Where the really hard problems actually are

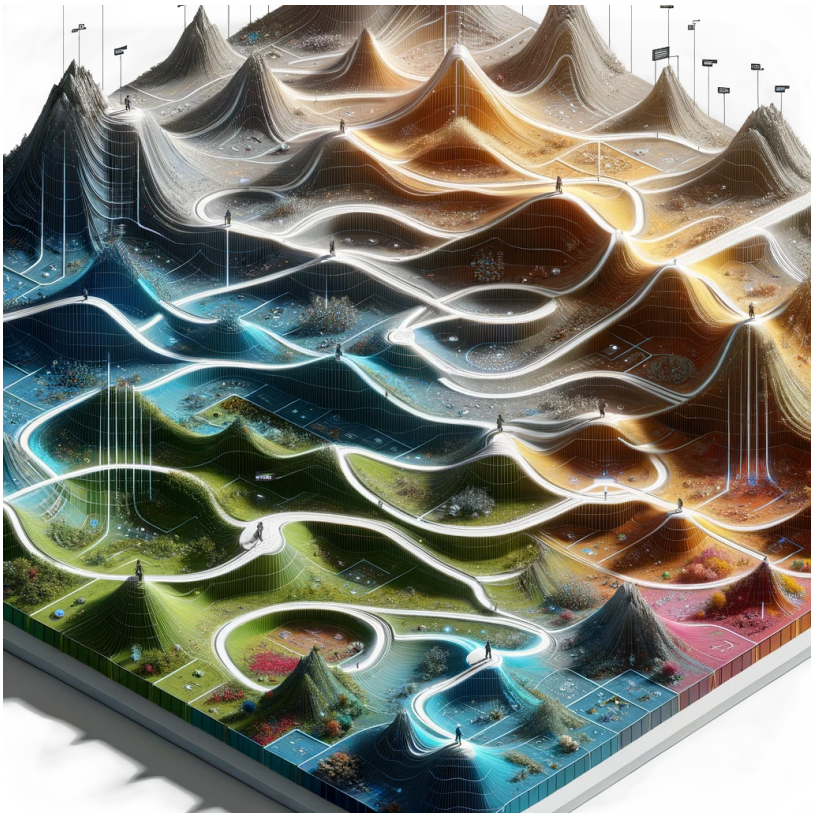
Hill-Climbing for Hardness on the Asymmetric Traveling Salesman Problem

submitted in partial fulfillment for the degree of master of science

Wouter Knibbe
12795526

Master Information Studies
data science
Faculty of Science
University of Amsterdam

Submitted on 25-06-24



	Supervisor	2nd Examiner
Title, Name	Daan van den Berg	Dr. Herke van Hoof
Affiliation	VU Amsterdam	University of Amsterdam
Email	d.van.den.berg@vu.nl	h.c.vanhoof@uva.nl

ABSTRACT

This study contributes to the quest of understanding what makes the Asymmetric Traveling Salesman Problem (ATSP) hard. It investigates over 1,000 randomly generated distance matrices with 30 to 70 cities, and integer values between 1 and a maximum ($rand_{max}$) of 10 through 50. Little's algorithm is used to solve the matrices and its expended number of iterations defines their hardness. Matrices are progressively made harder by mutating and solving them again, hill-climbing them on hardness for 4,000 generations. The hill-climber increased the hardness of every matrix investigated, up to a median increase within a parameter combination of at least 1,018 times. The results suggest the hill-climber is effective because ATSP's space of problem hardness can be highly irregular, with some small changes leading to shifts in hardness of over three orders of magnitude. However, the hill-climber does seem to get stuck in local optima. Phase transitions in hardness are found at critical $rand_{max}$ values. They occur at earlier $rand_{max}$ for smaller city sizes, and even earlier when hill-climbed. The hill-climber had the largest effect after the phase transition.

KEYWORDS

ATSP, TSP, Hill-climbing, Instance hardness, Phase transitions

GITHUB REPOSITORY

https://github.com/WouterKnibbe/ATSP_hillForHard

1 INTRODUCTION

The classic Traveling Salesperson Problem (TSP) one of the most famous and extensively studied problems in the field of combinatorial optimization. In the standard TSP, a salesperson is tasked with visiting a set of cities exactly once and returning to the starting city (i.e. a 'tour'), minimizing the total distance traveled. The usual method of efficiently representing the distances between each city to a computer is a distance matrix. The rows and the columns of the matrix both list all the cities in the instance. The values in the matrix represent the distances between each possible combination of cities. It is not allowed to travel from one city to itself, so the diagonal of the matrix is filled with 'infinity' or zero values. When mapping the distances between real cities, the distance matrix will always be diagonally symmetric because, in euclidean space, the distance from city A to B is the same as that from B to A. An example of such a matrix is shown in Figure 1, notice that the matrix is symmetric with zeroes on the diagonal.

In Asymmetric TSP (ATSP) the symmetry constraint is relaxed, allowing the distance from city A to B to be different than the distance from city B to A. Such instances are commonly referred to as 'non-euclidean', as they are not realistic in physical space. TSP problems with symmetric distance matrices will be referred to as euclidean TSP (ETSP). The points which require travelling to are still called 'cities' in all forms of TSP. For a detailed mathematical, yet approachable description of the problem, the work by Hoffman and colleagues is highly recommended as a starting point [9].

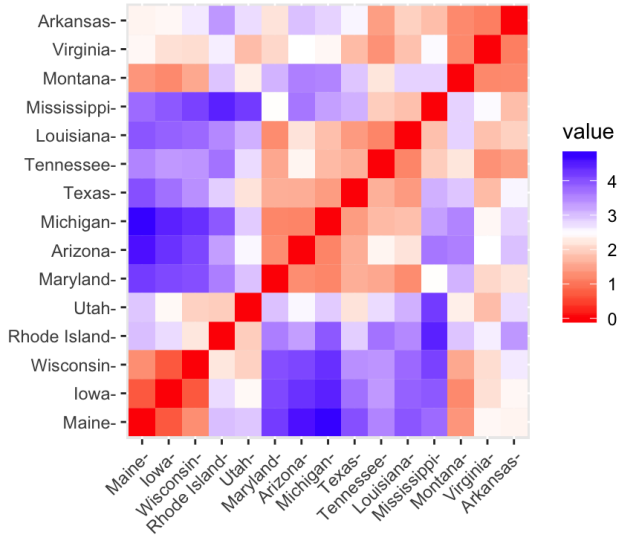


Figure 1: Floating-point distance matrix of distances between 15 states in the USA. Courtesy of [1]

ATSP optimization is NP-hard [13], which means that finding the shortest tour takes at least super-polynomial time. This is a statement about the time it takes to both find and verify solutions, as both are necessary to find the shortest possible tour. This is not always the case, as some problems may take super-polynomial time to find a solution, but only polynomial time to verify it is optimal (classified as NP). But because both finding and verifying the shortest ATSP tour take super-polynomial time, it is NP-hard [13]. In this field of study, hardness is the type of difficulty of a problem that has a numerical definition. Ways of defining hardness in various ways are described in Section 2.

Being able to predict the hardness from a distance matrix is beneficial because this guides the decision-making process distributing computational resources. Many real-world problems, from logistics to crystallography to DNA sequencing, can be abstracted as finding the shortest tour [2][4][3].

