 10^4, 1.103 \times 10^5]$	1.84×10^4	$[1.332 \times 10^4, 2.669 \times 10^4]$
	3500	$[1.698 \times 10^5, 6.479 \times 10^5]$	3.246×10^4	$[2.581 \times 10^4, 5.038 \times 10^4]$
	4000	$[7.5 \times 10^5, 3.809 \times 10^6]$	1.312×10^5	$[7.073 \times 10^4, 2.024 \times 10^5]$
	4500	$[3.301 \times 10^6, 2.245 \times 10^7]$	2.633×10^5	$[1.73 \times 10^5, 4.419 \times 10^5]$

Solver	n	Predicted confidence intervals RootExp. model	Observed median run-time Point estimates	Observed median run-time Confidence intervals
Concorde	2000	$[\mathbf{1528}, \mathbf{2269}]$*	1969	$[1739, 2222]$
	2500	$[\mathbf{4536}, \mathbf{8335}]$#	6149	$[4084, 8812]$
	3000	$[\mathbf{1.212 \times 10^4}, \mathbf{2.694 \times 10^4}]$*	1.84×10^4	$[1.332 \times 10^4, 2.669 \times 10^4]$
	3500	$[\mathbf{3.001 \times 10^4}, \mathbf{7.925 \times 10^4}]$#	3.246×10^4	$[2.581 \times 10^4, 5.038 \times 10^4]$
	4000	$[\mathbf{6.95 \times 10^4}, \mathbf{2.163 \times 10^5}]$*	1.312×10^5	$[7.073 \times 10^4, 2.024 \times 10^5]$
	4500	$[\mathbf{1.528 \times 10^5}, \mathbf{5.563 \times 10^5}]$*	2.633×10^5	$[1.73 \times 10^5, 4.419 \times 10^5]$

Solver	n	Predicted confidence intervals Poly. model	Observed median run-time Point estimates	Observed median run-time Confidence intervals
Concorde	2000	$[\mathbf{1228}, \mathbf{1795}]$	1969	$[1739, 2222]$
	2500	$[\mathbf{2737}, \mathbf{4771}]$	6149	$[4084, 8812]$
	3000	$[5252, 1.057 \times 10^4]$	1.84×10^4	$[1.332 \times 10^4, 2.669 \times 10^4]$
	3500	$[9149, 2.069 \times 10^4]$	3.246×10^4	$[2.581 \times 10^4, 5.038 \times 10^4]$
	4000	$[1.477 \times 10^4, 3.708 \times 10^4]$	1.312×10^5	$[7.073 \times 10^4, 2.024 \times 10^5]$
	4500	$[2.248 \times 10^4, 6.205 \times 10^4]$	2.633×10^5	$[1.73 \times 10^5, 4.419 \times 10^5]$

Table 7: Example output of ESA – bootstrap confidence intervals for observed and predicted running times for support data.



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

Empirical Scaling Analyser

23rd June 2015

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:

- $Exp[a,b](n) = a \times b^n$ (2-parameter Exp)
- $RootExp[a,b](n) = a \times b^{\sqrt{n}}$ (2-parameter RootExp)
- $Poly[a,b](n) = a \times n^b$ (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 [Dubois-Lacoste et al.(2015)Dubois-Lacoste, Hoos, and Stützle], 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. In the following, we say that a scaling model is in- consistent with observed data of the interval formed by the bounds on the observed medians if it is disjoint from the bootstrap confidence interval for predicted running times; we say that a scaling model is con- sistent 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 interval for observed medians is fully contained within the bootstrap confidence interval for predicted running times.

Closely following [Hoos(2009), Hoos and Stützle(2014)], 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.

1

Figure 1: Fitted models of the medians of the running times. Both models are fitted with the medians of the running times of EAX solving the SAT instances from the set of RUE instances of $500 \leq n \leq 1500$ variables, and are challenged by the medians of the running times of $2500 \leq n \leq 4500$ variables.

