On the Empirical Scaling of Run-time for Finding Optimal Solutions to the Traveling Salesman Problem

Holger H. Hoos[a,*], Thomas Stützle[b]

[a]*Computer Science Department, University of British Columbia*
2366 Main Mall, Vancouver, BC, V6T 1Z4, Canada
Phone: +1 604 822-1964, Fax: +1 604 822-5485
[b]*IRIDIA, CoDE, Université Libre de Bruxelles (ULB)*
CP 194/6, Av. F. Roosevelt 50 – B-1050 Brussels, Belgium

Abstract

The *travelling salesman problem (TSP)* is one of the most prominent NP-hard combinatorial optimisation problems. After over fifty years of intense study, the TSP continues to be of broad theoretical and practical interest. Using a novel approach to empirical scaling analysis, which in principle is applicable to solvers for many other problems, we demonstrate that some of the most widely studied types of TSP instances tend to be much easier than expected from previous theoretical and empirical results. In particular, we show that the empirical median run-time required for finding optimal solutions to so-called random uniform Euclidean (RUE) instances – one of the most widely studied classes of TSP instances – scales substantially better than $\Theta(2^n)$ with the number n of cities to be visited. The *Concorde* solver, for which we achieved this result, is the best-performing exact TSP solver we are aware of, and has been applied to a broad range of real-world problems. Furthermore, we show that even when applied to a broad range

*Corresponding author
Email addresses: hoos@cs.ubc.ca (Holger H. Hoos), stuetzle@ulb.ac.be (Thomas Stützle)

of instances from the prominent TSPLIB benchmark collection for the TSP, *Concorde* exhibits run-times that are surprisingly consistent with our empirical model of *Concorde*'s scaling behaviour on RUE instances. This result suggests that the behaviour observed for the simple random structure underlying RUE is very similar to that obtained on the structured instances arising in various applications.
Keywords: Combinatorial optimisation, travelling salesman problem, concorde, empirical scaling analysis

---

1. Introduction

The travelling salesman problem (TSP) is one of the most prominent combinatorial optimisation problems [2, 16]. It has been studied for over fifty years by mathematicians, computer scientists and researchers from various other fields, largely motivated by the fact that it is conceptually simple (and can be easily explained to anyone unfamiliar with it), yet computationally very challenging. As a result, an extraordinary amount of work has been dedicated to this problem, comprising both, theoretical analyses as well as empirical investigations [3, 5, 13, 21]. In addition, the TSP has played, and continues to play, a pivotal role in the development of algorithmic techniques for solving hard combinatorial optimisation problems.

Of the many types of TSP instances that have been studied, two-dimensional Euclidean instances represent the most commonly considered case. These instances arise in various transportation and logistics applications, as well as in the optimisation of production processes (such as drill-path optimisation in the fabrication of printed circuit boards), and can be easily visualised. TSPLIB, a collection of benchmark instances for the TSP that has been used extensively to evaluate

TSP algorithms [22], contains predominantly 2D Euclidean TSP instances, and the same holds for a more recent collection of benchmark instances maintained by William Cook [6].

Numerous theoretical results exist regarding the computational complexity of the TSP and various special cases [5, 11, 20, 24]. The general TSP is NP-hard [8], and the same holds for the special case of 2D Euclidean TSP instances [19]. However, while for the general TSP, no polynomial-time approximation algorithm exists (unless P = NP), for Euclidean distances, a polynomial-time approximation scheme is known; still, the time required for finding good solutions increases exponentially as the gap to optimality narrows (unless P = NP) [5]. Therefore, it is commonly believed that the run-time of any exact TSP algorithm scales exponentially with instance size n, even when applied to (non-trivial) 2D Euclidean TSP instances. Smith & Wormald [24] and Hwang *et al.* [11] have established a worst-case time-complexity of $O(n^{\sqrt{n}})$ for solving Euclidean TSP instances using geometric separator techniques;[1] it appears, however, that these techniques have not been exploited in any TSP solver currently available [25].

Knowledge of the empirical complexity of the TSP is considerably sparser. State-of-the-art solvers can solve many types of TSP instances with $n > 1\,000$ within hours of CPU time on commodity hardware (see, *e.g.*, [2]), but relatively little is known about the scaling of their run-time with n for interesting distributions of TSP instances. Arguably the most prominent empirical scaling analysis for the TSP is found in the recent book by Applegate *et al.* [2], who investigated the mean run-time required by Concorde, the state-of-the-art exact solver for the

[1] As clearly acknowledged by Hwang *et al.*, the complexity result for the Euclidean TSP can be traced back to earlier work by W.D. Smith.

TSP, for solving random uniform Euclidean (RUE) instances. Based on a graphical analysis of empirical mean run-times observed for Concorde on large sets of RUE instances, illustrated in Figure 16.1 of their book and reproduced in Figure 1 here, Applegate *et al.* [2] observed (p. 496):

> *The plot of mean values in Figure 16.1 indicates that the running times are increasing as an exponential function of* n*, [...]*

While the precise nature of the exponential function has not been further investigated, it is tempting to conclude that the scaling curve asymptotically approaches a straight line in the semi-logarithmic plot, indicating simple exponential scaling of the form $a \cdot b^n$, and that systematic deviations for small n may reflect the effects of preprocessing performed by the solver (Concorde carries out a limited number of iterations of the Chained Lin-Kernighan heuristic local search procedure during the initial stages of its computation).

Concorde [2, 3] is of special interest, because it is currently the best-performing exact TSP solver. For example, it has been used to solve the largest non-trivial TSP instances for which provably optimal solutions are known. (As of this writing, the largest non-trivial TSP instance for which a provably optimal solution is known is TSPLIB instance *pla85900*, a 2D Euclidean instance with $n = 85\,900$ cities derived from a real-world circuit design application – see [2, 4].) Concorde is based primarily on a complex branch & cut algorithm that uses a multitude of heuristic mechanisms to achieve good performance on a wide range of TSP instances.