Many previous authors have contributed to the current understanding of what makes ATSP problem instances hard. In this work phase transitions in hardness are found related to characteristics of the distance matrix. The term phase transition comes from chemistry and thermodynamics. It describes that water stays liquid at many temperatures, but when it goes below a certain threshold, it suddenly all becomes ice very quickly. In the context of Computer Science this means that at a critical threshold of a numerical characteristic there is a sudden change in another characteristic. An example phase transition in ATSP is when the fraction of distinct distances in the matrix crosses a critical threshold, its hardness falls dramatically [18].

A parameter related to distinct distances, which has also been observed to cause phase transitions in hardness for ATSP, is the

random maximum ($rand_{max}$), investigated by Zhang & Korf [19]. It defines the integer values between 1 and a maximum value (exclusive) any randomly generated number in the matrix may take on. Zhang & Korf found phase transitions of multiple complexity indicators for branch-and-bound algorithms in integer distance matrices of $nCities$=100 between a $rand_{max}$ of 1 and 100 [19].

This study builds onto this work to progress the understanding about what makes distance matrices hard to solve for branch-and-bound algorithms. These algorithms are interesting, because they produce exact solutions to the problem. Distance matrices with varying city size ($nCities$) and $rand_{max}$ are investigated. A combination of mutations and hill-climbing are used to create ever harder distance matrices for Little's algorithm to solve. Harder matrices are those who take more iterations of Little's algorithm.

The research question that guides this study is: *Can distance matrices be hill-climbed on hardness?*

Sub-questions to investigate include:

- How does hardness progress between different $rand_{max}$ within the same $nCities$?
- Do matrices with the same parameters converge to a certain hardness?
- Are there any phase-transitions in hardness to speak of?
- How do the results relate to previous work?

The rest of this thesis is structured as follows. The next Section provides more context and work done in various related research. Section 3 describes the proposed methodology, including definitions of terms and an implementation of the experiment based on similar studies used as baselines. Section 4 gives an overview of the most important results, including analysis of a collection of tiled graphs showing the hardness of all the runs. Finally Section 5 and 6 interpret results, relate them to existing work, and propose directions for future work.

2 RELATED WORK

This Section provides more context and work done in various related research.

2.1 Importance of Structure

The study of Fischer and colleagues investigates the impact of varying structures of ETSP instances on the runtime of two algorithms. The study considers ETSP instances derived from highly structured problems that can be solved in polynomial time, and analyzes the effect of modifications like removing cities or perturbing city coordinates [6].

They start with ETSP instances that are structured by a checkerboard pattern on a grid and by fractal patterns in 2d space. Then, perturbations are applied to remove an increasing percentage of structure from each instance. For the grid instances, the cities were moved to a random place in 2d space, called 'shaking'. For the fractal pattern, the perturbation removed increasing number of cities, called 'reduction'. Reduction being a perturbation is based on the dynamic that fractal pattern ETSP instances are easily solved. This is because the tour can be found by moving to the closest neighbour every step. However, when cities are removed, the structure falls apart and the algorithm has to be increasingly clever to solve them. Unless most cities are removed, as an instance with few cities is also easily solved.

Results indicate that increasing the percentage of cities perturbed also increases the hardness of the instance. Also suggesting that random ETSP instances are actually relatively hard compared to structured instances. The research confirms that fractal instances indeed follow the logic proposed above, with them getting harder at first and then easier, the more cities are removed. The research finds different results for increasingly shaken grid instances. Hardness increases with the degree of shaking up to a certain level (about 50%), but for higher shaking levels, hardness stayed the same. The definition used for hardness in this research was computation time of two algorithms: Applegate et al.'s branch & cut code and Helsgaun's iterated Lin-Kernighan algorithm.

2.2 Exactness of Algorithms

The two algorithms used in Fischer et al. were found to excel at different instances of the problem, with one being especially capable at Koch fractal patterns. This is because the hardness of a problem instance can really only be defined as a function of the algorithm that is used to solve it [11]. One algorithm in their research was based on the Lin–Kernighan heuristic.

Heuristics are mathematical rules of thumb on which algorithms can be based. Such algorithms trade optimally, completeness, accuracy, and/or precision for speed [14]. Little's algorithm is exact and thus will only output an optimal result at the very end of its calculation. The output of a heuristic algorithm is not guaranteed to be the optimal solution. However, because of the difficulty of ATSP, exact algorithms struggle to produce a provably optimal solution in a reasonable time. Heuristics have the advantage that they can be stopped at any time, and they will output their best solution found so far. They are widely accepted in various contexts where these trade-offs are worthwhile [2].

Another option in such contexts are approximating algorithms. They are similar to heuristic algorithms in the sense that they have the same trade-offs. But crucially, approximation algorithms have additional complexity built in that allows them to guarantee that their solutions are within a certain range of the optimal solution. For example, the Christofides–Serdyukov algorithm, guarantees that its solutions for an ETSP instance will be within a factor of 1.5 of the optimal solution length, while running in in polynomial time [8]. This guarantee offers a quantifiable measure of optimality in the results, the absence of which makes the applicability of heuristic algorithms problematic. While still quite new, approximating algorithm's trade-offs strikes a powerful balance between being exact and fast.

2.3 Goldberg's Theoretical Foundations

Goldberg's 1989 paper served as a foundation for many modern studies into combinatorial problems and algorithms. This Section follows his exploration of the theoretical underpinnings that impact the efficiency and efficacy of optimization algorithms, with a focus on exploitation, convergence, hill-climbing, and genetic algorithms [7]. These concepts are fundamental in understanding how algorithms navigate and solve complex problem spaces.

2.3.1 Exploration and Exploitation. In optimization, exploration and exploitation refers to an algorithm's ability to cover the space of possible solutions to a problem. It is akin to exploring a landscape in the sense that there are different strategies to do so. In this context, exploration refers to looking at the landscape, while exploitation refers to the algorithm's ability to use the gathered information to refine its search in promising areas. In practice, this process is a balancing act of exploring and exploiting regions that are far apart, as well as improving on the best solutions locally. Effective algorithms balance these strategies to maximize efficiency. Genetic algorithms exemplify this balance well: Crossover mechanisms help explore promising new solution areas globally, while mutations and selection exploit solutions to refine the search further locally.

2.3.2 Convergence in Optimization Algorithms. Convergence is a critical aspect in evaluating the performance of an optimization algorithm. It describes the process by which an algorithm approaches a stable solution. Convergence signifies the point at which further iterations no longer yield a significantly better solution. An algorithm might converge prematurely to a local optimum unless mechanisms are employed to facilitate broader search capabilities.

2.3.3 Hill-Climbing and Its Limitations. Hill-climbing algorithms are straightforward yet powerful tools for optimization, particularly known for their simplicity and speed. They iteratively move to a better neighboring solution, optimizing locally until no further improvements can be made. However, their major drawback is the tendency to get stuck in local optima, failing to recognize better solutions beyond immediate neighbors. This limitation makes them less suited for problems with landscapes where local optima are abound.

2.4 Optimization Landscapes

The research of Ochoa & Veerapen made a significant contribution to the understanding of what makes combinatorial problems hard [12]. It's investigation starts with a commonly held view among researchers of TSP optimization: The 'big valley' hypothesis. Under this view, there are many local optima that are easy to escape from, and macro-level gradient over the entire landscape that leads to the global optimum. At the time when the research was conducted, recent studies on ETSP landscapes had revealed a more complex picture, but the literature on characterising its landscapes was mostly lacking. In an attempt to fill this gap, Ochoa & Veerapen used the local optima network model to characterise and visualise the global structure of ETSP fitness landscapes. To this end, they looked for 'funnels' in the data, which are groups of local optima which are close in configuration space within a group, but well-separated between groups. This structure is important because modern techniques can escape local optima relatively easily, but funnels are much harder to escape. They define local optima as solutions where 10,000 iterations of their algorithm does not yield a better solution.

