Sleegers, Joeri; Olij, Richard; van Horn, Gijs; van den Berg, Daan

Article

Where the really hard problems aren't

Operations Research Perspectives

Provided in Cooperation with:
Elsevier

Suggested Citation: Sleegers, Joeri; Olij, Richard; van Horn, Gijs; van den Berg, Daan (2020) : Where the really hard problems aren't, Operations Research Perspectives, ISSN 2214-7160, Elsevier, Amsterdam, Vol. 7, pp. 1-6, https://doi.org/10.1016/j.orp.2020.100160

This Version is available at:
https://hdl.handle.net/10419/246430

Operations Research Perspectives 7 (2020) 100160

Contents lists available at ScienceDirect

Operations Research Perspectives

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

Where the *really* hard problems aren't

Joeri Sleegers[a], Richard Olij[a], Gijs van Horn[b], Daan van den Berg[*,c]

[a] University of Amsterdam, The Netherlands
[b] Indivirtual and University of Amsterdam, The Netherlands
[c] Informatics Institute, University of Amsterdam, The Netherlands

ARTICLE INFO

Keywords:
ATSP
TSP
Replication
Instance hardness
Branch and bound

ABSTRACT

Not all problem instances in combinatorial optimization are equally hard. One famous study "Where the *Really* Hard Problems Are" shows that for three decision problems and one optimization problem, computational costs can vary dramatically for equally sized instances. Moreover, runtimes could be predicted from an 'order parameter', which is a property of the problem instance itself. For the only optimization problem in the study, the asymmetric traveling salesman problem (ATSP), the proposed order parameter was the standard deviation in the probability distribution used for generating distance matrices. For greater standard deviations, most randomly generated instances turned out to be easily solved to optimality, whereas smaller standard deviations produced harder instances. In this replication study, we show these findings can be contested. Most likely, the difference in instance hardness stems from a roundoff error that was possibly overlooked. This gives rise to a sudden emergence of minimum-cost tours, a feature that is readily exploited by most branch and bound algorithms. This new contradiction renders the earlier proposed order parameter unsuitable and changes the perspective on the fundamentals of ATSP instance hardness for this kind of algorithm.

1. TSP

Surely one of the most iconic problems in computational history, the Traveling Salesman Problem[1] (TSP), has a broad range of appearances and applications. In its most general formulation, and without presumptions about specific instances, it falls into one of two complexity classes. As a decision problem ("Is there a tour smaller than k?"), it is NP-complete, because although finding such a tour takes non-polynomial time in the worst case for the best known algorithm, verifying whether a found solution actually *is* shorter than k takes only polynomial time: simply add up all the individual tour segments. Contrarily, TSP in its optimization formulation ("what is the shortest tour?") is NP-hard, but not NP-complete, because finding the shortest tour not only takes nonpolynomial time (again), but there is also no known faster way to verify that a given solution is actually shortest than to simply try all the tours - in again nonpolynomial time. Many authors, including us, believe that in the most general sense, the optimization problem is the harder of the two, not only for the lack of a non-polynomial verification procedure, but also because the optimization problem usually involves finding *the* shortest tour, instead of finding *any* shorter-than-k tour, which is the case for the decision problem. There is an intimate relationship between the two though: when the optimization problem is treated as a sequence $t = \{1, 2...n\}$ of search problems with an increasingly tightening upper bound upp_t, there might be a polynomial multiplicative runtime relationship between the two if the minimum difference between upp_t and upp_{t+1} is known, such as when the cost matrix is integer based, especially when the numerical diversity of its entries is low.

Exact algorithms for the TSP optimization problem always give the shortest tour, regardless of instance specifics. Perhaps the most straightforward of these is an enumeratively exhaustive depth-first search which runs in factorial runtime [1]. In practice, extending the algorithm by pruning off partial tours along a problem instance's upper and lower bounds ("branch and bound") can reduce these runtimes considerably, especially when starting off with a good initial bound, but its worst case is still $O(n!)$. The branch and bound paradigm has two major 'algorithmic families', depending on the underlying data structure: a stack-based depth-first implementation, or a (priority) queue based design. An early implementation from this second family is the algorithm by Little et al. (henceforth: 'Lital'), which runs in a

* Corresponding author.
E-mail addresses: joeri.sleegers@student.uva.nl (J. Sleegers), richard.olij@student.uva.nl (R. Olij), contact@gijsvanhorn.nl (G. van Horn), d.vandenberg@uva.nl (D. van den Berg).