Overall, we are interested in using empirical methods to characterise the computational complexity of problems and the performance achieved by state-of-the-art algorithms, with a focus on practically relevant combinatorial problems. We

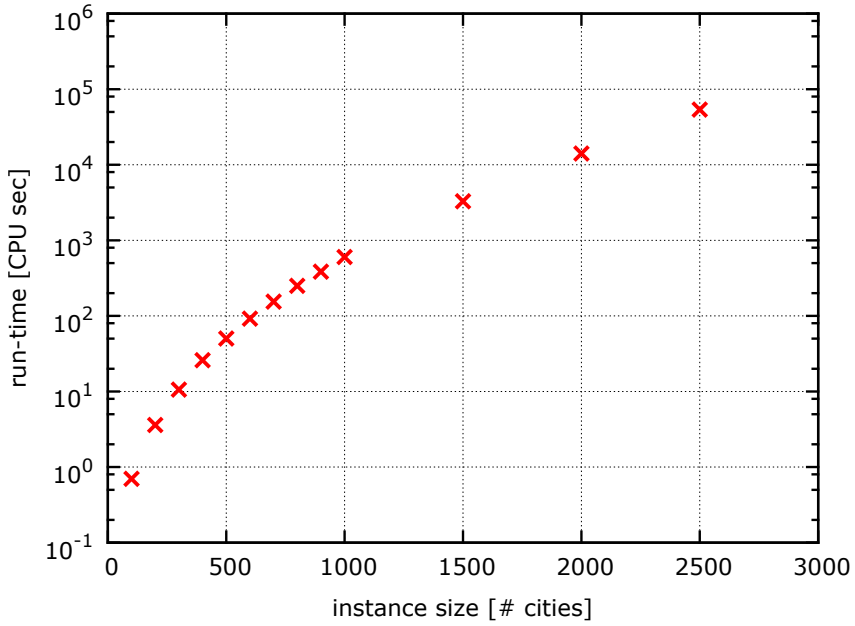

Figure 1: Scaling of mean run-time required by the Concorde solver for solving RUE TSP instances with instance size n. This plot reproduces Figure 16.1 from p. 496 of the book by Applegate *et al.* [2] and has been generated based on data from their Table 16.6; mean run-times for $n \leq 1000$ are based on 10 000 instances per instance size and those for $n > 1000$ are based on 1 000 instances per instance size.

see such methods as complementary to theoretical approaches that can yield more rigorous, general results, but are often not applicable to algorithms that show state-of-the-art performance in practice. The empirical characterisations we aim for typically take the character of models that are based on the observed behaviour of an algorithm and that are capable of producing predictions that can be critically assessed using further computational experiments. Aside from their practical utility (e.g., for assessing the suitability of a given algorithm in a particular application context or guiding algorithm development), such models can in principle also

inform theoretical research [see, *e.g.*, 12, 23].

In this work, we chose to study the empirical complexity of the 2D Euclidean TSP, primarily because of its status as a particularly prominent and well-studied NP-hard computational problem with a clearly established state-of-the-art solver of considerable practical importance, Concorde. In particular, we were interested in building parametric models for the scaling of Concorde's run-time with instance size. We developed and thoroughly tested such models for the widely studied class of RUE instances and subsequently evaluated them on a variety of benchmark instances from TSPLIB. In doing so, we originally intended to make more precise the claim by Applegate *et al.* [2], and to test to which extent it applies to more structured TSPLIB instances.

To our surprise, we found that exponential scaling models of the form $a \cdot b^n$ for Concorde's mean and median run-time over sets of RUE instances are not significantly more precise than a polynomial model (of degree about 3.78). A scaling model of the form $a \cdot b^{\sqrt{n}}$, on the other hand, turned out to be surprisingly accurate, even when evaluated on instance sizes up to three times larger than those used for fitting its parameters. (To the best of our knowledge, this is the first time that such a root-exponential scaling model has been used to model the run-time of a state-of-the-art algorithm for TSP or any other NP-hard problem.) Furthermore, we found that Concorde's run-time on RUE instances of a given size n appears to be log-normally distributed with a coefficient of variation (standard deviation over mean) that is independent of n, which implies that even high quantiles of these distributions show root-exponential scaling. Finally, we observed that the simple scaling model for Concorde's run-time thus obtained agrees surprisingly well with the run-times observed on more structured instances from TSPLIB.

The remainder of this article is structured as follows. In Section 2, we give technical details on the computational experiments that form the core of our study and describe the method we developed for conducting empirical scaling analyses in a statistically rigorous way. Section 3 presents the results from these experiments, and Section 4 provides additional discussion of those findings. Finally, in Section 5, we draw some general conclusions from our results and outline several directions for further investigation.

2. Materials and Methods

TSP solver. The algorithm we study in this work is Applegate *et al.*'s well-known branch & cut solver, Concorde, version Concorde-03.12.19 [3], the best performing exact algorithm for the TSP available today. Concorde makes use of an external linear-programming (LP) solver, and in the experiments reported here, we used it in conjunction with QSopt 1.01, an LP solver specifically designed to be used with Concorde. All parameters were left at their default settings, and the random number seed was always set to 23.

TSP instance sets. We performed experiments on two widely studied classes of 2D Euclidean TSP instances: RUE instances, and instances from TSPLIB. RUE instances are generated by placing a number of points uniformly at random within a unit square; those points represent the cities to be visited in the corresponding TSP instance, and the distances between those are simply determined as the Euclidean distances between the respective points. For each instance size n, this random generation method induces a probability distribution over RUE instances. The RUE instances used in our study were generated using the `portgen` generator from the 8th DIMACS Implementation Challenge.

For $n = 500, 600, \ldots, 2000$, we generated 1000 instances and for $n = 2500$, 3000, $3500, 4000$ and 4500, 100 instances per instance size. Smaller instances are too easy for Concorde and result in floor effects, *i.e.*, inaccurate CPU time measurements due to limited resolution of commonly used methods for process timing and distortions in scaling behaviour (as running time becomes dominated by Concorde's preprocessing stages); larger instances require infeasibly long runs of Concorde. In addition, we have selected all instances from the TSPLIB web site [22] of 500 to 4500 cities that had edge types EUC 2D, CEIL 2D and ATT (these are all derived from 2D Euclidean distances); this resulted in a set of 29 instances.

Execution environment. All runs of Concorde were performed on a cluster of identical machines, each equipped with two dual-core 2.4 GHz AMD Opteron 2216 processors with 2×1MB L2 cache and 4GB main memory, running Cluster Rocks Linux version 4.2.1/CentOS 4. The programme was compiled with gcc-3.4.6-3, and only one CPU core was used for each run. The CPU time required by solving a given TSP instance with Concorde was directly taken from Concorde's standard output. The run-time measurements thus obtained for individual instances were collected for each set of instances of a given size n, and descriptive statistics of the distribution of run-times over the instances in each set were computed from these data. Unless explicitly stated otherwise, we ensured that Concorde runs on all instances of a given set completed (*i.e.*, for each instance, an optimal solution was found and proven to be optimal).