They considered 20 instances of city sizes in the range of 500 to 1500 or so, and different types. A thousand independent runs of the exact Chained Lin–Kernighan implementation of the exact Concorde algorithm were executed for each ETSP instance. They considered two initialisation mechanisms, one producing better initial solutions than the other, in order to have a broader picture of the search space. Each run continued until at least 10,000 consecutive iterations were performed without finding an improving solution. After this, the unique nodes and edges produced by this sampling process were used to make the network.

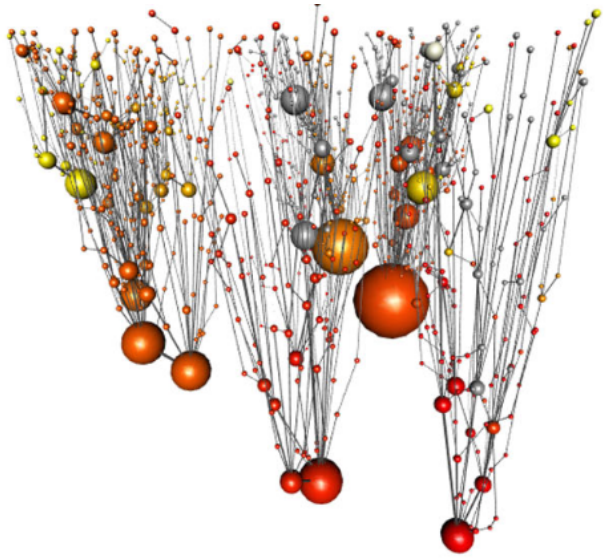


Figure 2: 3D visualization of the local optima network for a selected random instance where cities = 1243. Funnels are colored. Red indicates the funnel containing the global optimum and the yellow gradient the difference in optimality relative to it. Grey nodes belong to more than one funnel. The size of a node indicates how many nodes lead to it. [12]

Figure 2 shows an ETSP instance containing 19 funnels, crushing the assumption that there is a coarse level gradient leading to the global optimum everywhere in every instance. The authors then show additional instances with 4 to 2 funnels each.

Furthermore, they found significant differences among the studied instance classes. Randomly generated instances had a single global optimum while, structured instances from the TSPLIB generally had more than one different global optima, with some instances featuring a large global optimum plateau. They expected this was because there were many pairwise city distances with the same value in those instances. However, the random instances revealed a much larger number of funnels as compared with the structured ones. They also found that when the size of the instance increases, the number of funnels does too, while the size of the funnel containing the global optima does the opposite.

2.5 Relating Characteristics to Hardness

The same authors also measured the strongest correlations between the success rate of their algorithm and features of the distance matrices. Please refer to their table 5 for the full list of features. For uniform instances the feature that made the instances the hardest was the relative in-strength of not-global-optimum nodes without outgoing edges. Its complement, global-optimum nodes without outgoing edges made the algorithm much more likely to succeed.

In the 2010 publication by Smith-Miles & van Hemert, the Chained Lin-Kernighan- and Lin-Kernighan with Cluster Compensation algorithm were analyzed for their performance on ETSP [18]. It yielded various results of interest related to this study.

First of all, the authors defined a Feature Space ($\mathcal{F}$) of relevant, measurable characteristics of ETSP instances. Then, the main contribution of their work showed the influence of those characteristics on the hardness of ETSP problem instances for their chosen algorithms. They defined hardness as the computation time in real-world seconds of the algorithms of interest. They first used an evolutionary algorithm to create problem instances that are designed to be hard, and others to be easy, for their two algorithms, resulting in four categories. They then used a Self-Organizing Map to generate clusters onto a 2-dimensional map based on all 12 features, but not computation time itself. This process also generated four clusters, indicating that the attempt of making problems that are hard for one, and easy for another algorithm was successful. They then colored the map according to the computation time.

The Figures show characteristics that seem to correlate with the hardness of problem instances, and that the clusters are filled with computation times of the expected duration. Their visual analysis is validated with decision trees trained on the characteristics. The decision trees achieve a ~97% accuracy on unseen problem instances, confirming the characteristics as highly indicative of hardness for the respective algorithms. The rules the highly predictive decision trees generated were built around the coefficient of variation of the normalised nearest-neighbour distances (nNNds), fraction of distinct distances, cluster ratio, mean radius of the clusters, and outlier ratio. NNDs are used to study TSP by other authors as well [2]. The least indicative characteristics were the coordinates of the instance centroid and rectangular area within which the cities lie.

Attempts have also been done at tackling hardness-related characteristics of ATSP problem instances specifically. One such attempts introduced the standard deviation as predictor for hardness [5]. However, a replication study later showed that the results were likely due to rounding errors instead of patterns in the nature of ATSP [17]. Building onto this research, this study uses the same stack-based depth-first exact algorithm from Little et al. to define hardness [10].

2.6 Value Distributions and Hardness

A related study to Sleegers et al. is that of Zhang & Korf, which also focused on the phase transitions in complexity of solving ATSP using branch-and-bound algorithms [19]. They examined how the distribution of intercity distances influences these transitions.

Their research identified that ATSP exhibits distinct phase transition patterns depending on the range and distribution of expected intercity distances. When distances are uniformly distributed, there is an easy-hard transition as the expected range increases. Specifically, for distances uniformly selected from the set 0,1,2,..., $rand_{max}$, the problem tends to be easy for small $rand_{max}$, and becomes harder as $rand_{max}$ increases. They investigated 1,000 randomly generated problem instances the $nCities$=100 and $rand_{max}$ 1-10^7, inclusive, with steps of 1, 2, 3, ..., 9, 10, 20, 30, ..., 90, 100, 200, 300, and so forth, for a total of 8,000 instances.

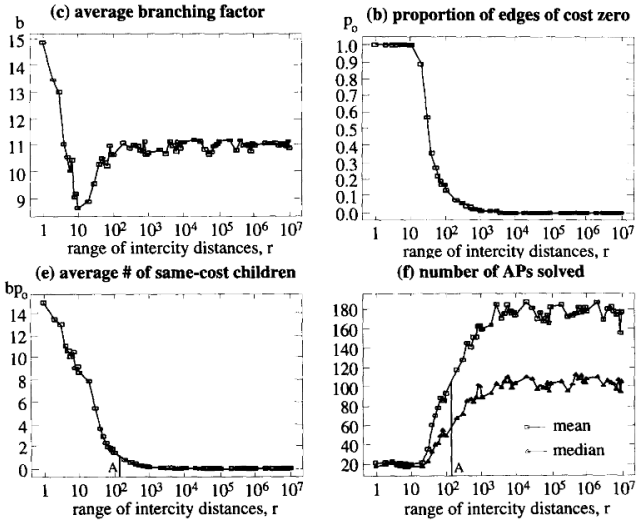


Figure 3: Phase transitions in Zhang & Korf [19]. Each point shows a matrix's relationship between its $rand_{max}$ and the characteristic in the title of the sub-figure.

Zhang & Korf show that hard and easy matrices may be separable according to their $rand_{max}$. Their results suggest hardness in ATSP evolves with an easy-hard phase transition between a $rand_{max}$ of 1-100. For example, they found that the median number of search tree nodes generated in their algorithm increased with the $rand_{max}$. This metric is similar to the amount of iterations when measuring the hardness with Little's algorithm. The relationship followed the curve in Figure 3. When $rand_{max} < 20$, the problem is easy, and becomes very hard at $rand_{max} > 100$.

