Clearing the Combinatorial Fog: Tracing the Hidden Paths of TSP Heuristics

Jonathan Heins
"Friedrich List" Faculty of Transport and Traffic Sciences, TU Dresden & ScaDS.AI Dresden/Leipzig
Dresden, Germany
jonathan.heins@tu-dresden.de

Sebastian Dengel
"Friedrich List" Faculty of Transport and Traffic Sciences, TU Dresden
Dresden, Germany
jonathan.heins@tu-dresden.de

Darrell Whitley
Department of Computer Science, Colorado State University
Fort Collins, CO, USA
whitley@cs.colostate.edu

Pascal Kerschke
"Friedrich List" Faculty of Transport and Traffic Sciences, TU Dresden & ScaDS.AI Dresden/Leipzig
Dresden, Germany
pascal.kerschke@tu-dresden.de

ABSTRACT

Over decades of Traveling Salesperson Problem (TSP) research, powerful heuristics have been developed that efficiently solve many TSP instances. Among them, the local search optimizer LKH and the genetic algorithm EAX stand out as the two complementary state-of-the-art solvers. Yet, the links between instance structures and solver complementarity remain obscure, i.e., it is often unclear how instance structures affect solver performance and behavior.

While visualization techniques have advanced our understanding of TSP instance structures, there is still no general method to assess the behavior of TSP heuristics directly based on the structural characteristics of the instance to be solved. Existing approaches, such as search trajectory networks, are often optimized for graph-theoretic properties that may not reflect true similarities between tours. Consequently, visually comparing solvers remains difficult, especially when few identical solutions are encountered.

This paper introduces a framework for visualizing the search behavior of TSP heuristics, aiming to illuminate the still poorly understood key differences in their search behavior. To this end, we adapt the dimensionality reduction technique PaCMAP to map intermediate solutions into two-dimensional space, preserving distances that meaningfully reflect tour similarity. To support exploration, we provide an interactive dashboard that allows users to inspect individual tours and visually compare selected tour pairs. The resulting visualizations reveal fundamental differences in how LKH and EAX explore the search space, including their initialization strategies and trajectory structures. We further show that a recent EAX-inspired and performance-improved LKH variant still behaves similarly to LKH, highlighting the untapped potential for further algorithmic improvements.

FOGA '25, August 27–29, 2025, Leiden, Netherlands

ACM ISBN 979-8-4007-1859-5/2025/08.
https://doi.org/10.1145/3729878.3746623

CCS CONCEPTS

• **Mathematics of computing** → **Combinatorial algorithms**; **Combinatorial optimization**; *Combinatoric problems*; • **Computing methodologies** → *Discrete space search*; **Genetic algorithms**; • **Theory of computation** → **Evolutionary algorithms**.

KEYWORDS

Combinatorial Optimization, Traveling Salesperson Problem, Search Space Analysis, EAX, LKH, PaCMAP, Exploratory Visualization, Evolutionary Algorithms, Local Search

ACM Reference Format:
Jonathan Heins, Sebastian Dengel, Darrell Whitley, and Pascal Kerschke. 2025. Clearing the Combinatorial Fog: Tracing the Hidden Paths of TSP Heuristics. In *Foundations of Genetic Algorithms XVIII (FOGA '25), August 27–29, 2025, Leiden, Netherlands.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3729878.3746623

1 INTRODUCTION

Few optimization problems are as well-known and versatile as the *Traveling Salesperson Problem (TSP)*. For decades, it has served as an easily accessible introduction to combinatorial optimization in textbooks across a wide range of disciplines, including computer science, logistics, operations research, and transportation science [2]. It also forms the basis for many practically relevant extensions, including the vehicle routing problem (VRP). Yet, despite its long-standing role in research and teaching, our understanding of how specific structural properties of TSP instances affect the behavior, and hence performance, of optimization algorithms and search heuristics remains limited.

The combinatorial optimization problem underlying the TSP is conceptually straightforward. Given a graph $G = (V, E, c)$, consisting of (1) a set of vertices $V := \{v_1, \dots, v_N\}$ (representing nodes or cities), (2) a set of edges $E := \{e_k = (v_i, v_j) \in V \times V\}$ connecting all vertex pairs, and (3) a cost function $c : E \to \mathbb{R}$ assigning a real-valued cost to each edge. Then, the objective is to find a cost-optimal tour, which consists of exactly N edges connecting all N vertices under the constraint that each vertex has one incoming and one outgoing edge without any duplicated edge in the tour.

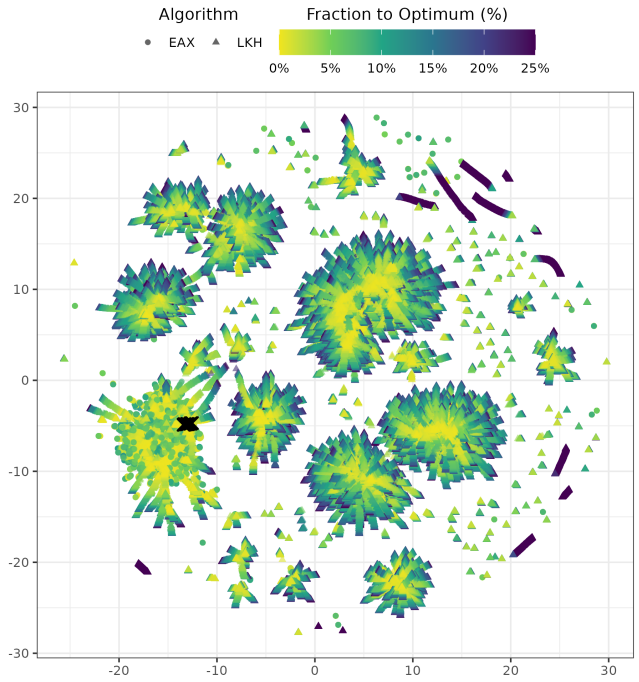

Figure 1: All tours from ten runs of EAX (circles) and LKH (triangles) on TSPLIB instance d1291, represented in a two-dimensional embedding space. The two solvers explore distinct regions of the feasible space, with EAX more concentrated around the region containing the optimal tours (black crosses). Tour colors indicate the relative deviation from the optimal length, expressed as the fraction of additional tour length compared to the best known solution.