[1] Without loss of generality, traveling sales nowadays can readily be carried out by men, women, persons or artificially intelligent airborne agents. For historical reasons however, we will stick to 'salesman'.

https://doi.org/10.1016/j.orp.2020.100160
Received 9 November 2019; Received in revised form 23 June 2020; Accepted 23 June 2020
Available online 27 June 2020

dinosaurical $O(2^{(n^2)})$, but still managed to solve relatively large instances for the time, which might be due to its clever choice of branching [2]. A better runtime upper bound is achieved by another member of this algorithmic family, the algorithm best known as 'Held-Karp', but discovered slightly earlier by Richard Bellman [3,4]. It runs in $O(n^2 \cdot 2^n)$ and thereby to this date still has the lowest time complexity of all known exact algorithms. In fact, finding an exact algorithm for TSP that runs in $O^*(c^n)$ with $c < 2$ was stated as an open problem (#31) by Gerhard Woeginger in his survey on exact algorithms for NP-hard problems [5].

By explicitly stating this complexity bound as an open problem, Woeginger firmly underlines the impracticality of exact algorithm runtimes. It is hardly surprising therefore, that many generic heuristic algorithms such as simulated annealing, ant colony optimization, genetic algorithms and the plant propagation algorithm have been applied to the TSP [6–10]. Even though heuristic algorithms, contrary to exact algorithms, waive the absolute guarantee of an optimal solution, many give good answers in little time, sometimes even providing an 'anytime solution': the longer it runs, the better it gets. But a good heuristic for TSP can also be a tool for 'hacking' its NP-hardness: if one heuristically finds a very good tour, but then subsequently proves that no better tour exists, the end result is the same as running an exact algorithm " without suffering its oppressive runtime.

Such is the work of Applegate et al. [12]. Using the specialized Lin-Kernighan heuristic, Dantzig et al.'s cutting plane method and a branch-and-cut approach, they found optimal tours for instances of 15,112, of 24,978 and of 85,900-city[2] Euclidean TSP-instances from Gerhard Reinelt's benchmark TSP-library [12–16]. The computation took 136 CPU-years, but their work of heuristically finding a good solution first and actually proving it to be optimal *afterwards* pries at the very barrier that separates P from NP, but also stirs the interplay between problem, instance, and algorithm [17].

Even though these achievements are respectable, their results are not general. It raises the question how hard these individual instances actually are, and how their method scales to a broader range of input data (for a similar discussion on a decision problem, see [18]). A partial answer is given in a particularly interesting study by Stützle et al., who deployed both Applegate's Concorde and Helsgaun's Lin-Kernighan implementation to (partially) structured Euclidean TSP-instances made up from grids and fractals, so that the shortest tour is a priori known. When perturbing these TSP-instances with various intensities, these authors show that as the structure deteriorates, instances gets harder to solve [19].

For the asymmetric (and therefore non-Euclidean) TSP, an influential study by Cheeseman et al. (henceforth: 'Cetal') exists. Cited over 1400 times, the study reports an experiment on the asymmetric TSP that shows there is great variety in solving difficulty from one instance to the next [20]. When the random generator used for making cost matrices is parameterized with a low standard deviation (σ), the computational effort is a number of magnitudes greater than for cost matrices with high standard deviations, which are solved almost instantly. As such, σ was proposed an 'order parameter' (or 'predictive data analytic') for accurately predicting algorithmic runtimes. Identifying such order parameters might have profound implications for problems in NP, and eventually possibly the $P \stackrel{?}{=} NP$ problem itself.

These findings however, are debatable, as their findings appear to be an effect of the specific distribution of the random generator, the subsequent roundoff to integer values, and the resulting sudden emergence of zero-cost tours, which effectively nullifies an instance's hardness. In other words: the choice of σ as a predictive analytic for ATSP is not generally applicable, and relies on a possibly overlooked roundoff error. And even though the literature on (A)TSP is vast, a direct replication of Cetal's experiment appears unavailable. Their experiment on the Hamiltonian cycle problem from the same study however, has been successfully replicated [21].

The rest of this paper is organized as follows: first we will describe Cetal's original experiment on the ATSP as closely as possible. Then, we will describe our replication study, after which the results are presented. Finally, an explanation for the differences between the original results and our replication study is given, discussed, and positioned in the context of related work.

2. The original experiment

In their original experiment on the asymmetric TSP, Cetal create an unspecified number of "[integer-valued cost matrices that in general are not symmetric. We choose cost matrices with a mean edge of 10 but with varying standard deviations. The results of running Little's algorithm for 16, 32 and 48 city instances with random cost matrices constructed according to a log-normal distribution with a given standard deviation are shown.]"[3]. Though Cetal neither explicitly state the values for σ nor the number of generated cost matrices, one can visually infer from their Figure 5 that for each of the three instance sizes, standard deviations are 26 values of $\sigma = \{0, 0.2, 0.4, ...4.8, 5.0\}$ with approximately 20 random cost matrices each, totaling 520 random matrices for each of the 16-, 32- and 48-city instance sizes (also see the inset in Fig. 1). We were unsuccessful in contacting the authors to verify these exact numbers.