Solver	Model	Confidence interval of a	Confidence interval of b
EAX	Exp.	[0.80556, 0.83992]	[1.002, 1.002]
	RootExp.	[0.083217, 0.090067]	[1.1424, 1.1451]
	Poly.	$[1.0646 \times 10^{-6}, 1.4288 \times 10^{-6}]$	[2.2129, 2.2551]

Table 4: Bootstrap intervals of model parameters for the medians of the running time

References

[Dubois-Lacoste et al.(2015)Dubois-Lacoste, Hoos, and Stützle] 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.

[Hoos(2009)] 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.

[Hoos and Stützle(2014)] 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

Figure 6: Snapshot of the technical report generated by ESA.

3 Automated Interpretation of Scaling Results

In addition, ESA generates automated interpretations for scaling analysis results. It evaluates how well a model fits the given data based on the percentage of challenge sizes for which the model predicts the corresponding running times reasonably accurately. If a model predicts well for most challenge sizes, then the model should be accepted as a good fit. Technically, the evaluation is based on the percentage of the challenge input sizes for which the model predictions are strongly, weakly, and inconsistent with the observed data; where we say a model is strongly consistent for a challenge input size if model's predicted bootstrap interval completely contains the observed bootstrap interval for that input size, a model is weakly consistent if the intersection of the model's predicted bootstrap interval with the observed bootstrap interval is non-empty and a model is in-consistent if the intersection of the two bootstrap intervals is empty. It especially emphasizes the challenge points for larger input sizes, as those provide more information regarding whether the model predicts well. The detailed criteria, which we design based on extensive experiments with SAT and TSP algorithms , as well as artificially generated data, are as follows:

- very good fit: we say a model fits the data very well if 90% or more of the predicted bootstrap intervals are strongly consistent with the observed data, or at least 90% of the predicted bootstrap intervals are weakly consistent with the observed data and 90% of the larger half of the predicted intervals are strongly consistent with the observed data;
- fair fit: we say a model tends to fit the data if 90% or more of the predicted bootstrap intervals (or the larger half of the predicted intervals) are weakly consistent with the observed data;
- tends to under/over-estimate: We say a model tends to under/over-estimate the data if more than 10% of the predicted bootstrap intervals (or the larger half of the predicted intervals) are in-consistent with the observed bootstrap intervals, and at least 95% of the predicted bootstrap intervals (or the larger half of the predicted intervals) are above/below or are consistent with the observed data;
- under/over-estimate: we say a model under/over-estimates the data if more than 70% of the predicted bootstrap intervals (or more than 70% of the larger half of the predicted intervals) are in-consistent with the observed bootstrap intervals and are below/above the observed intervals;
- poor fit: we say a model does not fit the data well if more than 10% of the predicted bootstrap intervals are in-consistent with the observed bootstrap intervals and more than 5% of the predicted intervals are above the observed intervals and more than 5% of the predicted intervals are below the observed intervals.

Note that when medians (or other quantiles) are not definitely known (due to instances with unknown running times), we compare the intervals of the medians against the predicted bootstrap intervals. To be more precise, for instance, we say a challenge point is below the predicted bootstrap interval, if the upper bound of the median is smaller than the lower bound of the bootstrap interval.

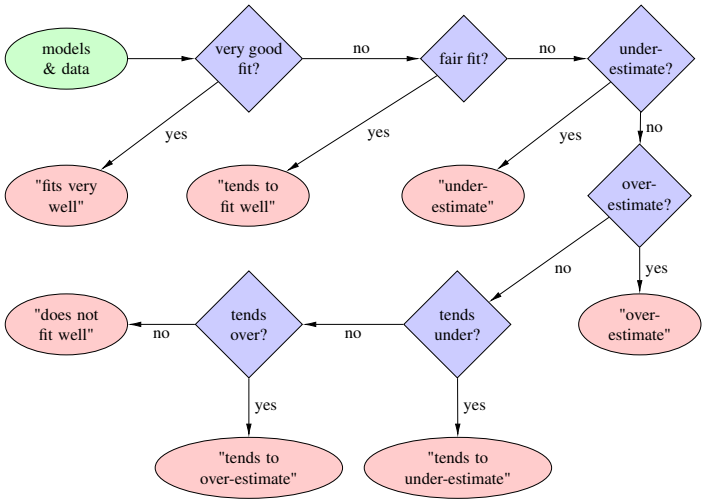


Figure 7: Flow diagram on how ESA automatically interprets the fitting results. Detailed definitions of the conditions are given in the main text.