Over the years, a wide range of solution approaches for the TSP have been developed, broadly classified into two categories: exact algorithms and heuristic solvers. Exact algorithms, if successfully terminated, guarantee the optimality of their found solution. However, they are often computationally much slower and even fail to terminate within a reasonable timeframe when processing challenging instances. In contrast, heuristic methods typically produce high-quality solutions much faster, though without any guarantee of optimality. Over the past two decades, EAX [24, 25] and LKH [14, 15] have emerged as state-of-the-art heuristic TSP solvers. Both heuristics, including their various refinements proposed in recent years [7, 10, 11, 34], have demonstrated strong performance across a variety of experimental studies. Noticeably, both strategies regularly demonstrated complementary performance, which might be caused by their fundamentally distinct concepts: EAX is a highly sophisticated genetic algorithm, while LKH at its core and in the default settings represents a finely tuned local search variant.

To date, the idea that the conceptual differences between EAX and LKH drive their performance variations remains speculative. Several studies have attempted to investigate these differences from various angles. For example, [3] evolved TSP instances that were particularly easy for EAX and difficult for LKH (and vice versa), and then analyzed the produced instances for structural patterns that might explain the observed performance gaps – without a clear finding. In fact, recent works have shown that the 'difficulties' produced during the evolution process are particularly specific to the versions of EAX and LKH that were considered back then. The recently proposed variants of both algorithms showed that they handle these previously challenging instances much more effectively [10, 11]. Another line of research approached the problem by studying instance features, i.e., numeric values that characterize structural properties of the underlying TSP instances and which might be used to group similar TSP instances [8, 9]. These features served as input for algorithm selection models, in which machine learning models aim to predict the most promising solver for a given instance [16, 17]. However, a common finding across these studies is that the models mostly rely on the algorithm with superior average performance (typically EAX) and focus primarily on identifying instances where that algorithm is substantially outperformed by its competitor(s). As a result, these models rarely learn deeper, more nuanced patterns that could explain performance variations in a more general and interpretable way. For very small instances, the search behavior of algorithms can be analyzed in more detail [19]; however, these instance sizes do not typically induce large performance differences and are not representative of the scenarios for which the algorithms are designed.

An alternative approach to gaining deeper insights into the structural characteristics of TSP instances, which might help to explain algorithmic search behavior, is through visualization. For example, [29] utilize local optima networks (LONs) to depict a TSP instance based on its local and global optima. These visualizations help identify local optima and their corresponding basins of attraction, which might trap an algorithm during its search process. However, while LONs effectively capture the structural properties of problem instances, they offer limited insight into the dynamics of algorithmic search behavior. As a result, they are not well suited for directly comparing differences between algorithmic searches. In the context of continuous optimization, [23] proposed a visualization technique that projects all instances of a test suite into a two-dimensional space. Each instance is then color-coded according to the algorithm that performs best on it, effectively revealing the algorithmic 'footprint' across the entire suite. While this method offers valuable insights at the suite level, it does not capture or illustrate differences in the search behavior of algorithms on the level of individual instances.

In this work, we aim to address the outlined research gap by introducing a visualization tool that maps algorithmic search behavior into a two-dimensional space using the dimensional reduction method Pairwise Controlled Manifold Approximation (PaCMAP) [37]. A result of such a projection can be seen in Figure 1. Our proposed tool enables

(a) the visualization of exploration and exploitation phases within an algorithm's search process,
(b) the identification of basins of attraction in the studied TSP instance, and
(c) the direct comparison of search behavior across multiple search heuristics.

While our tool is, in principle, applicable to a wide class of optimization problems, we focus here on the TSP to showcase its functionality.

The remainder of this paper is structured as follows. Section 2 reviews existing visualization techniques. Section 3 introduces the dimensionality reduction method employed, along with the algorithms under study and their relevant characteristics, which help explain the patterns observed in later visualizations. The experimental setup and procedure are described in Section 4. Based on the collected data, an interactive dashboard is developed in Section 5 to showcase the visualization results. Section 6 discusses the key patterns discovered, and Section 7 concludes the paper.

2 RELATED WORK

Local Optima Networks (LONs) [29] have proven to be valuable tools for analyzing optimization problem instances. They have been adapted to a wide range of problems, including neural network architecture search [31], the TSP [30], and others [32, 36]. The idea of representing optimization problems as networks was originally inspired by energy landscape networks [5, 6], where local energy optima are connected by edges if the transition barrier between them is sufficiently small.

LONs typically employ an iterated local search heuristic with multiple random initializations. A local optimum is defined as a solution that cannot be improved further by the local search. In the TSP context, the method perturbs (or "kicks") a found local optimum to generate a new solution, which is then subjected again to local search. If this leads to a new local optimum, both the original and resulting optima are recorded and connected by a directed edge. This process continues until either the global optimum is found or no new optima can be discovered from a given initialization. The procedure is then restarted with a new initial solution.

The resulting local optima and their transitions are represented as a graph – the local optima network – where nodes correspond to local optima and directed edges represent transitions via kicks. The layout of the graph is then optimized according to aesthetic and structural criteria such as evenly distributed nodes, minimal edge crossings, uniform edge lengths, and visually aligning similar regions. While these layout strategies produce interpretable diagrams, they rely only on partial adjacency information and encode limited distance relationships between solutions. As a result, the global structure of the search space is only partially captured.

Search Trajectory Networks (STNs) [27, 28] build upon the concepts introduced by LONs, but shift the focus from characterizing the problem landscape to examining the dynamic behavior of search algorithms. STNs have been adapted for multi-objective evolutionary algorithms [18] and multi-objective combinatorial optimization problems [26]. Unlike LONs, STNs represent not only local optima but also intermediate (and potentially aggregated) solutions encountered during the search process.

Edges in STNs correspond to transitions from one iteration of the algorithm to the next. This representation allows to visualize regions where algorithms stall at local optima, and to identify shared solutions or paths discovered by multiple algorithms. However, when solutions differ, STNs fall short of revealing whether two algorithms are traversing similar regions of the search space or entirely distinct ones. Moreover, when multiple attractive local optima exist, it may not be possible to see whether they belong to close basins of attraction or represent structurally different areas of the landscape. To address these limitations and to visualize solution similarities more comprehensively, we propose a dimensionality reduction-based approach.

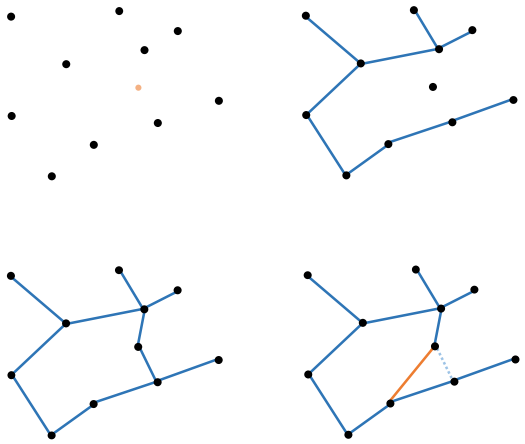

Figure 2: Top left: Select a special node (orange) from the set of points. Top right: Compute the MST on the remaining nodes. Bottom left: Connect the special node to its two nearest neighbors. Bottom right: To compute an edge's α-value (orange), subtract the length of the longest removable edge (blue dashed) that maintains the 1-tree structure.