Sleegers et al. [17] related their work to that of Zhang & Korf. They state that they suspect the pattern Zhang & Korf found would also apply to their algorithm, as the "*matrix reduction step would generate a relatively high number of zeroes, which again triggers the sudden emergence of a minimal-cost tour*". However, Sleegers et al. do not expect the $rand_{max}$ parameter to be enough to characterize the hardness of ATSP for Little's algorithm. Elaborating further, they explain it depends most on the lowest value in the matrix, rather than the range of values in the matrix.

The connection between the findings in ATSP and the study of the partition problem by Sazhinov et al. [15] further illustrates the impact of value distribution on problem hardness for branch-and-bound algorithms. Sazhinov et al.'s work on the partition problem reveals that the difficulty of solving an instance is not solely determined by the number of integers but also by their magnitudes, quantified in informational bits. They show that the specific distribution of these bits significantly influences the instance hardness for both exact and heuristic algorithms. Instances with a higher ratio of informational bits to the number of integers tend to have fewer perfect solutions, thus increasing the computational cost required to find optimal solutions.

This concept aligns closely with the observations in ATSP regarding the distribution of intercity distances. In both cases, NP-hard problems are observed to experience a phase transition in hardness

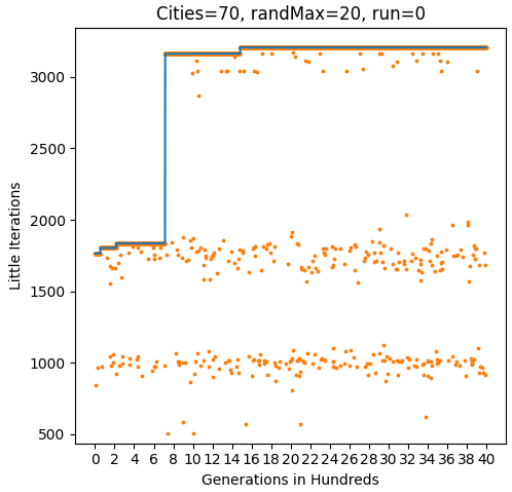


Figure 4: A single run of hill-climbing $nCities$=70, $rand_{max}$=20. In orange, every hardness measured is chronologically plotted over the generations, with the hardest matrix found at that generation in blue.

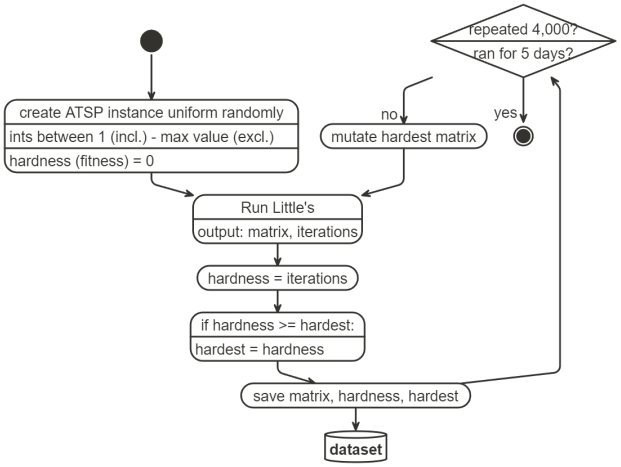


Figure 5: Summary of the methodology for generating matrices

as the range of possible values increases. Sazhinov et al. found that instances with large integers relative to the number of integers are harder to solve, much like how ATSP instances with a higher $rand_{max}$ exhibit a sudden increase in complexity, and likely hardness. The relationship between how instances are generated and their hardness are further explored with a new experiment based on these previous works, described in the next Section.

3 METHODOLOGY

The purpose of this study is to contribute to the existing quest of understanding what makes a ATSP problem instances hard. A combination of mutations and hill-climbing are used to create ever harder distance matrices for Little's algorithm to solve. Harder matrices are those who take more iterations of Little's algorithm.

3.1 Aim

Distance matrices are initialized according to a set of parameters. These are the city size ($nCities$) and the allowed integer values between 1 and an exclusive maximum ($rand_{max}$). A 'run' is the process of hill-climbing one distance matrix until the time limit or generation limit is reached. A visualization of one such run is reported in figure 4.

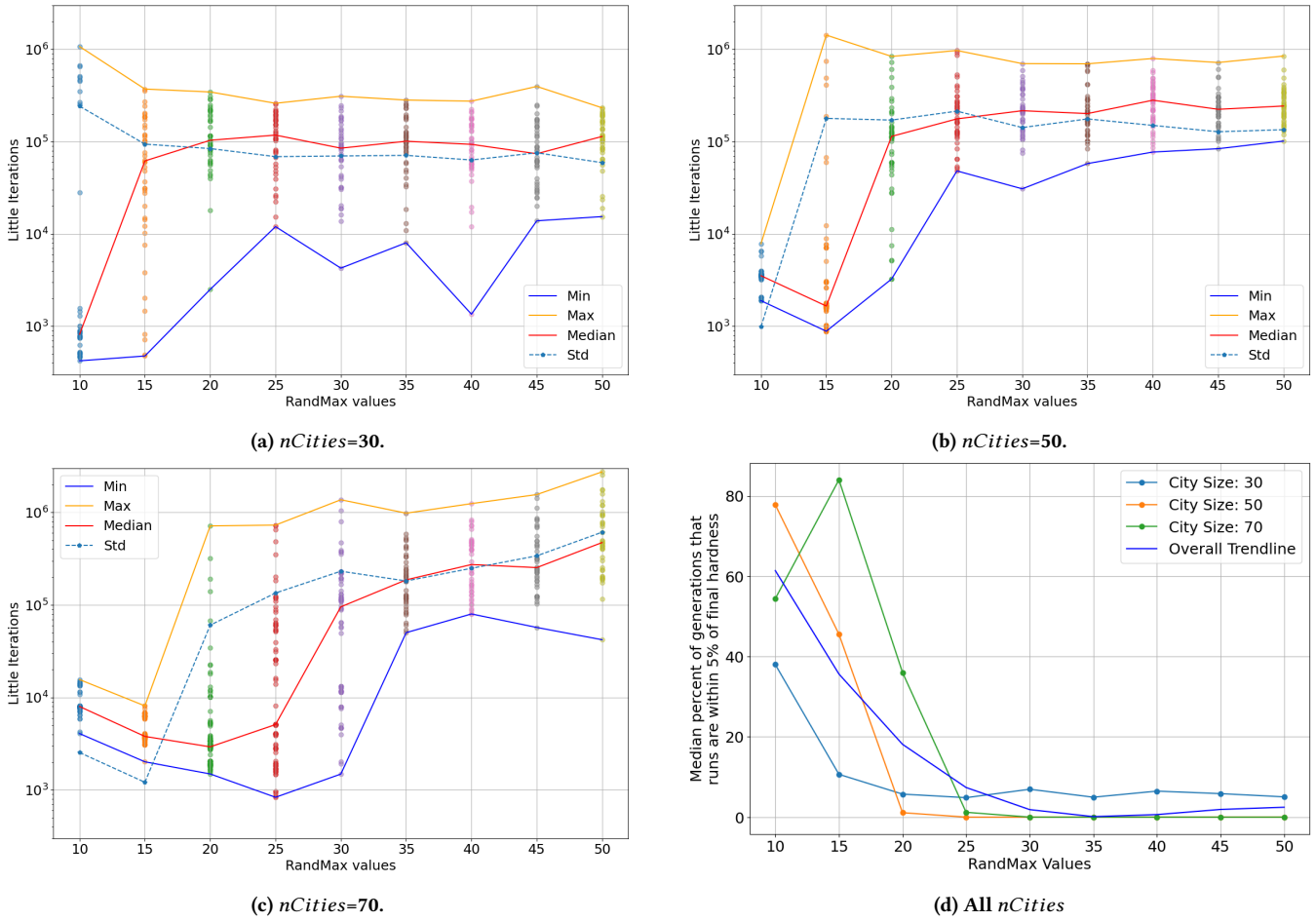
Integer distance matrices are investigated with $nCities$ = 30, 50, and 70, each with values between 1 and a $rand_{max}$ of 10 through 50, exclusive, with intervals of 5. So, 27 combinations of parameters in total. High performance computing (HPC) is used to hill-climb many distance matrices at the same time. Each ATSP parameter combination is run 43 times. Each run is capped by a real-world time limit of five days and a maximum generation limit of 4,000. This means that the achieved hardness of the runs sets a lower bound on the hardness that can be achieved using this method.