Scaling analysis. Given instance sizes $n_1, n_2 \ldots, n_k = 500, 600, \ldots, 1500$ and $m = 1000$ RUE instances per size, we used the following bootstrapping procedure to assess the predictions made by an empirical scaling model, M: For

each instance size n_i, we independently drew $r = 1000$ samples $I_{i,1}, \ldots, I_{i,r}$, where each $I_{i,j}$ consists of m instances and is determined by independent uniform random sampling *with replacement* from the full set of size n_i instances. Then, for each of the 1000 series of instance sets $I_{1,j}, I_{2,j}, \ldots, I_{k,j}$, $j = 1, \ldots, 1000$, we fitted a parametric scaling model M_j to the corresponding observed run-times $B(I_{1,j})$, using the nonlinear least-squares Marquardt-Levenberg algorithm as implemented in the widely used *Gnuplot* software. Next, for a given instance size n, we used models M_j to obtain a set of predictions $P := \{P_1(n), \ldots, P_r(n)\}$, where $P_j(n)$ is the prediction obtained from model M_j. From the set of predictions P, we determined the *bootstrap percentile confidence interval* for a given confidence level α as $CI := [q_{(0.5-\alpha/2)}, q_{(0.5+\alpha/2)}]$ [see, *e.g.*, 7], where $q_{(x)}$ denotes the x-quantile of the empirical distribution of the values in P; we used a standard value of $\alpha = 0.95$. Confidence intervals on the model parameters were obtained analogously from the sets of parameter values for the models M_j. Finally, we compared the actual run-times observed for the original set of instances for size n against the confidence intervals thus obtained. To the best of our knowledge, this is the first use of this type of bootstrap analysis, which has been developed by Hoos [9] for this purpose, for characterising the empirical behaviour of an algorithm for any NP-hard problem.

3. Results

We started our scaling analysis by a visual inspection of the mean and median run-time of Concorde for the RUE instance sets with $n = 500, \ldots, 1500$. As can be seen in Figure 2 (top pane), contrary to what could be expected based on the results from the book by Applegate *et al.* [2], the fact that those scaling curves

appear as straight lines in a log-log plot suggests polynomial scaling behaviour. The same observation was made for the scaling of the 0.1- and 0.9-quantiles of the distribution of Concorde's run-time over the sets of instances for each n. Another surprising observation is that the slope of the scaling lines for the mean and the three quantiles considered here appears to be the same, which indicates that, regardless of instances size, those statistics differ only in a constant factor.

We then fitted three parametric scaling models to the median run-time data for $n = 500, \ldots, 1500$. We chose the median for this and many of the subsequent analyses, because medians generally tend to be statistically more stable than means and, more importantly in our context, can be estimated reliably in the presence of censored data: here, runs of Concorde that have timed out after some large cut-off time without solving a given instance. We considered the following three scaling models:

- a 2-parameter exponential model: $Exp[a, b](n) := a \cdot b^n$
- a 2-parameter polynomial model: $Poly[a, b](n) := a \cdot n^b$
- a 2-parameter *square-root-exponential*, or short, *root-exponential* model:[2] $RExp[a, b](n) := a \cdot b^{\sqrt{n}}$

The first model was chosen based on the results of Applegate *et al.* [3], the second based on our earlier observation, and the third was chosen based on a preliminary

[2]In computing science, this scaling is sometimes referred to as *sub-exponential* (see, *e.g.*, [15]); more precisely, is has occasionally also been called *weakly* or *mildly exponential*. In physics, the term *stretched exponential* is used to refer to a very similar type of function. We chose the term *root-exponential*, which is also commonly used for this type of function in physics and geography, because it characterises our model more precisely.

manual assessment of several models that would produce scaling behaviour in-between the first two models. We note that all three models have two degrees of freedom, and we generally fit these models on at least 11 data points (obtained from a total of at least $11\,000$ independent runs of Concorde); we therefore see no risk of unfair comparisons between models and only minimal risk of overfitting the data.

As can be seen in the bottom pane of Figure 2, while both, the exponential and polynomial models approximate the data they were fitted on quite well (RMSE = 12.81 and 15.38, respectively), the root-exponential model produces a much better fit (RMSE = 8.83), especially for small instance sizes. When we challenged these models by using them to predict median run-times for $n = 1600, 1700 \ldots, 2000$ and $n = 2500, 3000, \ldots, 4500$, we found that, while the exponential and polynomial models are similarly inaccurate, the predictions made by the root-exponential model match the observed median run-times for these instance sets very accurately. Table 1 further illustrates these results. The exponential model severely overpredicts median run-times (up to a factor of 30 for $n = 4500$), while the polynomial model consistently underpredicts the actual median run-times. The root-exponential model, in contrast, is always within a factor of slightly below 1.6 of the observed median run-times, and there is no evidence that its accuracy deteriorates with increasing instance size.

We note that, although we allowed very long run-times of Concorde on each instance, for instance sizes $n = 4000$ and $n = 4500$, not all instances could be solved. However, because in both cases we managed to solve substantially more than 50% of the instances in these sets, we were able to correctly estimate median run-times.