These 1560 randomly generated TSP-matrices are then solved by Lital's algorithm ("the best exact algorithm we could find")[20], which was developed nearly three decades earlier on an IBM 7090 at the MIT Computation Center[2]. Indeed a surprisingly sophisticated algorithm for the time, it deploys the branch and bound[4] principle from the 'best-first family' in a somewhat counterintuitive but very effective way. Instead of stepwise extending partial tours, Lital's algorithm chooses individual costs between cities (cells in the cost matrix), independent from the order in which they would appear in the final tour. This particular method of choice aims to minimize the total number of steps needed to exhaustively find the shortest tour - and thereby the total runtime. Although deployed to integer matrices in both Cetal's experiment and Lital's original report, the algorithm can operate on floating point value matrices just as well.

Lital's algorithm starts off by "reducing the matrix": from each row r, it subtracts the minimum value in that row $r_{\min}$ from all cells, resulting in at least one zero for each row in matrix M. It then does the same for minimum values $c_{\min}$ for every column c, and adds up all $r_{\min}$ and $c_{\min}$ which amount to a lower bound on any final tour on matrix M. One of these zero cells, and its corresponding city pair (x, y), will be selected for branching. To optimally decide *which* zero cell to choose, a value $\theta(x, y)$ is calculated for each zero cell, which is the sum of the lowest value in the zero cell's row and column, other than the zero cell itself. Thereby, θ can be seen as the minimal consequence of *not* choosing (x, y). The zero cell with the highest θ is then selected for branching. In doing so, the 'right-hand' branch holds the subset of all the tours containing (x, y); it has the same lower bound as M, since a zero cell was chosen. The 'left-hand' branch holds the subset of all tours without (x, y); it has the same lower bound as M plus θ(x,y), the minimum consequence of not choosing (x, y).

Then the algorithm "[selects the next node for branching, by picking

[2] The first two instances contain real topographical data from cities in Germany and Sweden. The third and largest instance is actually a VLSI routing problem from the chip manufacture industry, and therefore does not contain "cities" as such, but has been remodeled to Euclidean TSP by the authors.

[3] Even though Cetal use the word "city", their TSP-instances are clearly non-Euclidean.

[4] Lital's study actually introduced the term "branch and bound", but the authors rightfully cite Ailsa Land and Alison Doig, who invented the principle [22].