3.2 Implementation

This study's methodology for generating the matrices was based on that of Smith-Miles & van Hemert, while contrasting against it through some important changes. Firstly, this study adopts a hill-climbing instead of evolutionary approach to increasing hardness. It then uses the number of iterations required for the algorithm to yield a solution as an additional definition for hardness. Computation time always depends on hardware, while iterations are a more reproducible metric. Little's algorithm is used as the solver, instead of a heuristic algorithm. This change is quite crucial, as using an exact algorithm guarantees that the relationship between characteristic and hardness is done given optimal solutions.

The experiment begins by generating a random distance matrix. This matrix is solved using Little's algorithm, and both the matrix and the number of iterations required for the solution are recorded. Subsequently, a cell within the matrix is randomly mutated, and the matrix is solved anew. If the newly mutated matrix proves to be as hard or harder than the previous matrix, it is then used as the basis for further mutations in subsequent generations. Thus, each generation of matrices derives from the most challenging matrix found in the prior generations, ensuring a progressive increase in problem difficulty. This iterative process of mutation and selection is graphically summarized in figure 5.

The parameters of this study's methodology lie somewhere in between three previous studies [19][18][17]. Investigating matrices with 30, 50, and 70 cities, as opposed to 16-100 cities by the baselines. Going higher than the chosen values risks wasting limited computing access of the supercomputer as instances become harder super-polynomially given the usage of an exact algorithm. Moving on to the next parameter, this study investigates the set of $rand_{max}$ values from 10 to 50 with intervals of 5, as opposed to between $1 - 10^7$, not specifying value ranges, and ranges according to a log-normal distribution with a given standard deviation between 0 and 5. Finally, this study investigates the hardest instances at every



(a) $nCities$**=30.**

(b) $nCities$**=50.**

(c) $nCities$**=70.**

(d) All $nCities$

Figure 6: Min, max, mean, median and standard deviation of final hardness values in each run, plotted for each city size. The hardness in Little's iterations is plotted against the investigated $rand_{max}$ **values. (d) shows the median percentage of generations a run's hardness was within 5% its final hardness for each** $rand_{max}$ **value, graphed for each city. Trendline in blue.**

generation out of 4,000 generations per ATSP formulation, as opposed to not hill-climbing at all, and 600 evolutionary generations in Smith-Miles & van Hemert.

3.3 Little's Algorithm

The Python implementation of Little's algorithm, as introduced in the study by Sleegers et al., is used to measure hardness [17]. Little's algorithm, known for its exact, deterministic approach to solving TSP, operates by iterating over a distance matrix using the branch-and-bound technique. This algorithm was selected over heuristic methods to ensure the accuracy and reproducibility of hardness measurements across different problem instances.

The algorithm begins with a reduction phase, where each row and column of the distance matrix is adjusted to ensure that every row and column contains at least one zero. This is achieved by subtracting the minimum value in each row from all elements of that row, followed by a similar operation on each column. This process simplifies the matrix and establishes a lower bound on the total cost of any tour.

Following the reduction, the algorithm identifies the optimal branch for further exploration. This is done by calculating a penalty value for each zero cell, which is the sum of the smallest values in the corresponding row and column, excluding the zero itself. The zero cell with the highest penalty is chosen for branching, as this selection minimizes the increase in the lower bound when excluding that path.

The algorithm recursively branches by including and excluding the selected zero cell, updating the matrix and lower bound accordingly. For the inclusive branch, the selected path is added to the tour, and the corresponding row and column are eliminated from the matrix. For the excluded branch, the zero cell is prohibited, and its penalty is added to the lower bound. This dual-branch approach ensures comprehensive exploration of potential tours while efficiently narrowing down the solution space.

The algorithm iterates through these steps, progressively refining the tour and its cost, until all potential tours have been evaluated. The process terminates when the lowest possible tour cost is identified, ensuring an exact solution to the TSP instance.

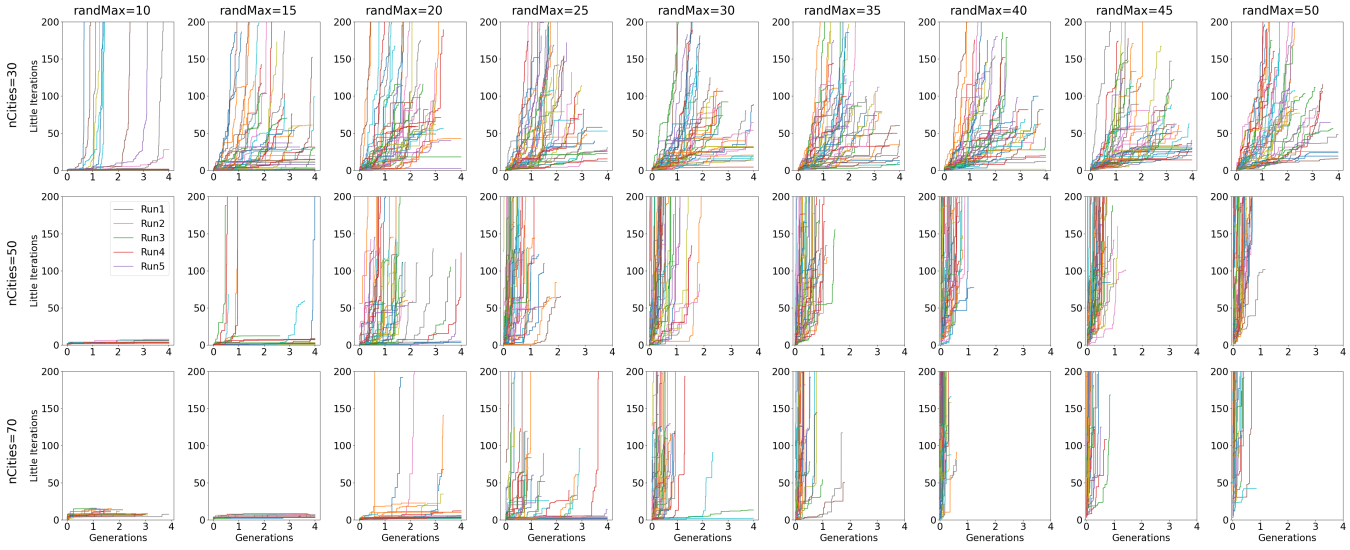


Figure 7: All hill-climbed runs graphed against one another for each *nCities* and $rand_{max}$. Axes are ticked per 1,000 units.