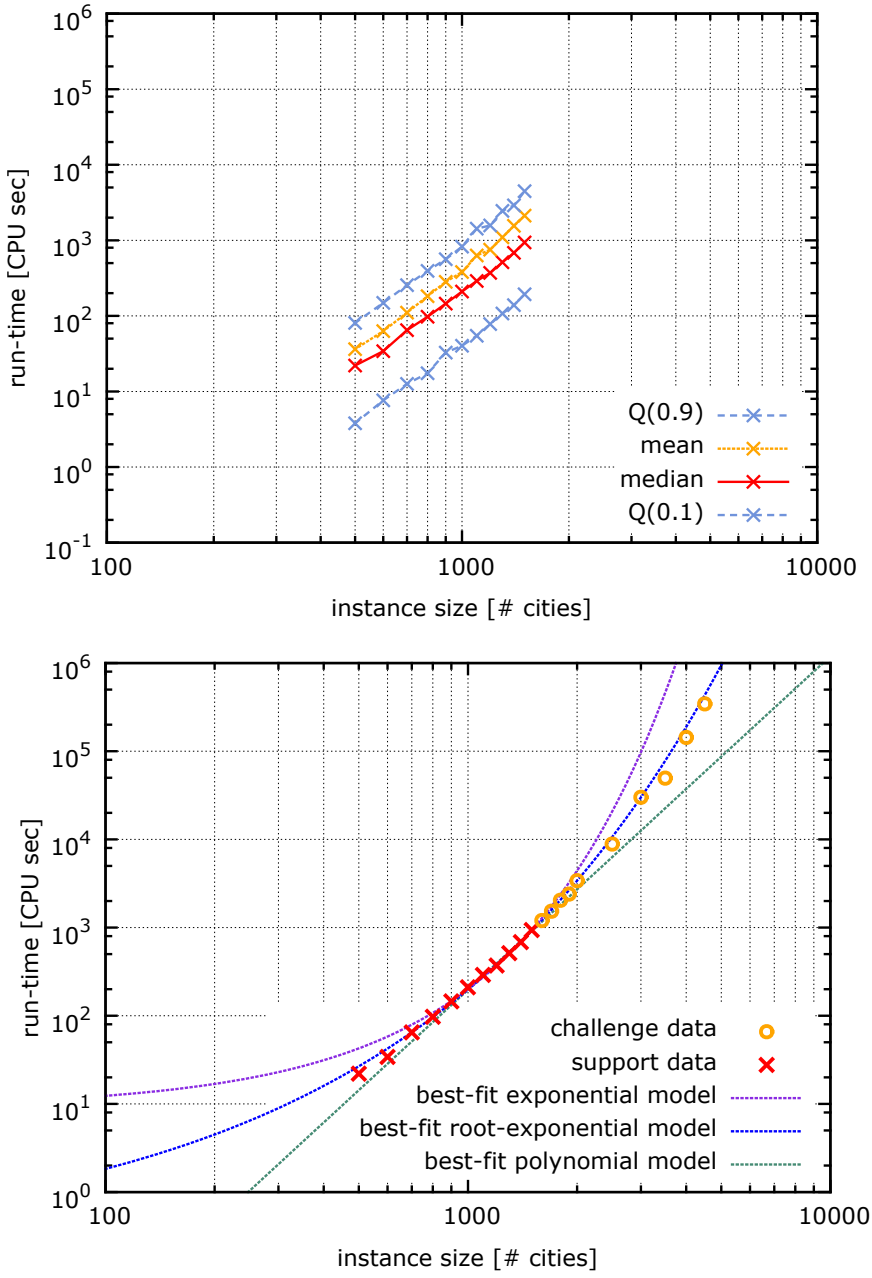

Figure 2: Scaling of run-time required by the Concorde solver for finding optimal solutions to RUE TSP instances and proving their optimality. The best-fit exponential model is $8.90 \cdot 1.00311^{n}$, the root-exponential model is $0.21 \cdot 1.24194^{\sqrt{n}}$ and the polynomial model is $8.83 \cdot 10^{-10} \cdot n^{3.78}$. The run-time statistics for each instance size used for the curves on the top pane and for the fits are based on 1000 randomly generated instances; the data points used in the bottom pane to illustrate the predictive accuracy of the three scaling models are based on 1000 instances up to $n = 2000$ and on 100 instances for larger n.

	predicted median run-time			
instance size	polynomial model	exponential model	root-exponential model	observed median run-time
2 000	2 709.80	4 446.53	3 393.90	3 400.82 (1000/1000)
2 500	6 302.61	21 020.46	10 651.96	8 855.28 (100/100)
3 000	12 561.47	99 371.86	29 957.63	30 024.49 (99/100)
3 500	22 505.10	469 769.31	77 531.36	49 476.78 (100/100)
4 000	37 294.77	2 220 781.76	187 871.30	143 334.46 (90/100)
4 500	58 229.53	10 498 496.81	431 400.74	344 131.05 (65/100)

Table 1: Predicted median run-times from polynomial and exponential fits for Concorde on RUE instances (finding optimal solutions and proving optimality) *vs* observed median run-times; the instance sizes shown here are larger than those used for fitting the models. The numbers in the last column shown in parentheses indicate the fraction of instances within each set that were solved by Concorde; the remaining instances could not be solved within time much larger than that of the longest completed run on the same set, but have no impact on median run-time.

To further validate our scaling results, we computed bootstrap confidence intervals for the predictions. Table 2 shows that the observed run-times for instance sizes $n \geq 2000$ are not consistent with confidence intervals for the predictions made by the exponential and the polynomial models. The accuracy of the root-exponential model, on the other hand, is confirmed by the results from this analysis: For all instance sizes except $n = 3500$, the observed run-times fall within the confidence intervals obtained for the respective predictions, and for $n = 3500$, the observed run-time is less than 5% below the lower limit of the confidence interval.

Next, we investigated in more detail the distribution of Concorde's run-time over RUE instances of the same size n. Figure 3 shows the empirical cumulative distribution functions for the observed run-times on instances of size $n = 500, 1000$ and 1500. We refer to these distributions as *solution cost distributions* (*SCDs*). Examining these empirical SCD curves, it appears that, while

	predicted median run-time			
instance size	polynomial model	exponential model	root-exponential model	observed median run-time
2 000	[2 298.22 , 3 160.39]	[3 793.00 , 5 266.68]	**[2 854.21 , 3 977.55]**	3 400.82 (1000/1000)
2 500	[4 987.78 , 7 870.81]	[16 378.03 , 28 010.39]	**[8 266.46 , 13 601.02]**	8 855.28 (100/100)
3 000	[9 430.35 , 16 615.93]	[70 584.38 , 147 716.74]	**[21 549.28 , 41 271.35]**	30 024.49 (99/100)
3 500	[16 087.86 , 31 246.26]	[304 354.12 , 780 313.20]	[51 883.73 , 114 841.57]	49 476.78 (100/100)
4 000	[25 552.92 , 54 042.99]	[1 305 365.53 , 4 110 129.86]	**[117 547.77 , 298 973.70]**	143 334.46 (90/100)
4 500	[38 431.20 , 87 841.09]	[5 616 741.54 , 21 733 073.57]	**[253 401.82 , 734 363.20]**	344 131.05 (65/100)

Table 2: Bootstrap confidence intervals for $\alpha = 0.95$ for median run-time predictions for Concorde on RUE instances (finding optimal solutions and proving optimality); the instance sizes shown here are larger than those used for fitting the models. The $\alpha = 0.95$ bootrap confidence intervals for the model parameters are $[8.69 \cdot 10^{-11}, 8.44 \cdot 10^{-9}], [3.46, 4.11]$ for the polynomial model, $[6.71, 11.21], [1.00292, 1.00334]$ for the exponential model and $[0.115, 0.373], [1.2212, 1.2630]$ for the root-exponential model.

there is substantial variation of run-time for each given instance size (the ratio between the 0.9- and 0.1-quantiles is roughly 20), there is no significant change in the shape of the distributions (when run-time is plotted on a log-axis) as n increases. Consistent with our earlier observation, this suggests that the quantiles of these distributions are related to each other by a factor that does not depend on n, and therefore show the same scaling behaviour as the median.