3 METHODOLOGY

To interpret the visual patterns and validate their meaning, we first introduce the two heuristics under study as well as PaCMAP, the dimensionality reduction technique used. A solid understanding of both the algorithmic behavior and the embedding process is necessary to explain the structures revealed by the visualizations.

3.1 LKH

LKH (Lin-Kernighan-Helsgaun) [14, 15, 34] is Helsgaun's versatile and efficient advancement of the classical Lin-Kernighan (LK) heuristic [20]. The original LK heuristic builds upon the well-known 2-opt move [4], which removes two edges from a tour and reconnects the two resulting paths with two different, shorter edges to form a new valid tour. In the Euclidean TSP — where the triangle inequality holds — repeated application of 2-opt moves guarantees tours without crossing edges, which would otherwise indicate an improvable structure [4]. The LK heuristic generalizes this idea by constructing k-opt moves, where k edges are removed and replaced with k new edges to form an improved tour. Rather than exhaustively exploring arbitrary k-opt combinations, LK chains together sequential 2-opt moves: each deleted edge must be incident to an added edge, and vice versa. This is akin to an AB-cycle structure but without a second parent tour. It is important to note that this chaining strategy does not guarantee finding all possible improvements, as general non-sequential k-opt moves are not considered, i.e., moves that swap any k edges that do not have to be chained.

A general n-opt move could, in principle, transform any solution into the global optimum in a single step, but the computational cost would be prohibitive.

Two key design choices make the LK heuristic highly efficient:

(1) *Gain criterion*: Only sequences of moves that are likely to result in an overall gain are explored. This is based on the observation that if the sum of a sequence of N real numbers x_i is positive, there exists a cyclic permutation of the sequence with $\forall M \leq N, \sum_{i=0}^{M} x_i > 0$ [20]. If there are negative numbers in the sequence consider the ith number until which the sum is lowest. All subsequences of numbers until the end of the complete sequence must fulfill the non-negative sum property since otherwise, the ith number would not be the one until which the sum is lowest. Chaining the beginning of the sequence after this subsequence does not break the property since the overall sum must be positive. In the context of move sequences, this means that if the gain at any point becomes negative, that sequence can be pruned without considering further extensions since any potential beneficial move can be found from another starting point.

(2) *Candidate sets*: Rather than considering all possible edges for reconnection, LK limits attention to a candidate set – typically, the five nearest neighbors for each node. While this significantly reduces the search space, it carries the risk of excluding critical edges that could be part of the optimal tour. If this happens, LK might struggle to find the optimal tour at all as there are no other occasions at which new edges are introduced besides the closing of a k-opt move which might not be sufficient.

Building on the LK framework, Helsgaun introduced several significant improvements in the Lin-Kernighan-Helsgaun heuristic [14]. Most notably, the candidate set construction was redesigned based on the concept of minimum 1-trees [12, 13]. A minimum 1-tree is formed by computing a minimum spanning tree (MST) over all but one node, and then connecting the excluded node to the MST via its two closest neighbors. See Figure 2 for a visualization. If each node in the 1-tree had exactly two incident edges, it would constitute a Hamiltonian cycle – the solution to the TSP. Candidate edges in LKH are ranked based on their α-values, which measure the additional cost of including an edge into the minimum 1-tree. If an edge already belongs to the minimum 1-tree, its α-value is zero. Otherwise, the α-value is the cost difference between adding the edge and removing the longest edge necessary to maintain the 1-tree structure. Helsgaun showed that taking the edges with the lowest α-value instead of those with the smallest distances decreases on average the rank in terms of order in the candidate set of optimal edges. Therefore, the required edges are more likely to be considered during the search even with smaller candidate sets.

To further refine candidate selection, LKH employs subgradient optimization [12, 13]. Subgradient descent iteratively transforms the distance matrix to bring the minimum 1-tree structure closer to a Hamiltonian cycle, while preserving the optimal tour. Specifically: Distances to nodes with more than two incident edges in the current 1-tree are increased and distances to nodes with only one incident edge are decreased by a fixed value. Because the cost of any tour is shifted uniformly at each node, the identity of the optimal tour remains unchanged, while the structure of the 1-tree is nudged toward a cycle since a node that is closer to all other nodes will in tendency have more connections in the MST and fewer if it is farther away. If the subgradient procedure eventually transforms the minimum 1-tree into a Hamiltonian cycle, the optimal tour is found. Typically, however, it merely improves the proximity between the 1-tree and the desired cycle, leading to better candidate edges during the subsequent LK search [14].

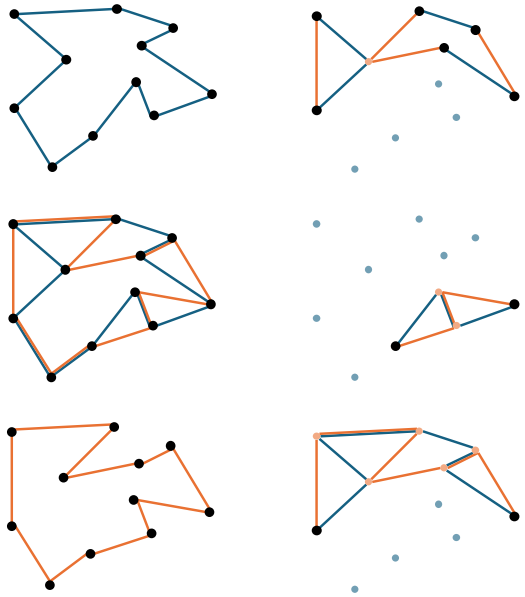

Figure 3: Left: Parent A (blue, top) and Parent B (orange, bottom) form a union graph, with all edges (middle). Right: Examples of different E-sets. Points are distinguished into A- (blue), B- (red), and C-nodes (black).