4 Implementation

ESA is implemented in python 2.7 and calls gnuplot to generate plots. All provided gnuplot scripts are prepared for gnuplot version 5.0, but only minimal modifications, if any, will be needed to use with another gnuplot version. The online service has a clear user interface implemented with HTML/CSS, and the back-end service was realized using python CGI.

ESA was designed to work with customizable LaTeX and gnuplot templates (see Sec. 1 for details). There is a special LaTeX syntax that can be used as "variables" in the LaTeX template, which will be replaced by actual content when ESA runs. The LaTeX template can also be compiled "as-is" to make it easier for users to adapt it to their specific needs. We chose gnuplot for plot generation so that users can easily supply a template for customized figure formatting. ESA also supports user-defined models for scaling analysis. To achieve this, ESA defines functions on-the-fly from user-supplied strings.

5 Downloading and Running ESA

5.1 Downloading and Running ESA from the Command Line

ESA can be downloaded from the project page online at `www.cs.ubc.ca/labs/beta/Projects/ESA/`. After unzipping the compressed file, there will be a directory named ESA, which contains runESA.sh, the source code and other support files. A user needs to have python and gnuplot installed in order to run ESA. We used python 2.7 and gnuplot 5.0 in our environment, but expect ESA to work for other versions with minimal modifications, if any.

To run ESA from command line, a user should follow the following steps:

1. Create a directory for input and output files.
2. Put the primary input file for running time data into the directory.
3. Create files for models and LaTeX and gnuplot templates (optional; if not provided, ESA will use the default files distributed with the source code).
4. Create a configuration file named configurations.txt within the directory, telling ESA the details it requires, including names of the algorithm and the instance set/distribution, file names of running time data and, if used, the file names of model specifications and LaTeX and gnuplot templates.
5. Run the script in the ESA directory by `./runESA.sh <directory name>`, and ESA will run according to the specifications in <directory name>/configurations.txt.

5.2 Running ESA as a Web Service

ESA is also available online as a web service, which supports the essential but not all features of the command-line version. In particular:

- it only supports three predefined models:
  - exponential: $a \cdot b^n$,
  - root-exponential: $a \cdot b^{\sqrt{n}}$, and
  - polynomial: $a \cdot n^b$;
- it uses the default LaTeX and gnuplot templates for report generation;
- it only supports a limited number of statistics: median, mean, and the 75th, 90th and 95th percentiles.

To run ESA as a web service, visit `www.cs.ubc.ca/labs/beta/Projects/ESA/esa-online.html`, upload the file of running time data and fill in the other details of the form. After submission, ESA will run in the back and redirect the user to the output technical report. The user interface is shown in Figure 8.

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 *Proceedings of GECCO 2015*, pages 377–384. ACM, 2015.

[2] Holger 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.