Further examination suggested that these empirical SCDs closely resemble log-normal distributions. In fact, at the $\alpha = 0.05$ significance level, the Shapiro-Wilk test of normality on the log-transformed run-times fails to reject the hypothesis that the log-transformed run-time data were normally distributed for $n = 1000$ and $n = 1500$ (for $n = 500$, the hypothesis was rejected, mostly due to a decay in the tails of the distributions that was faster than that of a best-fit log-normal distribution). The same test on the instance sets for $n = 2000$ and 2500 could also

not be rejected and a visual inspection of the truncated SCDs for $n = 3500, 4000$ and 4500 (where the truncation results from the fact that exact run-times for some instances could not be measured, as previously explained) further supported our hypothesis that Concorde's run-times for sufficiently large instance sizes are log-normally distributed.

Furthermore, when examining various quantile ratios and the coefficient of variation of the empirical SCDs, we found no evidence that would suggest an increase in variance of the log-transformed data with n. This supports our earlier observation that on RUE instances, not only does Concorde's median run-time scale according to our 2-parameter square-root-exponential model with n, but the same holds for all quantiles of the distributions of Concorde's run-time over instances of size n, as well as for mean run-time.

When re-examining the data on the scaling of Concorde's mean run-time on sets of RUE instances from the book by Applegate *et al.* [2] (Table 16.6 on p. 496), we obtained results that are fully consistent with those from our previous analysis: fitting our three parametric models to their data produced clear indication that the best-fit root-exponential model fits the observed mean run-times much better (RMSE = 61.43) than either, the exponential and polynomial models (RMSE = 214.80 and 246.83, respectively). Furthermore, when fitting our models to the data for $n = 500, \ldots, 1500$ only, we obtained rather accurate predictions for $n = 2000$ and 2500 for the root-exponential model, while the polynomial model, and even more so the exponential model, give substantially poorer predictions.

Finally, we compared the scaling of Concorde's observed run-times on the earlier mentioned TSPLIB instances to those of the RUE instances. We found that Concorde could not solve instances *u1817* and *d2103* within 3 CPU days. For

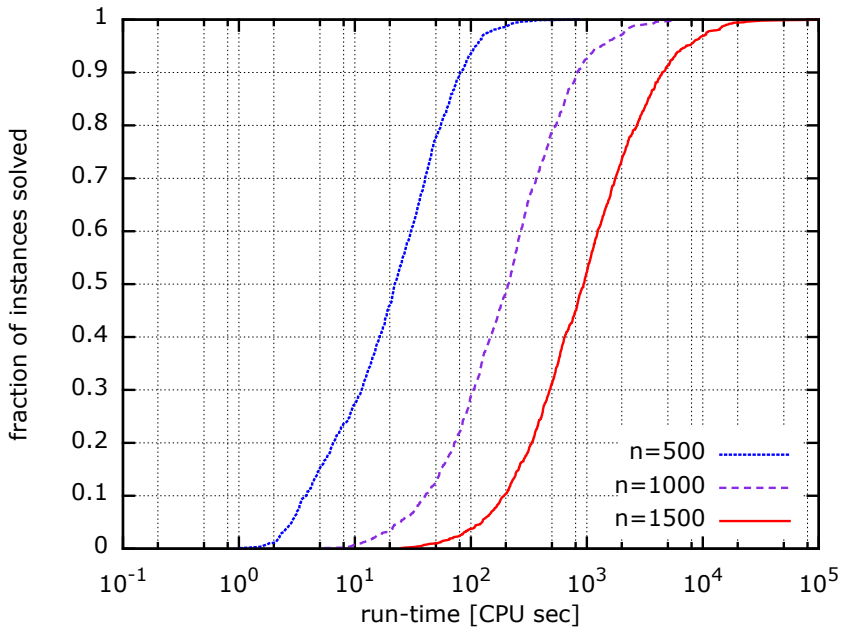

Figure 3: Distributions of run-time required by the Concorde solver for finding an optimal solution to a given TSP instance and proving its optimality over three sets of RUE instances (1000 instances per set). For $n = 1000, 1500$ (as well as for $n = 2000, 2500$), the log-transformed data pass the Shapiro-Wilk normality test for $\alpha = 0.05$.

the remaining 27 instances, we compared Concorde's observed run-times with the run-times predicted by our best-fit root-exponential scaling model. We also considered predictions obtained from our scaling model under the additional assumption that the 0.05- and 0.95-quantiles of Concorde's run-time for each n are located at constant factors from the median (we estimated this factor as 9.27 from the data obvserved for $n = 2000$, the largest set of 1000 instances used in our study). As can be seen in Figure 4, of the 29 TSPLIB instances considered, 21 fell within the band between our predictions for the 0.05- and 0.95-quantiles of Concorde's run-time, while 5 were higher than the 0.95-quantiles (instances *fl1400,*

d1291, u2319 in addition to *d2103* and *u1817*), and 3 were lower than the 0.05-quantile (instances *pr1002, pr2392, d1655*). It is remarkable that the predictions based on our model are accurate for 72% of the instances from TSPLIB, especially when considering that the model is solely based on Concorde's behaviour on RUE instances, which differ markedly from the Euclidean instances in TSPLIB in terms of the distribution of the cities to be visited. [3]

We performed an analogous analysis for the National instances with 500 to 5000 nodes and VLSI instances with 500 to 4500 nodes from the TSP webpage at `http://www.math.uwaterloo.ca/tsp/index.html`. The National instances are based on the locations of cities within individual countries, while the VLSI instances are from applications in VLSI circuit design. While all seven National instances in the indicated size range fell within the 0.05- and 0.95-quantiles of our predictions for Concorde's run-time, this was not the case for the VLSI instances, most of which turned out to be harder to solve than RUE instances of similar size (see Figure 5). Of the 54 VLSI instances in the indicated size range, Concorde could not solve 31 within 7 CPU days, indicating that our predictions are unreliable in this case. However, it seems that these instances are particularly hard for Concorde, as for many of the instances we could not solve, no provably optimal solution is currently known.[4]

[3] Independently from our work, David Applegate has conducted a similar investigation, based on RUE instances up to size $n = 2500$, using the scaling model $a \cdot b^{\sqrt[4]{n}}$ [1]. When fitting and assessing this model on our data, we found that for $n \geq 3000$, it usually underestimates observed median running times, and that for $n = 3000, 4000$ and 4500, those running times fell outside of the confidence intervals for the predictions obtained from this model.