In the original LK heuristic, the search begins from a completely random tour, based on the premise that a strong local search procedure can compensate for the lack of an initial construction heuristic. However, Helsgaun observed that better runtime performance can be achieved when the initial tour is constructed using a preference hierarchy: first, edges that are part of the minimum 1-tree; second, edges from the current best solution; and third, edges contained in the candidate set. If none of these criteria are satisfiable for a node – considering already selected edges – a random feasible edge is chosen. This initial solution is then locally optimized until no further improvement can be found, followed by multiple restarts with newly constructed initial tours.

3.2 EAX

EAX [24, 25] is a powerful genetic algorithm, which is named after its distinctive crossover operator: Edge Assembly Crossover. In this paper, however, the term EAX is exclusively used as reference to the entire heuristic, not to its crossover operator.

EAX begins with a population of tours that are already locally optimized using 2-opt. While this increases initialization time, it results in higher-quality starting solutions. Notably, even a relatively small number of 2-opt tours typically suffices to capture most of

the edges required for constructing optimal solutions [10, 35]. The main procedure of the genetic algorithm consists of two phases, which differ in how offspring solutions are constructed. In both phases, two parent tours, A and B, are first selected. A union graph G_{AB} is then formed, containing all edges from both parents. If an edge appears in both tours, it is included twice in G_{AB}. For an illustration of this and the further steps, see Figure 3. From G_{AB}, *AB-cycles* are extracted. An AB-cycle is a cycle in which edges from A and B strictly alternate. AB-cycles can be found by tracing a path of alternating edges in the union graph; whenever a node is revisited under the correct alternation, a cycle is identified. The detected AB-cycle is then removed from G_{AB}, and the traversal continues from the remaining path. In this way, all edges in G_{AB} can be assigned to AB-cycles. A collection of AB-cycles forms an E-set. Applying an E-set to parent A involves removing all A-edges contained in the E-set and adding all B-edges.

The mechanism for selecting AB-cycles for the E-set is the key distinction between the two phases of EAX. In the first, localized phase, only one AB-cycle is selected, introducing in tendency only small changes to the tour. In the second, globalized phase, multiple AB-cycles are selected via a tabu search. The goal is to construct a large E-set that induces minimal disruption, minimizing the need for costly repairs. When applying an E-set, subtours (i.e., disconnected partial tours) may arise. A repair procedure is then initiated: starting with the smallest subtour, the algorithm searches the neighborhood of its nodes to reconnect it to other subtours at the lowest possible additional cost. This process repeats until a single valid tour is restored. To estimate whether repairs will be necessary, EAX classifies nodes into three categories (see Figure 3):

- A-nodes: nodes, which are unaffected by the E-set; none of their incident edges are changed.
- B-nodes: nodes for which all four incident edges are involved in the E-set.
- C-nodes: nodes where exactly two incident edges are part of the E-set, i.e., where the new tour will transition from A to B edges and vice versa. Only the number of those transitions $\#C$ determines the number of possible subtours since a path only with edges from either A or B cannot contain a subtour.

To form a subtour two paths, one from A and one from B, have to be connected at both ends. Thus, the maximal number of subtours is $\#C/2$. Therefore, during the globalized phase, the tabu search aims to select an E-set that minimizes the number of C-nodes, reducing the likelihood and cost of repairs. For an illustration of A-, B- and C-nodes, see Figure 3. In both phases, the construction of an E-set is repeated 30 times, following the default parameters. For each resulting offspring, a fitness score is calculated based on tour cost and diversity. Diversity is measured by the change in edge entropy—i.e., the gain or loss of edge variety—relative to the current population, assuming the individual were to be accepted. The best offspring replaces parent A only if it results in a lower-cost tour.

3.3 PaCMAP

Dimensionality reduction plays a crucial role in analyzing high-dimensional data by revealing its underlying structure in a more interpretable form. A variety of techniques have been developed for this purpose [1, 21, 22], each balancing trade-offs between preserving local neighborhood information and capturing global geometric relationships. However, most methods emphasize one at the expense of the other. PaCMAP [37] is a recent approach explicitly designed to preserve both local and global structures simultaneously. It achieves this by defining three types of point pairs:

- Neighbors: the 10 nearest neighbors per point.
- Mid-near points: for each point, 5 samples of 6 randomly selected points are drawn, from which the second-closest point in each sample is retained.
- Further points: 20 randomly selected points that have not already been assigned to the point in other roles.

Using these relationships, PaCMAP defines the following loss function over a low-dimensional embedding:

$$\begin{aligned} \text{Loss} \quad = \quad & w_N \cdot \sum_{(i,j)\in\text{Neighbors}} \frac{\tilde{d}_{ij}}{10+\tilde{d}_{ij}} \\ & + w_M \cdot \sum_{(i,k)\in\text{Mid-near}} \frac{\tilde{d}_{ik}}{10\,000+\tilde{d}_{ik}} \\ & + w_F \cdot \sum_{(i,l)\in\text{Further}} \frac{1}{1+\tilde{d}_{il}} \end{aligned} \tag{1}$$

where $\tilde{d}_{ij} = \|\mathbf{y}_i - \mathbf{y}_j\|^2 + 1$ and $\mathbf{y}_i$ is the representation of point i in the lower-dimensional (here: two-dimensional) space. The weights w_N, w_M and w_F control the relative importance of the three types of relationships and are adjusted in different optimization stages. This loss is minimized via gradient descent.

Although originally developed for embedding points from high-dimensional feature spaces, PaCMAP can be adapted to work directly with pairwise distance matrices. The distances in the original space are only required to find the point pairs. This makes it well-suited for domains like combinatorial optimization, where explicit coordinate embeddings may not be available, and solution similarity must instead be inferred from structural metrics. In this work, we define the distance between two tours A and B, with edge sets E_A and E_B, respectively, as:

$$d(A,B) = |E_A| - |E_A \cap E_B| = |E_A \setminus E_B| \tag{2}$$

This distance thus corresponds to the number of edges present in tour A that are not shared with tour B. In other words, the distance is the smallest k such that a k-opt move could directly convert A into B.

4 EXPERIMENTS

To showcase our proposed visualization technique, we use instances from the benchmark set proposed by Bossek et al. [3]. These instances were evolved to be particularly difficult for EAX while being easy for LKH, and vice versa. Two mutation operator groups were employed in generating these instances: (1) simple operators, which used normal and random uniform mutations, and (2) sophisticated operators, incorporating strategies such as implosion, explosion, and grid-like structuring. This led to four types of problem instances: simple and sophisticated LKH-EAX (easy for LKH) and EAX-LKH (easy for EAX) variants. Each of the four types was generated for four instance sizes: 500, 1 000, 1 500, and 2 000 nodes.