The Python implementation replicates the algorithm's steps using libraries such as NumPy for matrix manipulations and data handling. The use of high-level Python constructs and efficient data structures ensures that the implementation is both clear and performant. The source code of the algorithm is publicly available [16].

3.4 Analysis

The average median hardness over the $rand_{max}$ is calculated for all *nCities* to determine the influence of the hill-climber. Then, a sigmoid function is fitted to the median hardness per $rand_{max}$ of both the begin and end matrices per *nCities* to determine the existence of a phase transition. The fit is characterized by its midpoint, steepness, R-squared value, and Coefficient of Variation of the Root Mean Squared Error (CVRMSE). The CVRMSE is taken instead of the regular RMSE because this allows for comparison between fits even though the absolute hardness values are different.

Next, the extent to which runs tend to converge is assessed. The percentage of generations a run is within a 5% margin of its final hardness is calculated. This is set to zero if the hardness was hill-climbed for less than 1,000 generations. This measure is taken from every of the 43 runs and the median from this distribution is plotted against $rand_{max}$ values for each *nCities*. A trendline over all lines is graphed as well. This analysis identifies how quickly the hill-climbing algorithm stabilizes around a maximally hard matrix, offering insights into the efficiency of the hill-climber in exploring the problem space. This measure gets skewed when there are few generations because this allows for a very small absolute value of 'converging' values to count as a high relative convergence. However, with such little data convergence can never be concluded. So, in inspecting the graph, one should also consult the first to see how many generations were actually completed.

4 RESULTS

Figure 6 reports how the minimum, maximum, standard deviation, and median of the highest hardness acquired in each run change with increasing $rand_{max}$, with a sub-graph per *nCities*. Each run is represented as a point on their respective $rand_{max}$. Five lines are drawn indicating per $rand_{max}$ the minimum in blue, maximum in yellow, median in red, and standard deviation of hardness in dotted sky-blue.

Figure 6d reports the evolution of the median percentage of generations that runs' hardness are within a 5% margin of the final hardness, over the $rand_{max}$ values for each city. This measure, called 'convergence', gives an indication of how many generations the runs have stopped significantly increasing in hardness, possibly finding a near-optimally hard matrix for that parameter combination. The higher the value, the faster the runs tend to converge.

Figure 7 reports the highest hardness at every generation of all runs in the experiment. Visualized as tiled sub-graphs in a three by nine matrix. Three columns labelled on the left for the *nCities* values and nine columns labelled at the top for the $rand_{max}$ values. Each colored line represents a different run of the same parameter combination. Axes are ticked per 1,000 units. The maximum allowed wall clock time of each run was five days. The vast majority of runs from $rand_{max}$ 25 did not reach the 4,000 generation limit in that time. Figure 8 reports for each *nCities* the percentage increase in median hardness at each $rand_{max}$ due to hill-climbing.

Figure 9 reports another single run, in addition to Figure 4. This run contains the greatest relative difference in hardness due to one mutation across all runs. The calculations were done by dividing the absolute difference by the lowest number, times 100. The hardness drops from 452,624 to 414 iterations, achieving a change in hardness of over 1,092 times.

Figure 10 reports the frequency of highest hardness values acquired in each of the runs for *nCities*=50 and 70. They were created after the plots above, dividing each of the *nCities* into two $rand_{max}$

$nCities$	$rand_{max}$	Steepness	r^2	CVRMSE
30	**14.8**	**3.2**	**0.86**	**15.2**
30	14.95	0.17	0.87	11.6
50	**21.19**	**0.33**	**0.95**	**13.64**
50	32.02	0.38	0.90	26.33
70	**46.86**	**0.12**	**0.94**	**27.08**
70	50.20	0.19	1.00	8.06

Table 1: Fit of Sigmoid Function on Different Cities with (bold) and without Hill-Climbing. Second column indicates the $rand_{max}$ on which the midpoint of the sigmoid lies. Hill-climbed fit in bold.

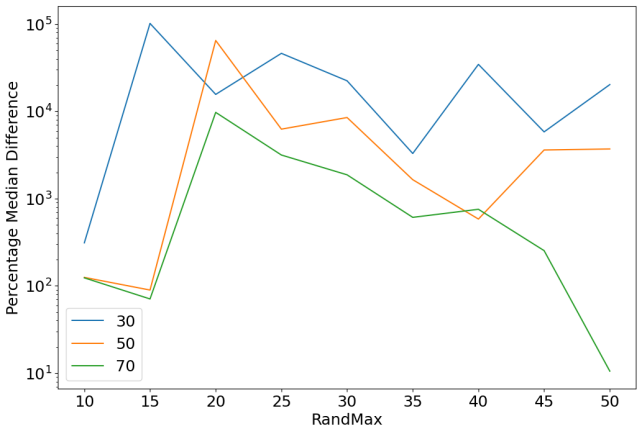


Figure 8: Percentage increase in median hardness at each $rand_{max}$ due to hill-climbing for each $nCities$.

ranges based on where a phase transition was observed. The left plot shows before and the right plot after the phase transition. The closer a square is to white, the more values lie within it. For $nCities$=70, the highest relative change was taken as the point of the phase transition, as observed in figure 6c, instead of the one calculated in table 1.

Figure 11 reports the evolution of median hardness of non-hill-climbed matrices over the standard deviation of the distance matrices. Being very similar to the experiment of Sleegers et al., their results are superimposed are on the graph.

4.1 Analysis

The results suggest hill-climbing to be a successful method of increasing hardness within and over parameter combinations. The average median hardness over the $rand_{max}$ grew for all $nCities$, with 79,125, 162,184, and 87,959 iterations respectively, which are increases of 27,806%, 9,931%, and 1,844%. The increase in hardness generally occurred at the $rand_{max}$ of the phase transition and after. Figure 11 shows how most patterns discussed so far become much less clear without hill-climbing.

Results suggest the existence of phase transitions in hill-climbed instances as described by Zhang & Korf, summarized in table 1. The phase transitions take place at different $rand_{max}$ for different $nCities$. For 30 cities, the hill-climbed condition shows a significantly steeper curve (3.2 vs. 0.17), which indicates a more pronounced phase transition. Both conditions have a similar r^2 value, indicating good fits, but the non-hill-climbed condition has a lower CVRMSE, suggesting better predictive accuracy. For the next city size the hill-climbed condition has a slightly lower steepness but a higher r^2, indicating a better fit to the data. The CVRMSE is also lower in the hill-climbed condition, suggesting more accurate predictions compared to the non-hill-climbed condition.

Finally, in the case of 70 cities, the non-hill-climbed condition shows a perfect r^2 value of 1.00, indicating a perfect fit, which is unusual. The CVRMSE is much lower in the non-hill-climbed condition, indicating better predictive accuracy. However, the lower steepness in the hill-climbed condition suggests a less pronounced phase transition. The lower the $nCities$, the lower the $rand_{max}$ at which the phase transition occurs.

Figure 8 reveals that the midpoint of the sigmoid curve coincides with the $rand_{max}$ at which the hill-climber had most effect except for $nCities$=70. From this point onwards, the hill-climber had orders of magnitude more effect on the hardness than before. Creating matrices that have a median hardness at least 1,018 times higher than before hill-climbing.