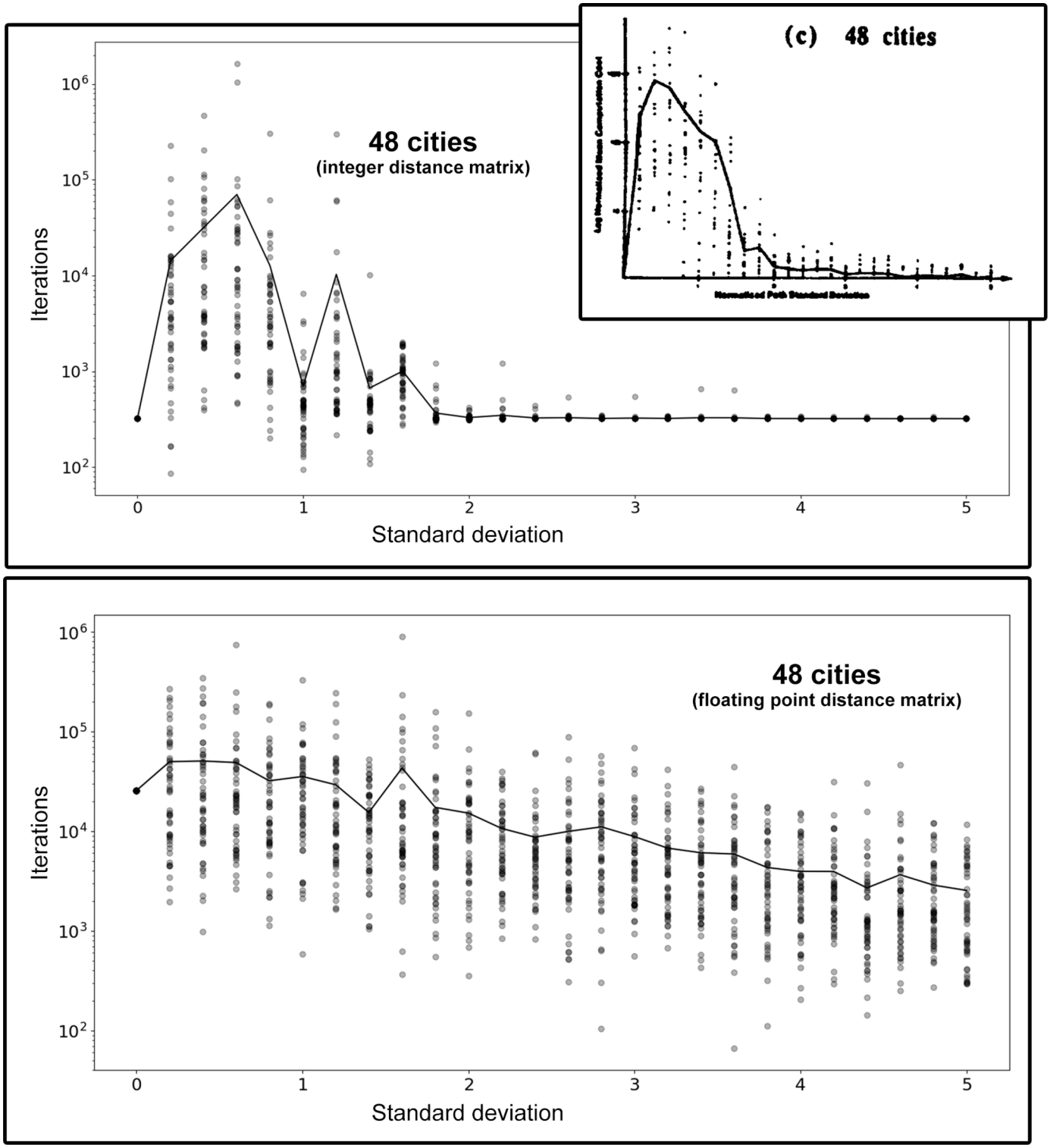

Fig. 1. Results from the replication of Cetal's experiment solving asymmetric TSP with Lital's algorithm on lognormally generated integer cost matrices (top half) are roughly equivalent to the originals (top right inset), showing large and sudden differences in computational costs as σ increases. However, if the roundoff to integers values is *omitted* for the same matrices, the effect completely disappears (bottom half). These graphs can interactively be consulted online [11].

the node with the smallest lower bound.]" [2]. This means two things: first, the algorithm uses (or could use) a priority queue[5], much like an efficient implementation of A*, or Dijkstra's algorithm [23,24]. Second, if a zero-cost tour is present in the matrix, the algorithm is likely to find it in the minimum number of operations ($O(n)$ operations for an n-city instance size) by constantly branching to the right. Since the algorithm after finding its first tour "[checks to see whether the search is finished - whether the best tour so far has cost less or equal to the lower bounds on all terminal nodes on the tree]"[6] [2], it will also immediately halt

[5] Maybe even *should* use a priority queue - a technicality inferred and implemented in this study, but not explicitly specified by Lital. The specific choice of data structure can result in small differences in the results.

after finding a zero-cost tour, since no shorter (partial) tour exists. This is an important principle for any branch and bound algorithm, but will also play a key role in explaining the results later on.

It should be carefully noted that both of both Cetal's and Lital's paper, various versions circulate that differ substantially in (technical) details. We supply pdfs of the referenced papers used for this study, as well as a refurbished version of Lital's paper [25]. Although Lital's study is well written and largely unambiguous, some implementational details could cause small differences in the results. The source code of our version of the algorithm therefore, as well as the source data of the TSP-matrices generated for this study, is publicly available [26], and we encourage other teams to replicate and further experiment with our results. Instead of also supplying a refurbished version of Cetal's paper, we intend to supply a completely new interactive online replication report once all four experiments from the original study are done. Graphs of this study however, are already interactively available online [11].

3. The replication and the results

For the replication, we generate 50 cost matrices with log-normally distributed values for each of the 26 standard deviations $\sigma = \{0, 0.2, 0.4, ...4.8, 5.0\}$ for each of the 16-city, 32-city and 48-city TSP-instances, resulting in 1300 random cost matrices for each instance size, totaling up to 3900 random integer matrices for the replication. Note that these numbers are approximately $2\frac{1}{2}$ times greater than Cetal's original study, thereby not only eliminating possible counting errors, but also increasing the robustness of the presumed effects. Like the original study, the matrices are integer valued, but before rounding off values to the nearest integer, we *also* ran Lital's algorithm on the 3900 corresponding floating-point matrices.

Our results when using the integer cost matrices are quite similar to Cetal's for all instance sizes (we only show 48 cities), and there indeed appears to be a critical zone with very hard TSP-instances for 'intermediate values' of the standard deviation, roughly $0.2 \leq \sigma \leq 1.6$. For larger values of σ, all instances are easy, which makes it an immensely strong predictive analytic for computation time. This effect however, completely disappears when the corresponding floating point matrices from *before* the roundoff are used (fig. 1)[7]. So the right question seems not to be why the effect disappears, but why it appears in the first place, and the answer is quite surprising.