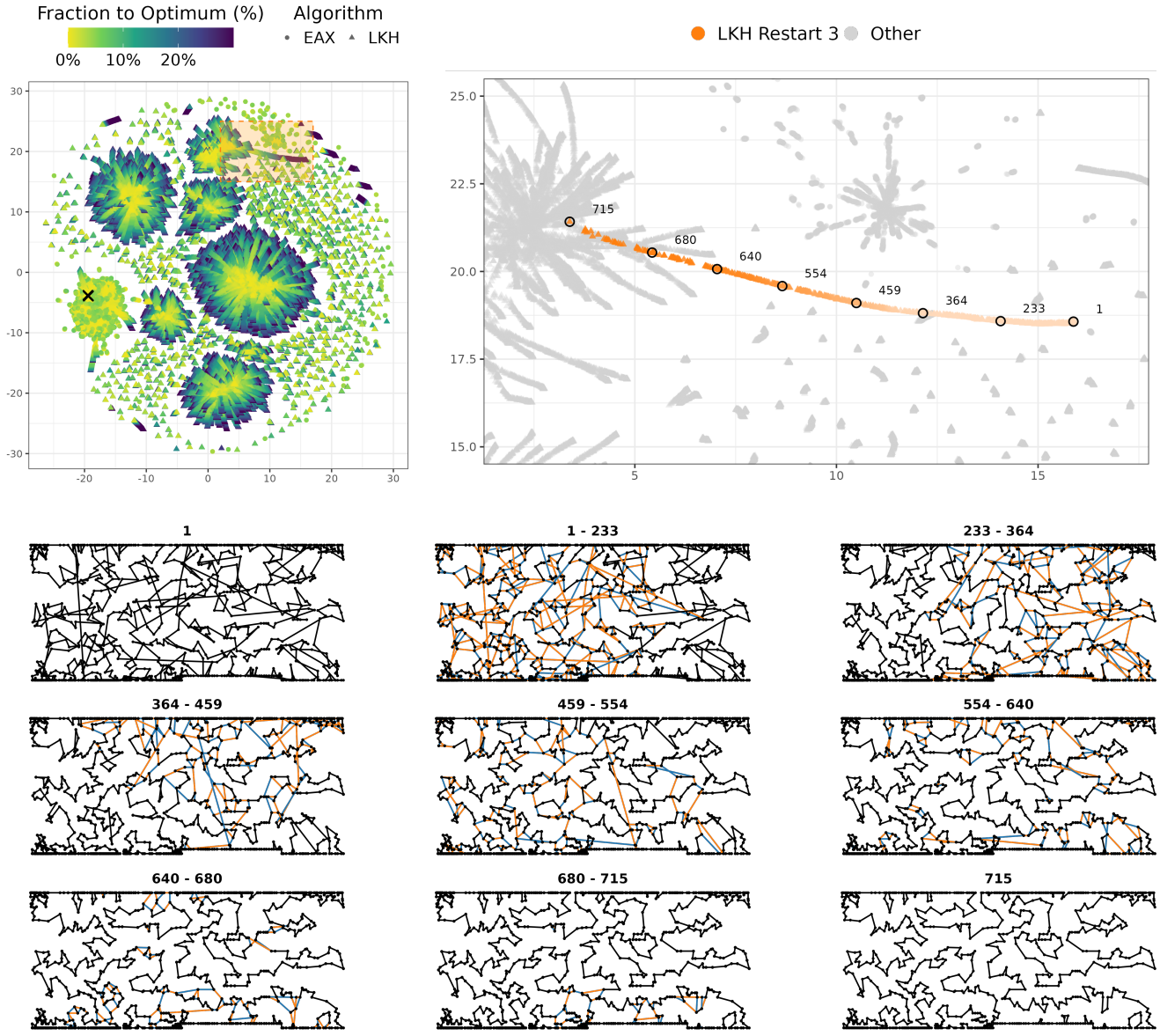

Figure 4: Top left: All recorded tours for TSPLIB instance `vm1084`, with a highlighted cutout shown at the top right featuring a search trace from LKH (in orange). The circled tours are visualized at the bottom. The first tour is shown individually, and each subsequent graph highlights the changes made to reach the next marked tour (added edges are shown in blue, removed edges in orange). Note that the initial tour in a trace often contains many crossing edges due to LKH's construction heuristics.

In our proof-of-concept experiments, designed to demonstrate the potential of our visual framework, we use the first five instances of each type with 500 nodes, along with selected instances from the TSPLIB benchmark set [33].

For each instance, both LKH and EAX are run ten times, either until the known optimum – which has been previously determined using Concorde [2] – is found or the search is terminated. For EAX, we record the entire population at each iteration. For LKH, we record the current tour after each successful k-opt move. To manage the number of total tours – and consequently the number of pairwise distance computations – we restrict EAX to a maximum population size of 100, and LKH to at most $\lceil n/3 \rceil$ restarts. Depending on the instance, this results in approximately 200 000 distinct tours. For every tour, we compute the pairwise distance to every other tour according to Equation (2). Note that regardless of the visualization technique, pairwise relationships between solutions must be computed, which inevitably leads to a quadratic increase