For every $nCities$, the standard deviation of the hardness between runs, in figure 6, starts growing right at the start of the phase transition. Before the standard deviation growth, the graph and its min and max seem like a narrow tunnel. But as the standard deviation grows, it is like the graph's min and max values bulge out. The phase transition finds itself at the center of this bulge. After that, the standard deviation stabilizes together with the hardness, and the min and max contract again to a more narrow tunnel. This effect is most visible for $nCities$=50 and 70. For $nCities$=30, the highest standard deviation is observed at the smallest $rand_{max}$, suggesting that the bulge lies before that point. For $nCities$=70 the standard deviation only achieves its highest value at the highest $rand_{max}$, possibly suggesting a continuation of its phase transition beyond the investigated $rand_{max}$.

Consulting figure 10 reveals how much the highest achieved hardness changes before and after the phase transition. The growth on the vertical axis indicates the rising hardness in iterations. In addition, the decreasing generation count means computational cost exceeded available computation time. Substantiating in another way that that the transition made the problem much harder. This also means that the reported hardness values only indicate a lower bound of achievable hardness for the matrices via hill-climbing. The lower the generation count, the higher the yet undiscovered hardness.

The same figure also reveals how the hill-climber tends to converge the runs to a certain level of hardness before the phase transition. However, this is in part due to the scale of the figure. For example most values for $nCities$=50 range anywhere between 900 and 10,000 iterations, with some outliers even reaching over 1.5 million iterations. So the hill-climber does not seem to drive the matrices towards the same optimally hard matrix within a parameter combination. Convergence does happen within runs, when the hill-climber barely improves a matrix over many generations:

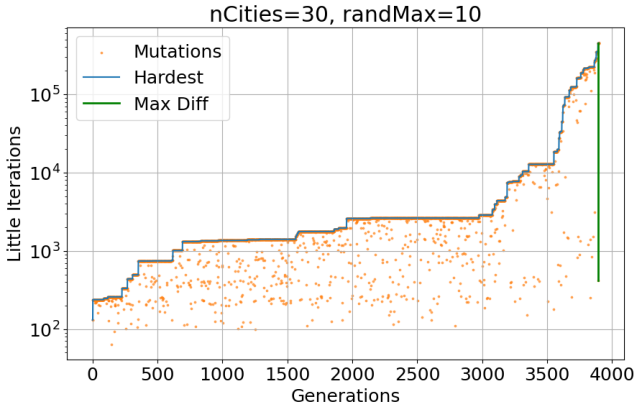


Figure 9: A single run of hill-climbing $nCities$=30, $rand_{max}$=10. In green, the greatest relative difference in hardness due to one mutation recorded across all runs. In orange, every hardness measured is chronologically plotted over the generations, with the hardest matrix found at that generation in blue.

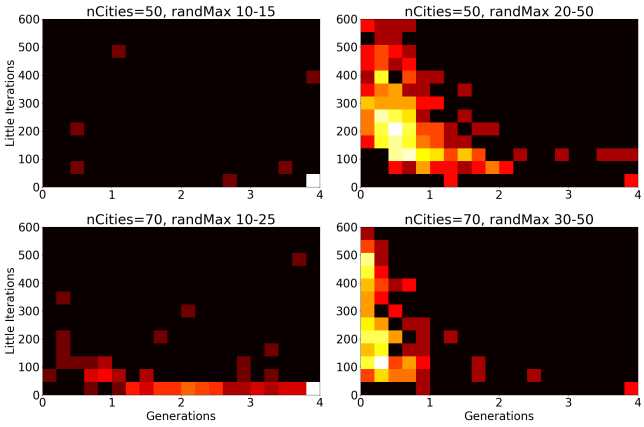


Figure 10: The frequency of highest hardness values acquired in each of the runs for $nCities$=50 and 70, for two $rand_{max}$ ranges, before and after the phase transition. The closer a square is to white, the more values lie within it. Axes are ticked per 1,000 units.

This characteristic seems to follow a negative logarithmic relationship to $rand_{max}$ in figure 6d. The sharp decline in convergence coincides with a rise in standard deviation of hardest hardness in the surrounding figures. For $nCities$=30, convergence seems to stabilize on 7-8%. However, there is not enough data on $nCities$=50 and 70 to tell how this relationship develops into these later $rand_{max}$ values. Consulting figure 7 reveals that almost all of the runs with this $nCities$ did not reach 1,000 generations beyond a $rand_{max}$ of 20, resulting in an assignment of a convergence of 0. Keep in mind that the research of Ochoa & Veerapen also considered a type of convergence, but only after 10,000 iterations without any improvement.

5 DISCUSSION

Hill-climbing ATSP distance matrices is a viable method to generate harder and harder problems for Little's algorithm to solve. The hill-climber increased the overall hardness of every city size, making matrices for a set city size at least 278 times harder. It's effect was the largest on $nCities$=30, $rand_{max}$=15, making the median hardness of the parameter combination at least 1,018 times harder. It also made the phase transitions occur at earlier $rand_{max}$ while maintaining a good r^2 value. Some results were specific per city size, like the drastic increase in steepness of the fitted sigmoid curve for 30 cities, and the significant improvement in both r^2 and CVRMSE for 50 cities.

Hill-climbing the distance matrices could be understood in terms of the optimization landscapes of Ochoa & Veerapen: As a search for the hardest landscape possible given the parameter combination and initial conditions. Each time a matrix is mutated, the landscape changes, and funnels are created and moved around. Significantly so, as figure 9 shows that changing one number in a 30 city matrix can drop the hardness by at least three orders of magnitude. Furthermore, two runs with the same parameter combination were also observed to differ from one another significantly. Differences between their final hardness of up to at least four orders of magnitude were recorded. Interesting to note is that both of these observations were done on $nCities$=30, $rand_{max}$=10, which one could have assumed as being the easiest parameter combination investigated.

These results suggest that the hard instances are well-mixed with the easier instances with respect to their $rand_{max}$, prohibiting them from being separated by their $rand_{max}$. Still, the global maximum hardness in a parameter combination seems to be inaccessible for a hill-climber from many initial conditions, as shown by the convergence. It seems, for example, that the lower $nCities$=30, $rand_{max}$=10 could never reach the same hardness as the hardest matrices in the parameter combination, because for the last 40% of generations their hardness did not change significantly. Suggesting that despite the well-mixed, irregular landscape, there are still local optima abound. This makes it challenging to generalize solutions across different instances and to design algorithms that are generally effective.

5.1 Comparison to Previous Studies

The overall increase in hardness generally occurred at the $rand_{max}$ of the phase transition and after. This suggests that characterizing phase transitions using this method, in addition to fitting a sigmoid curve, could be highly effective. It is possible that at this critical $rand_{max}$ value, the hill-climbed instances are leveraging more of the inherent complexity described by Zhang & Korf.

Phase transitions take place at different $rand_{max}$ for different $nCities$. The results suggest that the lower the $nCities$, the lower the $rand_{max}$ at which the phase transition occurs. These patterns are consistent with Zhang & Korf's results, as their phase transitions happen much later in their 100 city experiment than in the 50 city experiment in this thesis. They also found complexity phase transitions at $rand_{max}$ beyond the scope of this study. Future work could expand the search, looking for extended phase transitions at higher $rand_{max}$. Especially for $nCities$=70 and up, it seems necessary to increase the $rand_{max}$ to capture the phase transition fully.

Figure 11 reveals a pattern for the non-hill-climbed matrices beyond the scope of Sleegers et al., who reported no relationship between standard deviation and Little iterations, with 48 cities instead of 50. An explanation for it could be that Sleegers et al. would have found an increase in hardness if they would have investigated more standard deviations.

Another difference between Sleegers et al. on the one hand, and Zhang & Korf and this study on the other, is that the latter two used integer-only distance matrices, while the former allowed for floating-point values. This theoretically is a significant change, the ability to represent values more precisely means that the distance matrix can capture finer gradations in distances between cities.