[4] Note that these VLSI instances have a significant number of same-length edges and thus show edge-length distributions markedly different from those of RUE and many other types of Euclidean

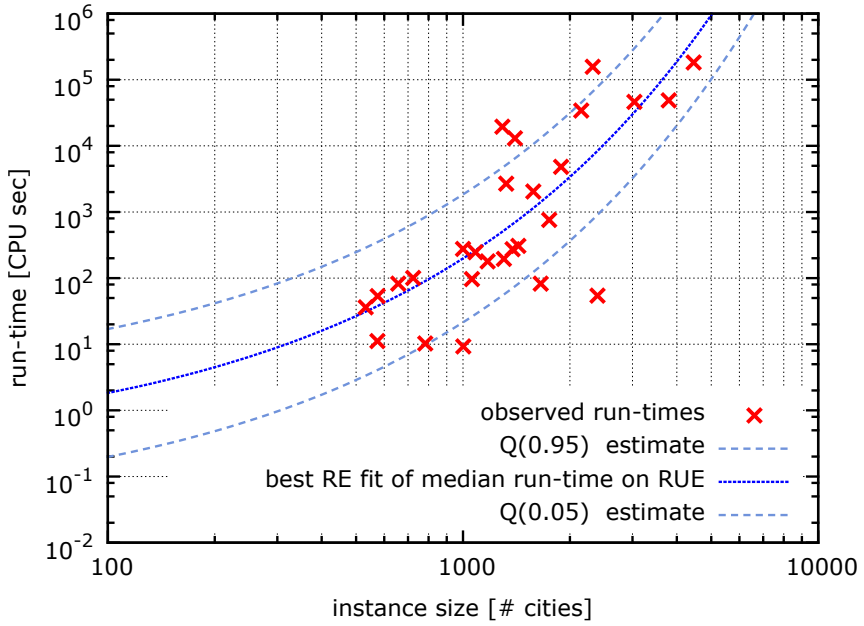

Figure 4: Run-times required by the Concorde solver for solving TSPLIB instances with size $n = 500 \ldots 4500$. The dashed lines represent the best-fit root-exponential scaling determined on RUE instances and estimates of the 0.05- and 0.95-quantiles of the distributions of run-times over sets of RUE instances of a given size. Two TSPLIB instances that also fall within the range of sizes considered here could not be solved by Concorde within 7 CPU days, and are therefore not shown in this plot.

4. Discussion

Considering widely known complexity results for the TSP and Euclidean TSP instances and the fact that current TSP solvers do not exploit geometric separa-

TSP instances. Interestingly, the TSPLIB instances that fall outside the 0.05- and 0.95-quantiles of our run-time predictions are instances from the drilling of PCB boards that show node distributions similar to those encountered in the VLSI instances.

tor techniques, we were surprised to find evidence for root-exponential scaling of Concorde's run-time with instance size on a class of widely used benchmark instances, such as RUE. We note that even with the worst-case time-complexity of $O(n^{\sqrt{n}})$ result of Smith & Wormald [24] and Hwang *et al.* [11] in mind, there remains a difference corresponding to a log factor in the exponent. In fact, we also fitted a model of the form $a \cdot b^{\sqrt{n} \cdot \ln n}$; interestingly, when checking the median run-time predictions for large instances from this model fitted on instances for n in 500 to 1500, we found that it usually overestimates the observed median run-times and that – except for instance sizes 2000 and 3000 – those observed run-times fell outside the confidence intervals of the bootstrap confidence intervals for the predictions obtained from this model.

Still, we see no contradiction between our empirical result and theoretical worst-case complexity results. Firstly, RUE instances are not truly Euclidean, since the distances are rounded to the nearest integers; therefore, theoretical results for Euclidean instances do not strictly apply. Secondly, it is entirely possible that a family of 2D Euclidean instances exists for which Concorde shows exponential scaling of the form $O(2^n)$ or $O(n^{\sqrt{n}})$. Our results suggest, however, that if such instances existed within the RUE distribution, they would become increasingly rare as n grows,[5] and that, furthermore, such instances do not appear to have structure that is commonly encountered, *e.g.*, in TSPLIB.

When considering known results on Concorde's run-time on larger, real-world Euclidean TSP instances, we obtained further evidence for its root-exponential scaling over an exponential scaling. The results for various large Euclidean TSP

[5]This follows from the scaling of median run-time and our observation that run-times over instance sets with fixed n are log-normally distributed.

instances published on the Concorde website [3] in many cases have been obtained using different versions of Concorde, non-default parameter setting, a different linear programming solver (we used QSopt 1.01 for our experiments) and an execution environment different from that considered in our study. Therefore, these cannot be directly compared against the predictions made by our models as was done for the TSPLIB instances considered in Section 3. However, we note that on large TSPLIB instances the run-times reported by Cook *et al.* on the Concorde website are many order of magnitudes below those predicted by the exponential model. For example, the solution of *d15112* involved running Concorde for about 22.6 CPU years, while the prediction of our best-fit exponential model exceeds 10^{13} CPU years. In contrast, our best-fit polynomial model predicts a median run-time of slightly over 2 CPU months, while the best-fit root-exponential model yields a prediction of slightly below 2500 CPU years. Similarly, the solution of instance *pla85900* took about 136 CPU years,[6] which can be contrasted with predictions of over 10^{109}, about 125 and over 10^{19} CPU years obtained from our exponential, polynomial and root-exponential models, respectively.

While, for the reasons mentioned earlier, these observations should not be seen as direct support for our model of Concorde's scaling behaviour, they certainly appear to be substantially more consistent with it than with the hypothesis of exponential scaling of the form $a \cdot b^n$. They are also consistent with our hypothesis that realistically structured 2D Euclidean TSP instances are not harder to solve than RUE instances – a hypothesis that parallels earlier findings on similarities in the gap between optimal tour length and the Held-Karp bound [14].

[6] see `http://www.tsp.gatech.edu/pla85900/compute/cpu.htm`; scaled to our execution environment, this figure could be up to a factor of about two lower.

Looking at the published run-time results for large TSPLIB instances, it seems likely that large real-world instances are substantially *easier* for Concorde than most similarly sized RUE instanced, but of course, this might be primarily due to the differences in the way Concorde was used when solving instances such as *pla85900*. In fact, the differences in run-time between our results for TSPLIB instance and those reported in the book by Applegate *et al.* [2] (see their Table 16.8 on p. 503 *vs* our Figure 4) are too large and too inconsistent between problem instances to be solely caused by differences in the respective execution environments; for example, on instance *u2152* we measured a run-time of 34 209.43 CPU seconds, while Applegate *et al.* [2] report 3 345.3 CPU seconds (also measured on 2.4 GHz AMD Opteron CPUs, thus using similar hardware as ours). This suggests that the LP solver used in the respective experiments, QSopt 1.01 in ours and ILOG CPLEX 6.5 in theirs, has a substantial impact on the performance of Concorde (which is consistent with earlier findings by Mittelmann for CPLEX 12.3 [18]). Nevertheless, the close resemblance between the parameters of the scaling models we derived from the RUE data by Applegate *et al.* [2] (which were measured using the CPLEX LP solver) and those fitted on our RUE data (which are based on QSopt) suggests that at least the choice between those two LP solvers has little or no impact on the scaling behaviour of Concorde.