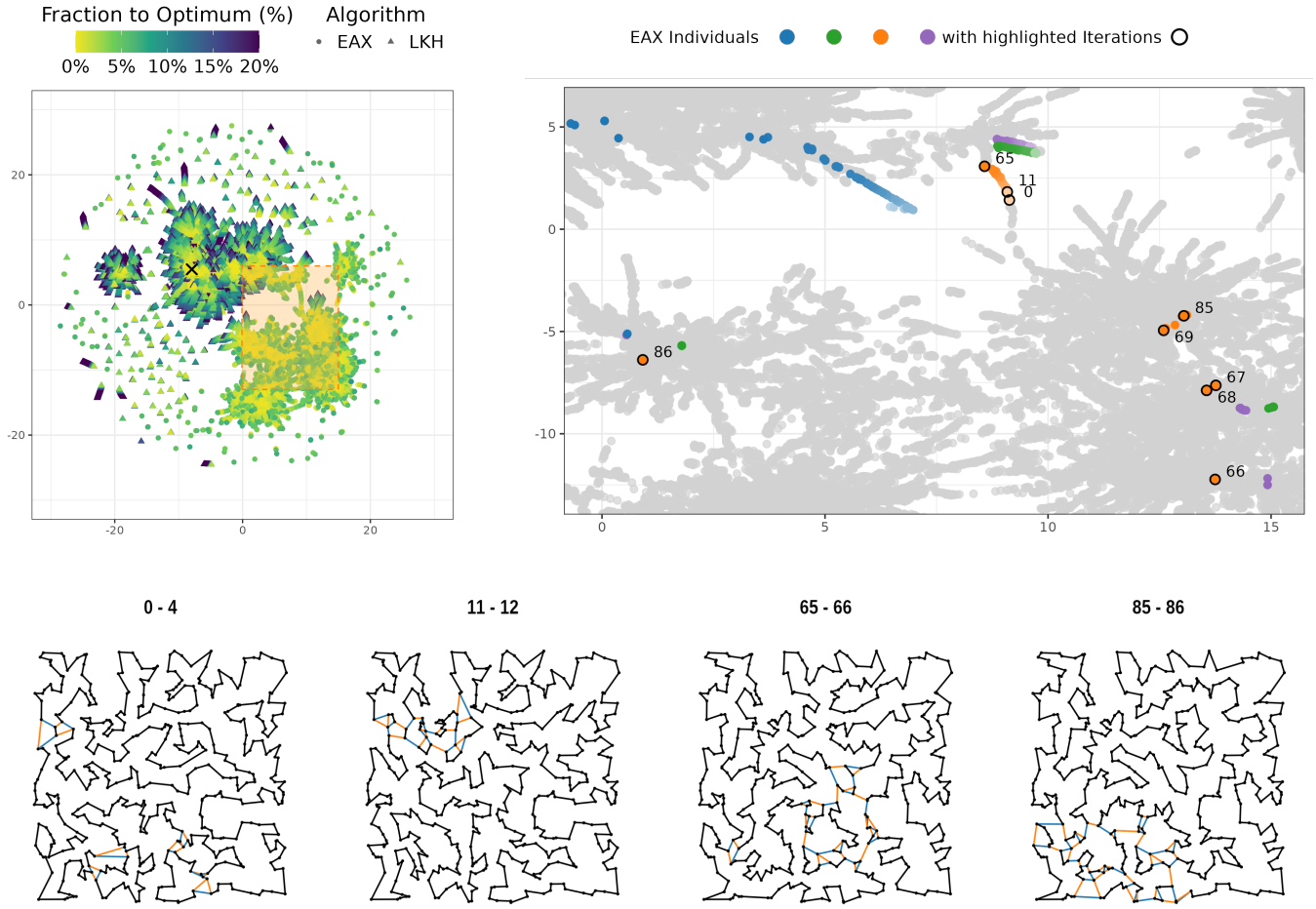

Figure 5: Top left: All recorded tours for TSP instance `500-3` from the simple LKH-EAX instance group, with a highlighted cutout shown in the top right, illustrating the evolution of several individuals over the course of the algorithm. Certain tours are circled to indicate different iterations. Bottom row: Selected tours from different phases of EAX. Red edges indicate deletions from the earlier tour; blue edges denote additions leading to the later tour specified in the plot titles. The first two plots depict steps from the local phase of EAX, which exhibits a search behavior similar to LKH. Notably, even when only a single AB-cycle is selected, the resulting changes can be substantial. The two rightmost plots show iterations from the global phase, where larger structural changes occur.

in computational effort with the number of solutions considered. These distances are then used to construct the three types of point pairs required by PaCMAP. The resulting low-dimensional embedding serves directly as the input for our visualizations. We do not explicitly connect each tour to its predecessor via edges in the derived plots, due to the large number of tours involved. Instead, relationships between tours are revealed through spatial proximity in the embedding.

5 VISUALIZATION DASHBOARD

To make our visualizations more accessible, we provide an interactive dashboard[1]. In the anonymized version, which we prepared to meet the double-blind requirements of this review process, only a single instance is displayed due to space limitations. The dashboard includes a control panel that allows users to filter which tours are displayed by selecting iteration ranges for each algorithm. Users can also opt to hide tours from either algorithm. In the embedding, EAX individuals are shown as circles, and LKH tours are shown as triangles. The points are color-coded according to the fraction of optimal tour length still missing – i.e., how much longer a given tour is compared to the minimal possible tour length. If a tour achieving this minimal length is among the displayed data, it is highlighted with a red circle. Otherwise, the optimal tour found using the Concorde exact solver [2] is marked with a red cross.