4. Explanation of the results

The lognormal probability distribution indeed provides an open-ended scale with nonnegative values for any given mean μ and standard deviation σ. But as its standard deviation increases, so do its extremities: a few very large values are 'compensated' by many small values, as μ remains fixed at 10. When subsequently converting to integers, any value smaller than 0.5 gets rounded to zero. As such, the σ-value of the distribution is directly related to the expected number of zeroes in the resulting cost matrix, and it can be calculated exactly.

As integer roundoff to zero occurs for floating points values below 0.5, the chance of any cell in the integer cost matrix being zero (or lower than 0.5 in the corresponding floating-point matrix) for some σ is given by the cumulative lognormal distribution:

$$F_X(x) = \phi\left(\frac{ln(x) - \mu}{\sigma}\right) \tag{1}$$

in which ϕ is the cumulative distribution function of the standard normal distribution. Substituting $x = 0.5$ and fixating $\mu = 10$ gives the probability $P_{zero}(n, \mu, \sigma)$ of a cell being zero in the integer cost matrix after roundoff (or equivalently: being smaller than 0.5 *before* roundoff) for any given from Cetal's experiment. The probability $P_{zero}(n, \mu, \sigma)$ increases rapidly with σ, is well over 0.5 for $\sigma = 3$ and very close to 1 for $\sigma = 5$ (see inset in Fig. 2). Multiplying $P_{zero}(n, \mu, \sigma)$ with $n{\cdot}(n-1)$ for an n-city asymmetric TSP instance gives $N_{zero}(n, \mu, \sigma)$, the expected number of zeroes in the integer cost matrix of the corresponding n-city instance.

As mentioned in Section 2, the existence of a zero-cost tour has a tremendous effect on Lital's algorithm, reducing its runtime to linear complexity. The probability of a distance matrix generated in Cetal's experiment actually *having* a zero-cost tour can also be calculated exactly, and there is an interesting relation with the first experiment (Hamiltonian cycle problem instance hardness) from the exact same study by Cetal.

With some abstraction, the probability of a zero-cost tour in an asymmetric TSP cost matrix of n cities and $N_{zero}(n, \mu, \sigma)$ zeroes is identical to the probability of an unweighted graph of n vertices and E randomly placed edges having a Hamiltonian cycle. This latter probability is given by János Komlós and Endre Szemerédi for undirected graphs as

$$P_{Hamiltonian}(V, E) = e^{-e^{-2c}} \tag{2}$$

in which

$$E = \frac{1}{2}\, V{\cdot}ln(V) + \frac{1}{2}\, V{\cdot}ln(ln(V)) + cV \tag{3}$$

[27]. In the set of equations (2) and (3), the term $\frac{1}{2}\, V \ln(V) + \frac{1}{2}\, V \ln(\ln(V))$ denotes the point of the 'phase transition', a sudden jump from the graphs being almost-surely-nonHamiltonian to almost-surely-Hamiltonian, and it gets steeper for larger graphs. The corresponding point for a *directed* Hamiltonian cycle, which is relevant for this investigation, is given by McDiarmid as

$$m = q{\cdot}|V{\cdot}(ln(V) + ln(ln(V))| \tag{4}$$

in which q is a constant [28]. The actual value of q at which the phase transition occurs[8] has been empirically estimated by Vandegriend and Culberson at $q = 1.08$, which is probably too high [29]. These authors understandably took the probability of 0.50 as threshold point of the phase transition, but theory dictates the point of the maximum derivative revolves around a somewhat counterintuitive $e^{-1} \approx 0.37$ (which also appears to fit their data better, see Figure 1 in [29]). Jäger and Zhang later estimated $q \approx 0.9$ [30].

From our randomly generated distance matrices after integer roundoff, we count the numbers of zeroes for all 50 matrices for any given $\sigma = \{0, 0.2, 0.4, ...4.8, 5.0\}$ and compare the average number of zeroes per matrix to the expected number of zeroes per matrix as given from equation (1) (results are in fig. 2). Then, for all $\sigma = \{0, 0.2, 0.4, ...4.8, 5.0\}$, we counted the fraction of the 50 matrices that turned out to have a zero-cost tour, and fitted equation (2) to the data points with the Levenberg-Marquardt method to approximate the point of the phase transition. From the fit to equation (4), we find that for 48-city instances, the threshold for existence of a zero-cost tour lies at 215 zeroes ($\chi^2 \approx 1.18{\cdot}10^{-5}$). This corresponds to $q \approx 0.86$ which is relatively close to Jäger and Zhang's value of $q \approx 0.9$ (whose value would correspond to 225 zeroes for equally-sized instances).

[6] "On the tree" can also be understood as "in the priority queue".

[7] We received feedback that for the inset of Fig. 1, the axes' lables are hardly readable. This is also the case in the original paper, which was the original motivation for this replication study.