Finally, it is interesting to contrast the methods introduced here with those used in other work on run-time prediction. There is, in fact, a growing body of literature on predicting the run-time of high-performance algorithms for solving difficult combinatorial problems, such as the TSP, based on instance features and, in some case, also algorithm parameters (see, *e.g.*, [10, 17]). These approaches generally use machine learning techniques to construct complex statistical models

of algorithm performance based on large sets of instance features; these models are then used to predict performance on instances similar to those used for model fitting. In contrast, the approach taken in our work here is focused on scaling of performance with instances size, and specifically, with the ability of scaling models to produce accurate performance predictions for instance sizes much larger than those used for model fitting. Furthermore, unlike other work on run-time prediction, our study uses rigorous statistical techniques to assess the validity of predictive models for the scaling of run-time.

5. Conclusions and Future Work

The empirical analysis presented in this work demonstrates that the state-of-the-art exact TSP solver *Concorde* [2] shows scaling of run-time with instance size n of the form $a \cdot b^{\sqrt{n}}$ for the widely studied class of uniform random Euclidean (RUE) instances; this is the case for the median run-time (with constants $a \approx 0.21, b \approx 1.24$), and also appears to hold for the mean and all quantiles of the distribution of run-time over instances of a given size n (although the evidence for means and high quantiles is weaker, due to unsolved instances for $n \geq 3\,000$). Interestingly, the scaling model for Concorde's run-time thus obtained is in good agreement with the run-times measured for Euclidean instances from TSPLIB, whose structure differs in many cases substantially from that of RUE instances. However, there are also classes of Euclidean TSP instances, such as the VLSI instances from the TSP webpage at `http://www.math.uwaterloo.ca/tsp/index.html`, for which Concorde shows significantly different scaling behaviour; we believe that this is caused by distributions of cities that give rise to many precisely identical edge lengths.

From the methodological perspective, we believe that our work demonstrates (i) that the root-exponential scaling model could serve as a standard alternative hypothesis to simple exponential scaling and (ii) that the bootstrap approach used here provides a general and statistically sound way for assessing the empirical scaling behaviour for practically relevant algorithms.

Considering the more theoretical side, it would be intriguing to explain analytically the scaling behaviour we have observed here. Due to the many heuristic components that are used in the actual codes, it seems unlikely that anyone would be able to produce such results for a state-of-the-art algorithm such as Concorde. However, it might be possible to prove at least the same asymptotic scaling (with much larger constants) for more abstract and simpler exact TSP algorithm that are more amenable to theoretical analysis. Previously mentioned work by Smith & Wormald [24] could provide a good starting point for such an investigation; it may also provide a route towards building a high-performance solver for 2D Euclidean TSP instances that has theoretical worst-case time complexity $O(n^{\sqrt{n}})$.

Overall, we feel that the results presented in this work challenge at least two common beliefs: (1) that the run-time for a state-of-the-art exact solver for an NP-hard optimisation problem such as the TSP should be expected to scale exponentially, and (2) that empirical complexity results regarding the behaviour of a state-of-the-art solver on a class of rather simplistic instances say little or nothing about the empirical complexity of solving a diverse range of structured benchmark instances. Of course, the way in which our findings contradict both of these intuitions might be quite specific to Euclidean TSP instances and to Concorde. Even so, considering the status of Concorde as the long-standing champion of among exact TSP solvers as well as the prominence of Euclidean TSP instances,

we believe them to be of broad interest and expect them to stimulate a substantial amount of further research on the TSP and other hard combinatorial problems.

Acknowledgments

We thank Jonathan L. Shapiro (University of Manchester) for useful discussion of ideas underlying the bootstrapping method used for assessing the scaling models during the 2008 Dagstuhl Seminar on Theory of Evolutionary Algorithms. We also thank William Cook and Keld Helsgaun for providing us with valuable information regarding the solution of large TSPLIB instances, David Applegate for technical details on his scaling analysis, and Warren D. Smith for information on his complexity result. Finally, we gratefully acknowledge useful comments on this work from David Johnson and David Applegate, as well as helpful comments from the anonymous reviewers of an earlier version of this article. Holger Hoos acknowledges support by NSERC. Thomas Stützle acknowledges support of the F.R.S.-FNRS of which he is a research associate.

References

[1] D. Applegate. Personal Communication, July 2012.

[2] D. Applegate, R. Bixby, V. Chvatal, and W. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2006.

[3] D. Applegate, R. Bixby, V. Chvátal, and W. Cook. The traveling salesman problem, concorde TSP solver. `http://www.tsp.gatech.edu//concorde/index.html`, 2012. Version visited last on 15 June 2012.

[4] D. Applegate, R. E. Bixby, V. Chvátal, W. Cook, D. Espinoza, M. Goycoolea, and K. Helsgaun. Certification of an optimal TSP tour through 85,900 cities. *Operations Research Letters*, 37(1):11–15, 2009.

[5] S. Arora. Polynomial time approximation schemes for Euclidean traveling

salesman and other geometric problems. *Journal of the ACM*, 45(5):753–782, 1998.

[6] W. Cook. The traveling salesman problem, TSP test data. `http://www.tsp.gatech.edu/data/index.html`, 2012. Version visited last on 15 June 2012.

[7] B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall, New York, NY, 1993.

[8] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of $\mathcal{NP}$-Completeness*. Freeman, San Francisco, CA, 1979.

[9] H. H. Hoos. A bootstrap approach to analysing the scaling of empirical run-time data with problem size. Technical Report TR-2009-16, University of British Columbia, June 2009. Available on-line at http://www.cs.ubc.ca/ hoos/Publ/Hoos09.pdf.

[10] F. Hutter, L. Xu, H. Hoos, and K. Leyton-Brown. Algorithm runtime prediction: Methods & evaluation. *Artificial Intelligence*, 206:79–111, 2014.

[11] R. Hwang, R. Chang, and R. Lee. The searching over separators strategy to solve some NP-hard problems in subexponential time. *Algorithmica*, 9:398–423, 1993.

[12] D. S. Johnson. A theoretician's guide to the experimental analysis of algorithms. In M. H. Goldwasser, D. S. Johnson, and C. C. McGeoch, editors, *Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implementation Challenges*, pages 215–250. AMS, Providence, RI, 2002.