[1] https://tsp-visualizations.shinyapps.io/search-algorithm-explorer/

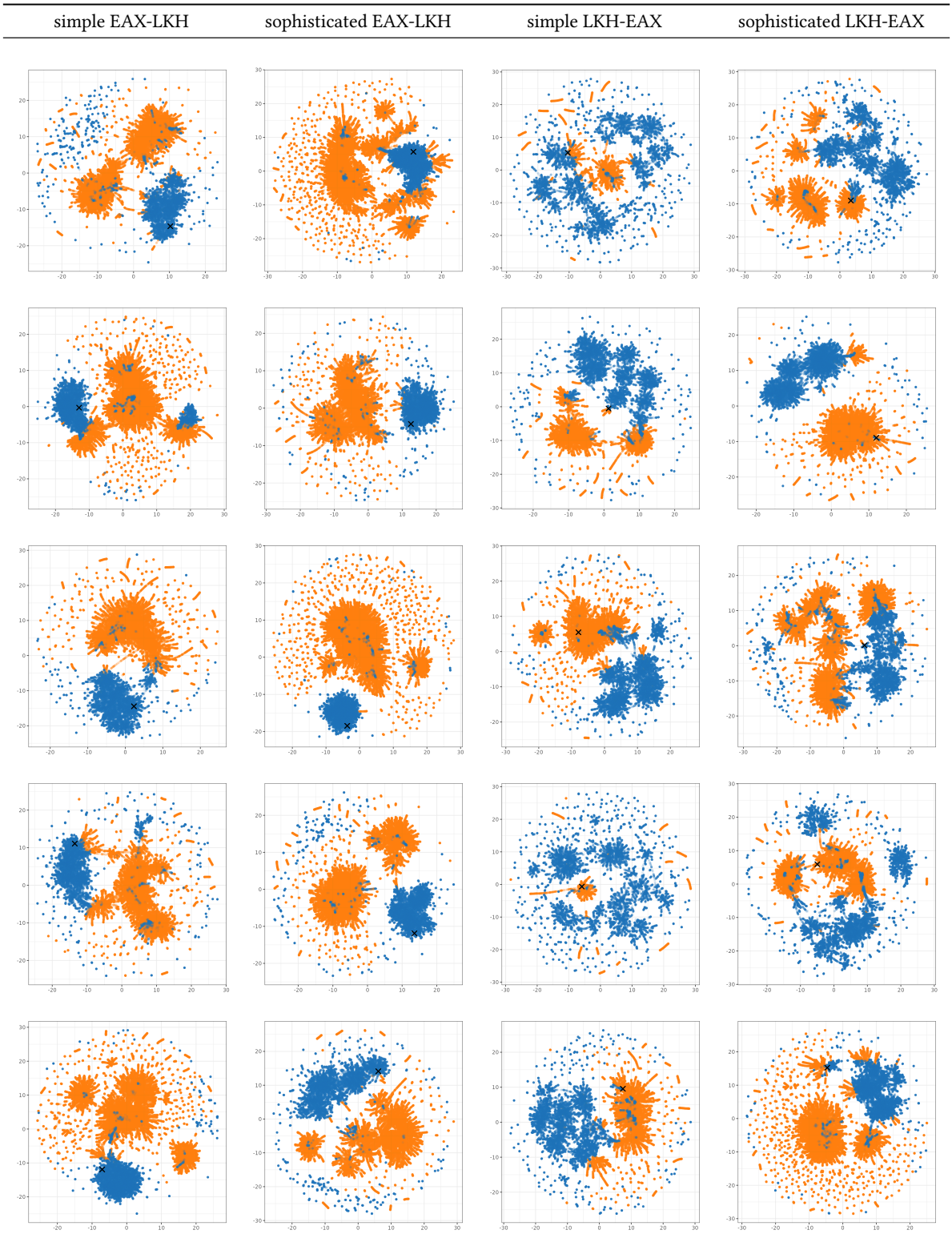

Figure 6: EAX (blue) and LKH (orange) depicted with the proposed visualization for instances simple for EAX or LKH. Both algorithms visibly have a very different search focus and depending on if the instance is difficult for LKH or EAX the optimum (black cross) is at the corresponding search focus.

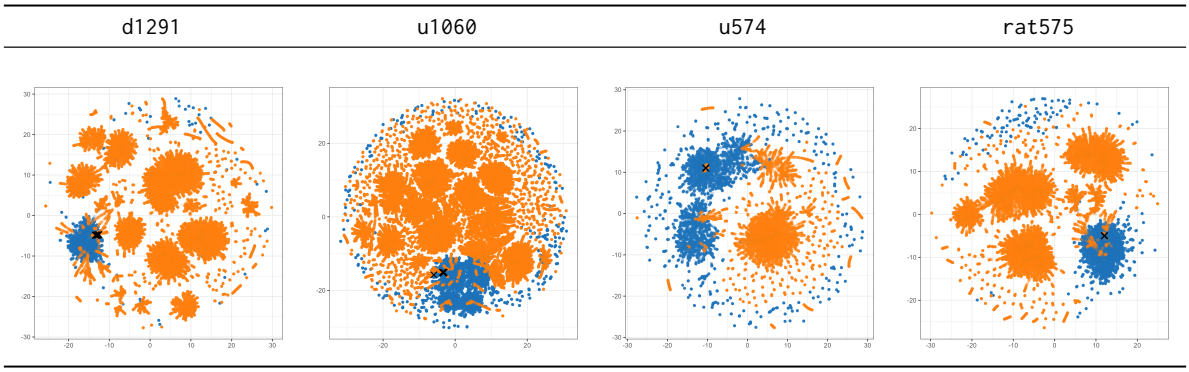

Figure 7: Visualization of EAX (blue) and LKH (orange) search behavior on four TSPLIB instances using our proposed method. Even on these standard instances, both algorithms exhibit distinctly different search focuses. Note that some instances feature multiple optima (black crosses).

To explore finer details, users can select an area within the plot to generate a focused zoomed-in view. Additionally, selecting a specific tour will display it alongside summary statistics. The selected tour is visualized with the optimal tour in the background using thicker orange edges. When a second tour is selected, edge overlap statistics are computed, and both tours are displayed together. Thereby shared edges appear in black, edges unique to the first tour in blue, and edges unique to the second tour in red. This comparative view helps highlight structural differences between tours, whether comparing two tours from the same run or across algorithms. Zoom and selections can be reset in the control panel. A checkbox enables an alternative selection mode: if activated, the area selection aggregates all selected tours into a single graph. In this view, edges that appear in all tours are shown as solid lines, while others are displayed with varying transparency according to their frequency. While this operation can be computationally expensive for large selections, it provides insight into the shared structural properties of tours within a particular region of the embedding.

6 RESULTS

In the following, we demonstrate which properties of the algorithms become visible through our visualizations, both in isolation and in direct comparison.

6.1 LKH

As shown in Figure 1 and Figure 4, LKH produces long traces in the embedding that represent the trajectory of the algorithm as it incrementally improves a tour. These traces often lead to a local optimum, which may either be isolated – unreached by other search trajectories – or situated within a larger basin of attraction that many paths converge toward. Each trace consists of a sequence of closely located tours in the embedding, reflecting the gradual nature of local search.

One particularly long trace is illustrated in detail in Figure 4. The algorithm typically starts from a poor initial tour, constructed by greedily selecting edges from the candidate set and completing it into a valid tour. Since no safeguards are in place to prevent crossing edges, many initial tours appear suboptimal. LKH then applies small local improvements in each iteration. In early iterations (see, e.g., the first three graphs in Figure 4), major improvements occur, but progress slows down as the algorithm approaches a local optimum.

Most trajectories ultimately lead to a few large basins of attraction. LKH performs well when the global optimum lies within one of these basins, but struggles on instances where it does not, as suggested in Figure 6. Additionally, many small, isolated traces can be observed, typically ending in poor local optima characterized by crossing edges that could be removed with 2-opt moves. However, LKH applies only a restricted form of 2-opt, and the necessary improving edges may not be present in the candidate set, preventing escape from these suboptimal solutions.

6.2 EAX

EAX produces search traces for individual solutions that are similar in structure to those of LKH, as illustrated in Figure 5. However, these traces typically begin from higher-quality tours – i.e., tours that are already a local 2-opt optimum – and tend to be shorter. The application of small AB-cycles to individuals is clearly visible in these sequences as illustrated in the first graph (from 0 to 4) in the bottom row of Figure 5. Still, it becomes evident that even when only a single AB-cycle is applied, the resulting change can be substantial – especially if the cycle itself is large – since no explicit mechanism limits the magnitude of the modification. During the global phase of EAX, when substantial changes are intended, individuals can make large jumps across the search space. Figure 5 shows two such jumps (from 65 to 66, and from 85 to 86), including one iteration (from 65 to 66) requiring a repair step.