[8] The variable q was originally designated as c in studies mentioned in this paragraph. We replaced it to avoid confusion with equation (2), which is used in a large number of studies on random graphs, including Komlós and Szemerédi's

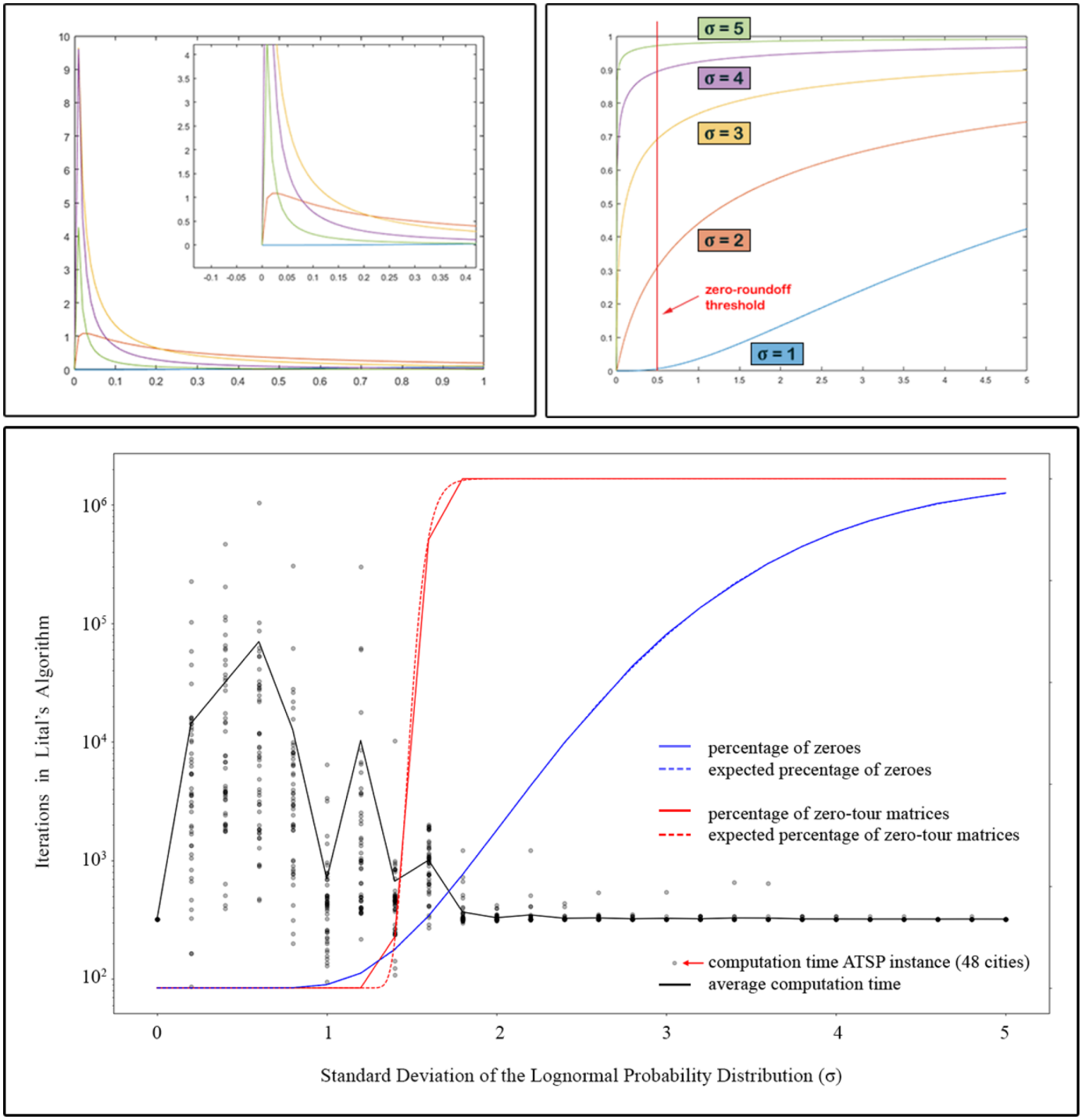

Fig. 2. The lognormal probability distribution (top left) and the related cumulative probability distribution (top right) as used by Cetal for generating asymmetric TSP cost matrices. As σ increases for a constant $\mu = 10$, the expected number of values below 0.5 increases along with it. In a roundoff, these values get reduced to zero, eventually leading to the sudden appearance of a zero-cost tour in the cost matrices (bottom). As Lital's algorithm carefully exploits lower bounds during its tour construction, its runtime is directly related to the existence of zero-cost tours.