[13] D. S. Johnson and L. A. McGeoch. Experimental analysis of heuristics for the STSP. In G. Gutin and A. Punnen, editors, *The Traveling Salesman*

Problem and its Variations, pages 369–443. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2002.

[14] D. S. Johnson, L. A. McGeoch, and E. E. Rothberg. Asymptotic experimental analysis for the Held-Karp traveling salesman bound. In *Proceedings of the seventh annual ACM-SIAM symposium on Discrete algorithms*, SODA '96, pages 341–350, Philadelphia, PA, USA, 1996. Society for Industrial and Applied Mathematics.

[15] G. Kuperberg. A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. *SIAM Journal on Computing*, 35(1):170–188, 2005.

[16] E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, and D. B. Shmoys. *The Traveling Salesman Problem*. John Wiley & Sons, Chichester, UK, 1985.

[17] O. Mersmann, B. Bischl, H. Trautmann, M. Wagner, J. Bossek, and F. Neumann. A novel feature-based approach to characterize algorithm performance for the traveling salesperson problem. *Annals of Mathematics and Artificial Intelligence*, 69:151–182, 2013.

[18] H. D. Mittelmann. Concorde-TSP benchmark: Concorde with different LP solvers. `http://plato.asu.edu/ftp/tsp_bench.html`, 2012. Version visited last on 15 June 2012.

[19] C. H. Papadimitriou. The Euclidean travelling salesman problem is np-complete. *Theoretical Computer Science*, 4(3):237–244, 1977.

[20] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization – Algorithms and Complexity*. Prentice Hall, Englewood Cliffs, NJ, USA, 1982.

[21] G. Reinelt. *The Traveling Salesman: Computational Solutions for TSP Applications*, volume 840 of *LNCS*. Springer Verlag, Berlin, Germany, 1994.

[22] G. Reinelt. TSPLIB. `http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95`, 2012. Version visited last on 15 June 2012.

[23] P. Sanders and R. Fleischer. Asymptotic complexity from experiments? A case study for randomized algorithms. In S. Näher and D. Wagner, editors, *WAE'00*, volume 1330 of *LNCS*, pages 135–146, Heidelberg, Germany, 2001. Springer Verlag.

[24] W. Smith and N. Wormald. Geometric separator theorems & applications. In *FOCS'98*, pages 232–243. IEEE Computer Society, Los Alamitos, CA, 1998.

[25] W. D. Smith. Personal Communication, June 2009.

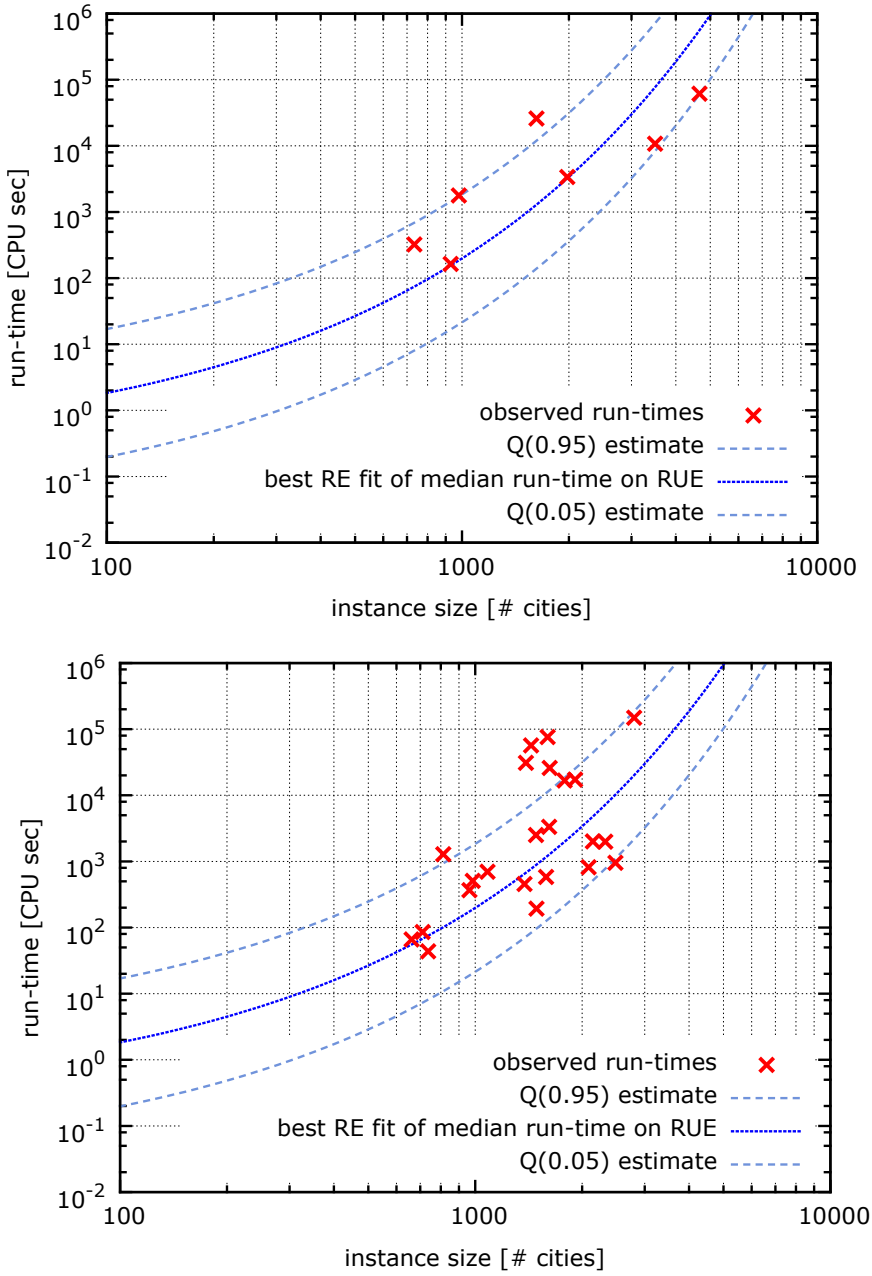

Figure 5: Run-times required by the Concorde solver for solving National instances with size $n = 500 \ldots 5000$ (top pane) and VLSI instances with size $n = 500 \ldots 4500$ (bottom pane). The dashed lines represent the best-fit root-exponential scaling determined on RUE instances and estimates of the 0.05- and 0.95-quantiles of the distributions of run-times over sets of RUE instances of a given size. 31 of the 54 VLSI instances that fall within the range of sizes considered here could not be solved by Concorde within 7 CPU days, and are therefore not shown in the plot.