5.2 Limitations

The use of Little's algorithm introduces a computational load that limits the scalability of the study. This algorithm, being exact, requires significant computational resources, especially as the hardness increases. Consequently, the findings of the study are somewhat constrained by the practical limits of computational power available for solving large instances of ATSP.

5.3 Future Work

If the phase transitions are due to the matrices being integer-only, there are implications for the design and evaluation of algorithms targeting the ATSP. The results show a significant drop in hardness in hill-climbed low $rand_{max}$ integer distance matrices. Suggesting that performance benchmarks should account for these critical $rand_{max}$ values to better understand algorithmic efficiency and scalability. If this evolution is different for floating-point matrices, this could have implications for algorithm design in the optimization field. For example, rounding values in the distance matrix to a certain degree could offer a performance boost.

An intriguing extension of this research would be to construct Local Optima Networks (LONs) as done in Ochoa & Veerapen to visualize the evolution towards maximally hard integer problem instances. The authors responded to this idea describing that this involves defining local optima through a hill-climbing algorithm and establishing edges via a perturbation operator. Despite the computational intensity noted by another expert, who highlighted the significant number of fitness evaluations required, leveraging high-performance computing resources could make this feasible. Alternatively, search trajectory networks (STNs) might offer a less computationally expensive method while still providing valuable insights into search dynamics. This approach could elucidate whether a coarse-level gradient exists towards harder instances, paralleling the methodology used in previous studies to understand easier problem landscapes. Integrating LONs or STNs into this subject would not only deepen our understanding of ATSP hardness but also contribute to the broader field of optimization landscapes by revealing structural complexities in the pursuit of maximal problem difficulty.

6 CONCLUSION

This study aimed to deepen the understanding of what makes instances of the Asymmetric Traveling Salesman Problem (ATSP) hard. Particularly in relation to the parameters of the distance matrix defined by city size, $nCities$, and the maximum value a randomly generated integer may take on between 1 and a $rand_{max}$. Through the application of hill-climbing techniques and Little's exact algorithm, the study successfully generated progressively harder ATSP instances and analyzed the resulting data to uncover trends and insights about problem hardness.

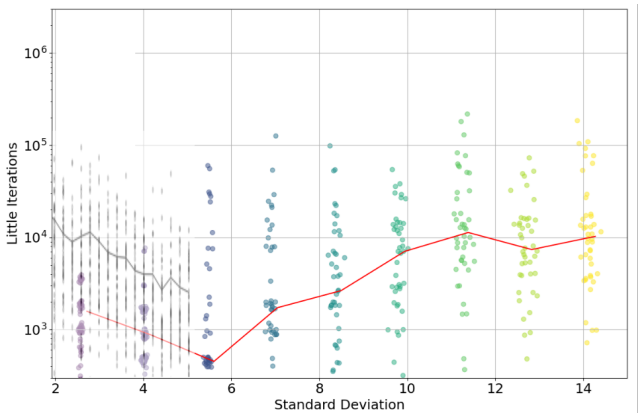


Figure 11: Non-hill-climbed matrices with $nCities$=50, standard deviation of values in the distance matrix on the horizontal axis. Median in red. Results of Sleegers et al. superimposed in grey.

The study finds that hill-climbing ATSP distance matrices is a viable method to generate harder and harder problems for Little's algorithm to solve. The hill-climber increased the hardness of every matrix investigated, up to a median increase within a parameter combination of at least 1,018 times. The results suggest the hill-climber is effective because ATSP's space of problem hardness can be highly irregular, with some small changes leading to shifts in hardness of over three orders of magnitude. However, the hill-climber does seem to get stuck in local optima. Phase transitions in hardness are found at critical $rand_{max}$ values. They occur at earlier $rand_{max}$ for smaller city sizes, and even earlier when hill-climbed. The hill-climber had the largest effect after the phase transition. These observation are consistent with work of Fischer et al.[6], Sazhinov et al.[15], Sleegers et al.[17], Smith-Miles & van Hemert[18], and Zhang & Korf[19].

7 ACKNOWLEDGEMENTS

I am grateful to Daan van den Berg for his outstanding mentorship, which has significantly enhanced my academic performance and reshaped my understanding of university education. I look forward to continuing our collaboration. I also extend my appreciation to our research group members, whose support and insights have been invaluable in advancing this research.

REFERENCES

[1] Andi (2018). Clustering Distance Measures - Datanovia.
[2] Bai, J., Yang, G., Chen, Y., Hu, L., and Pan, C. (2013). A model induced max-min ant colony optimization for asymmetric traveling salesman problem. *Appl. Soft Comput.*, 13:1365–1375.
[3] Ben-Dor, A., Chor, B., and Pelleg, D. (2000). Rho—radiation hybrid ordering. *Genome Research*, 10(3):365–378.

[4] Bland, R. and Shallcross, D. (1987). Large traveling salesman problem arising from experiments in x-ray crystallography. *a preliminary report on computation*, (730).

[5] Cheeseman, P. C., Kanefsky, B., and Taylor, W. M. (1991). Where the really hard problems are. *Ijcai*, 91:331–337.

[6] Fischer, T., Stützle, T., Hoos, H., and Merz, P. (2005). An analysis of the hardness of tsp instances for two high performance algorithms. *Proceedings of the Sixth Metaheuristics International Conference*, pages 361–367.

[7] Goldberg, D. (1989). Genetic algorithms in search, optimization and machine learning. addison-wesley longman publishing co., inc.

[8] Goodrich, M. T. and Tamassia, R. (2015). *Algorithm design and applications*, volume 363. Wiley Hoboken.

[9] Hoffman, K. L., Padberg, M., Rinaldi, G., et al. (2013). Traveling salesman problem. *Encyclopedia of operations research and management science*, 1:1573–1578.

[10] Little, J. D., Murty, K. G., Sweeney, D. W., and Karel, C. (1963). An algorithm for the traveling salesman problem. *Operations Research*, 11(6):972–989.

[11] Macready, W. and Wolpert, D. (2010). What makes an optimization problem hard. *Complexity*, 5:40–46.

[12] Ochoa, G. and Veerapen, N. (2018). Mapping the global structure of tsp fitness landscapes. *Journal of Heuristics*, 24.

[13] Orponen, P. and Mannila, H. (1987). *On approximation preserving reductions: Complete problems and robust measures*. University of Helsinki.

[14] Pearl, J. (1984). *Heuristics: intelligent search strategies for computer problem solving*. Addison-Wesley Longman Publishing Co., Inc.

[15] Sazhinov, N., Horn, R., Adriaans, P., and van den Berg, D. (2023). The partition problem, and how the distribution of input bits affects the solving process.

[16] Sleegers, J. (2019). Source code of the implementation. http://heuristieken.nl/wiki/index.php?title=CheesemanTSPReplication.

[17] Sleegers, J., Olij, R., van Horn, G., and van den Berg, D. (2020). Where the really hard problems aren't. *Operations Research Perspectives*, 7:100160.

[18] Smith-Miles, K., Van Hemert, J., and Lim, X. Y. (2010). Understanding tsp difficulty by learning from evolved instances. *Learning and Intelligent Optimization: 4th International Conference, LION 4*, 4:266–280.

[19] Zhang, W. and Korf, R. E. (1996). A study of complexity transitions on the asymmetric traveling salesman problem. *Artificial Intelligence*, 81(1-2):223–239.

Appendix A FIRST APPENDIX

Put your appendices here.