5. Conclusion and discussion

It seems that the hard-easy phase transition in Cetal's asymmetric TSP-instances is not a consequence of increasing σ in itself, but of the gradually increasing number of small values that become zeroes at roundoff, and the resulting sudden emergence of zero-cost tours (Fig. 2, bottom). Because Lital's algorithm (and many others) exploits zero-cost tours by preferring to expand on lower bound subsets via its matrix reduction and clever choice of θ, it is no surprise that the runtime drops dramatically as the probability of zero-cost tours in Cetal's randomly generated cost matrices increases.

These findings might be regarded as complementary to the work of WeiXiong Zhang and Richard Korf [31]. These authors find that instance hardness for the asymmetric TSP depends on the expected numerical diversity, instantiated by parameter $rand_{\max}$[9] when generating

[9] Originally designated as $r_{\max}$. Substituted to avoid confusion.

asymmetric TSP cost matrices from uniform random integers $0 \leq r \leq rand_{\max}$. Solving is relatively easy when $rand_{\max}$ is low, but instances get substantially harder as $rand_{\max}$ increases. This is interesting for three reasons.

First, their algorithm is branch and bound too, but from the *opposite* algorithmic family, being a depth-first branch and bound instead of a best-first branch and bound. Second, we suspect that their easy instances of the asymmetric TSP are *also* easy instances for 'our' branch and bound algorithm, because Lital's matrix reduction step would generate a relatively high number of zeroes, which again triggers the sudden emergence of a minimal-cost tour (which isn't necessarily zero in this case). Third, we suspect the converse holds too, as 'our' easy instances might also be easy for their algorithm, again from the existence of a zero-cost tour due to lognormal generation and integer roundoff.

But whereas the concept of expected numerical diversity in itself might be sufficient for characterizing the hardness of asymmetric TSP-instances for Zhang & Korf's algorithm, it is certainly not so for Lital's, which terminates quickly if the lowest value in the matrix (zero or otherwise) has a prevalence of at least $q \cdot \lfloor V \cdot (ln(V) + \ln(ln(V))) \rfloor$ (in which $q \approx 0.9$), regardless of its total numerical diversity. To what extend these principles are universal remains to be seen.

A last but important point of discussion concerns the origin of the order parameter. In the most desirable case, it is a property of the problem instance alone, serving as a predictive data analytic for the performance of one or more solving algorithms [32]. For many decision problems, like SAT, Hamiltonian cycle or even (A)TSP in its decision form, dramatic peaks in computational costs are found along a steep phase transition in solvability, which in terms usually aligns with the order parameter of constrainedness [33] [21] [34] [35] [36].

Most of these investigations involve large numbers of instances generated by parameterized random functions. By nature, randomized ensembles will *approach* a certain stochastic parameter (like μ or σ) in the size limit, but not *guarantee* it for an individual outcome (the difference is very well illustrated in a study by Barbara Smith [37]). In Cetal's study, this has a dramatic effect because of the possibly overlooked roundoff, but in every case, it inevitably brings noise to the robustness and predictive power of the resulting order parameter. Even though constraining variables brings along other problems, such as overrepresentation of instances in the order parameter's extremes, an exact instance property is still favourable over a stochastic one.

Declaration of Competing Interest

The authors declare that they do not have any financial or non-financial conflict of interests

Acknowledgements

Gratitude goes out to Peter van Emde Boas, Cees de Laat, Pieter Adriaans, Sandjai Bhulai, Richard Karp, Gerhard Woeginger, Richard Korf and especially WeiXiong Zhang for helping out with details and proofreads.