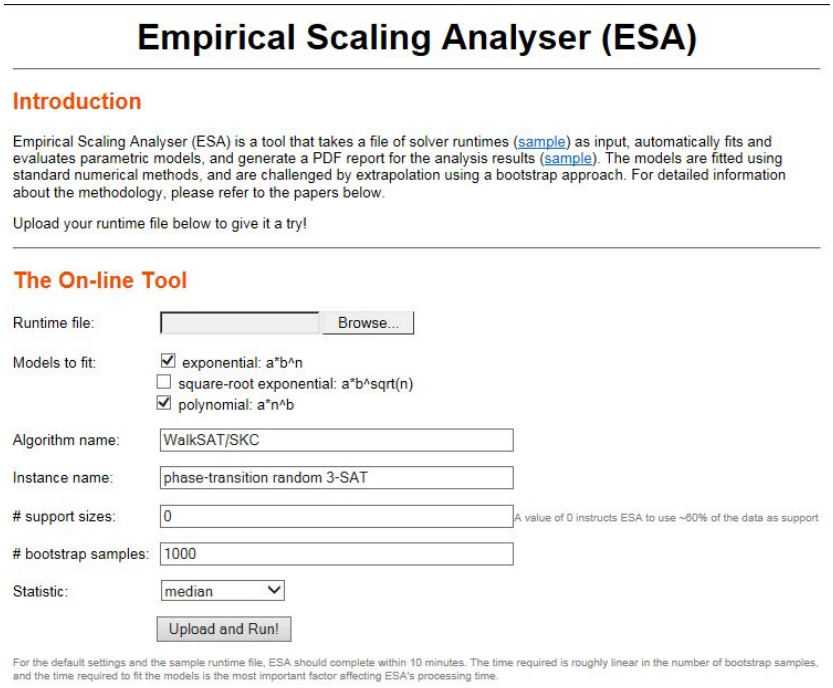


Figure 8: The user interface of ESA as a web service.

A Configuration File Options

- `fileName`
  - Default: `fileName : runtimes.csv`
  - Possible Values: `String`
  - Description: The running time data file name.
- `algName`
  - Default: `algName : Algorithm`
  - Possible Values: `String`
  - Description: The target algorithm's name. Used to populate the LaTeX template.
- `instName`
  - Default: `instName : the problem instances`
  - Possible Values: `String`
  - Description: The name of the problem instance set. Used to populate the LaTeX template.
- `modelFileName`
  - Default: `modelFileName : models.txt`
  - Possible Values: `String`
  - Description: The name of the file containing the models to be fit.
- `numTrainingData`
  - Default: `numTrainingData : 0`
  - Possible Values: `Integer`
  - Description: The number of instance sizes to be used as support sizes. If set to `0`, ESA will automatically select around 60% of the instance sizes.
- `alpha`
  - Default: `alpha : 95`
  - Possible Values: `Integer [1-100]`
  - Description: Controls the size of the bootstrap intervals.
- `numBootstrapSamples`
  - Default: `numBootstrapSamples : 100`
  - Possible Values: `Positive Integer`
  - Description: The number of bootstrap samples used by ESA. We recommend to use `500` or `1 000`.
- `statistic`
  - Default: `statistic : median`
  - Possible Values: `{median, mean, Q10, Q25, ...}`
  - Description: The running time statistic used in the primary analysis of ESA.

- `numRunsPerInstance`
  - Default: `numRunsPerInstance : 0`
  - Possible Values: `Non-negative Integer`
  - Description: Specifies the number of independent runs per instance (for randomized targer algorithms). If set to `0` ESA automatically detects the number. If positive, ESA will issue a warning if it finds one or more instances with the incorrect number of independent runs.

- `perInstanceStatistic`
  - Default: `perInstanceStatistic : median`
  - Possible Values: `{median, mean, Q10, Q25, ...}`
  - Description: The statistic used to aggregate running times across independent runs of the target algorithm on an instance, if multiple running times are provided.

- `numPerInstanceBootstrapSamples`
  - Default: `numPerInstanceBootstrapSamples : 10`
  - Possible Values: `Positive Integer`
  - Description: The number of bootstrap samples performed for independent runs on each instance.

- `logLevel`
  - Default: `logLevel : INFO`
  - Possible Values: `{ERROR, WARNING, INFO, DEBUG}`
  - Description: The log level that is printed to the console while running ESA.

- `logFile`
  - Default: `logFile : stdout`
  - Possible Values: `String`
  - Description: Can be used to specify where the ESA logging output is stored. If set to `stdout`, log messages are printed to standard out instead of a file.

- `latexTemplate`
  - Default: `latexTemplate : template-AutoScaling.tex`
  - Possible Values: `String`
  - Description: The name of the LaTeX report template.

- `modelPlotTemplate`
  - Default: `modelPlotTemplate : template-plotModels.plt`
  - Possible Values: `String`
  - Description: The name of the gnuplot template file for plotting the fitted models.

- `residuePlotTemplate`
  Default: `residuePlotTemplate : template-plotResidues.plt`
  Possible Values: `String`
  Description: The name of the gnuplot template file for plotting the residues of the fitted models.

- `gnuplotPath`
  Default: `gnuplotPath : auto`
  Possible Values: `String`
  Description: The absolute or relative path to the installation of gnuplot, e.g., `/cs/public/lib/pkg/gnuplot-5.0.3/bin/`. If set to "auto", then the location for gnuplot must be in the path variable.