References

[1] Russell SJ, Norvig P. Artificial intelligence: A Modern approach (third edition). Pearson Education, Inc. Upper Saddle River, New Jersey 07458; 2016.
[2] Little JD, Murty KG, Sweeney DW, Karel C. An algorithm for the traveling salesman problem. Oper Res 1963;11(6):972–89.
[3] Held M, Karp RM. A dynamic programming approach to sequencing problems. J Soc Indust Appl Math 1962;10(1):196–210.
[4] Bellman R. Combinatorial processes and dynamic programming. Tech. Rep.. The RAND Corporation, 1700 Main St., Santa Monica, California; 1958.
[5] Woeginger GJ. Exact algorithms for NP-hard problems: a survey. Combinatorial optimization – eureka, you shrink!. Springer; 2003. p. 185–207.
[6] Kirkpatrick S, Gelatt CD, Vecchi MP. Optimization by simulated annealing. Science 1983;220(4598):671–80.
[7] Dorigo M, Gambardella LM. Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans Evol Comput 1997;1(1):53–66.
[8] Grefenstette J, Gopal R, Rosmaita B, Van Gucht D. Genetic algorithms for the traveling salesman problem. Proceedings of the First International Conference on Genetic Algorithms and their Applications. 160. Lawrence Erlbaum; 1985. p. 160–8.
[9] Selamoğlu Bİ, Salhi A. The plant propagation algorithm for discrete optimisation: the case of the travelling salesman problem. Nature-Inspired Computation in Engineering. Springer; 2016. p. 43–61.
[10] Rego C, Gamboa D, Glover F, Osterman C. Traveling salesman problem heuristics: leading methods, implementations and latest advances. Eur J Oper Res 2011;211(3):427–41.
[11] Gijs van Horn. Interactive graphs. 2019. https://travelingsalesman.nl, Last accessed on April 23rd, 2020.
[12] Applegate DL, Bixby RE, Chvátal V, Cook WJ. The traveling salesman problem: A Computational study. Princeton University Press; 2006.
[13] Lin S, Kernighan BW. An effective heuristic algorithm for the traveling-salesman problem. Oper Res 1973;21(2):498–516.
[14] Helsgaun K. An effective implementation of *K*-opt moves for the Lin-Kernighan TSP heuristic. Roskilde University, Denmark; 2006.
[15] Dantzig G, Fulkerson R, Johnson S. Solution of a large-scale traveling-salesman problem. Journal of the operations research society of America 1954;2(4):393–410.
[16] Reinelt G. TSPLIB - A traveling salesman problem library. ORSA journal on computing 1991;3(4):376–84.
[17] Applegate DL, Bixby RE, Chvátal V, Cook W, Espinoza DG, Goycoolea M, et al. Certification of an optimal tsp tour through 85,900 cities. Operations Research Letters 2009;37(1):11–5.
[18] van den Berg D, Braam F, Moes M, Suilen E, Bhulai S. Almost squares in almost squares: solving the final instance. Data Analytics 2016 2016:81.
[19] Fischer T, Stützle T, Hoos H, Merz P. An analysis of the hardness of TSP instances for two high-performance algorithms. Proceedings of the 6th Metaheuristics International Conference. 2005. p. 361–7.
[20] Cheeseman P, Kanefsky B, Taylor WM. Where the *Really* hard problems are. IJCAI. 91. 1991. p. 331–40.
[21] van Horn G, Olij R, Sleegers J, van den Berg D. A predictive data analytic for the hardness of hamiltonian cycle problem instances. Data Analytics 2018 2018:101.
[22] Land A, Doig A. An automatic method of solving discrete programming problems. Econometrica 1960;28(3):497–520.
[23] Hart PE, Nilsson NJ, Raphael B. A formal basis for the heuristic determination of minimum cost paths. IEEE transactions on Systems Science and Cybernetics 1968;4(2):100–7.
[24] Dijkstra EW. A note on two problems in connexion with graphs. Numerische Mathematik 1959;1(1):269–71.
[25] Daan van den Berg. Refurbished version of Lital's paper. 2019. http://heuristieken.nl/wiki/index.php?title=CheesemanTSPReplication, Last accessed on April 26th, 2020.
[26] Joeri Sleegers. Source Code of the Implementation. 2019. http://heuristieken.nl/wiki/index.php?title=CheesemanTSPReplication, Last accessed on 2019-7-10.
[27] Komlós J, Szemerédi E. Limit distribution for the existence of hamiltonian cycles in a random graph. Discrete Math 1983;43(1):55–63.
[28] McDiarmid C. Clutter percolation and random graphs. Combinatorial Optimization II. Springer; 1980. p. 17–25.
[29] Vandegriend B, Culberson J. The $g_{n,m}$ phase transition is not hard for the hamiltonian cycle problem. Journal of Artificial Intelligence Research 1998;9:219–45.
[30] Jäger G, Zhang W. An effective algorithm for and phase transitions of the directed hamiltonian cycle problem. Journal of Artificial Intelligence Research 2010;39:663–87.
[31] Zhang W, Korf RE. A study of complexity transitions on the asymmetric traveling salesman problem. Artif Intell 1996;81(1–2):223–39.
[32] Sleegers J, van den Berg D. Plant propagation & hard hamiltonian graphs. Evostar 2020 æThe Leading European Event on Bio-Inspired Computationg. 2020.
[33] Mitchell D, Selman B, Levesque H. Hard and easy distributions of SAT problems. AAAI. 92. 1992. p. 459–65.
[34] Gent IP, Walsh T. The TSP phase transition. Artif Intell 1996;88(1–2):349–58.
[35] Gent IP, MacIntyre E, Prosser P, Walsh T. The constrainedness of search. AAAI/IAAI, Vol. 1. 1996. p. 246–52.
[36] Prosser P. An empirical study of phase transitions in binary constraint satisfaction problems. Artif Intell 1996;81(1–2):81–109.
[37] Smith BM. The phase transition and the mushy region in constraint satisfaction problems. Proceedings of the 11th European Conference on Artificial Intelligence. John Wiley & Sons, Inc.; 1994. p. 100–4.