Noticeably, in numerous instances, EAX identifies a single dominant basin of attraction, as shown in Figure 6. However, thanks to its global phase, EAX is also capable of jumping into basins discovered by LKH, even when there are no continuous local search paths leading to them. Interestingly, Figure 6 suggests that for instances difficult for EAX, the algorithm appears to be drawn toward a greater number of distinct basins of attraction. This observation, however, is based on a limited set of instances and warrants further investigation in future work.

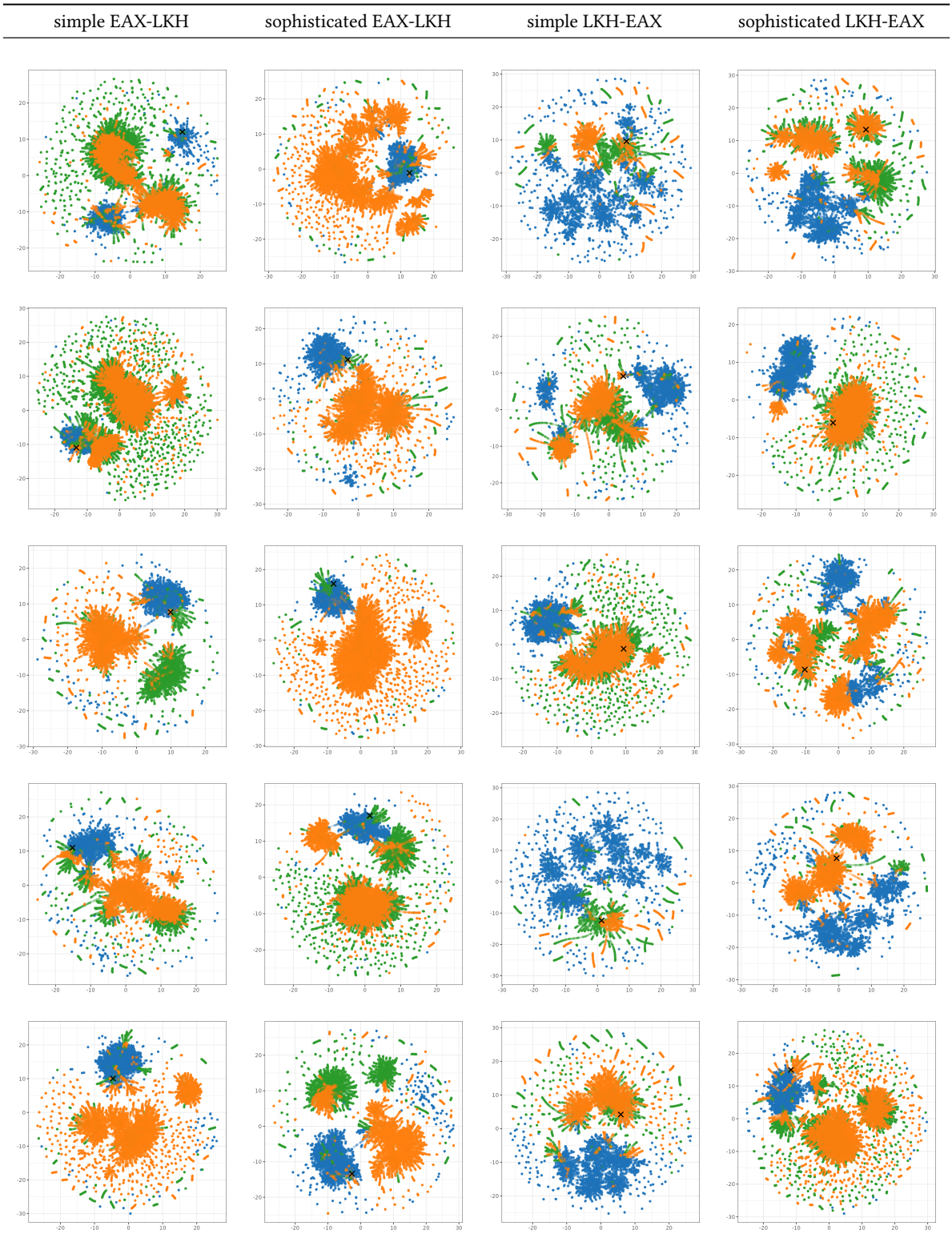

Figure 8: Visualization of EAX (blue), LKH (orange), and $LKH_{2\text{-opt}}$ (green) for instances classified as simple for either EAX or LKH. In most cases, the search behavior of $LKH_{2\text{-opt}}$ closely resembles that of standard LKH. However, for some instances shown here, $LKH_{2\text{-opt}}$ appears to explore regions within basins of attraction typically associated with EAX, which are not emphasized by standard LKH.

6.3 Comparing EAX and LKH

Across all examined instances, the comparison between EAX and LKH reveals a consistent pattern: both algorithms explore the search space with markedly different focuses and only partially overlapping regions. Figure 6 depicts their behaviors across the four types of evolved instances.

While only some of LKH's local search paths lead to basins of attraction that primarily draw EAX, EAX – if it enters these regions at all – tends to bypass these gradual paths entirely, jumping directly into the center of basins that predominantly attract LKH during the algorithm's second phase. Hence, the "difficulty" of an instance type appears to be largely determined by the basin in which the global optimum resides. Occasionally, both algorithms are drawn toward the same basin of attraction – as seen, for example, in the first instance of the simple LKH-EAX group. However, even in such cases, EAX and LKH approach the basin from different directions.

Notably, these distinctions are based purely on tour distance information; PaCMAP did not incorporate any other structural or algorithm-specific details. This pattern also holds for instances not specifically designed to challenge either algorithm, as illustrated in Figure 7 using TSPLIB instances.

6.4 Inspecting LKH with 2-opt Candidates

Inspired by recent findings showing that tour-based (rather than fixed) candidate sets can speed up LKH by an order of magnitude [10], we replicate this variant by constructing the candidate set from the 2-opt tours produced by EAX. Specifically, we collect all tours from EAX's initial population, extract the unique edges along with their frequencies, and include all unique edges as candidate edges for their incident nodes. Each edge is assigned an α-value based on the inverse of its frequency. The remainder of the LKH algorithm remains unchanged, and we refer to this variant as $\text{LKH}_{\text{2-opt}}$. Results of the comparison are shown in Figure 8.

Interestingly, there appears to be no major difference in search behavior between LKH and LKH2-opt. Both variants tend to be drawn toward the same large basins of attraction and exhibit similar patterns with respect to their initial tours. Although LKH2-opt is equipped with the same candidate edges as EAX, its search behavior remains largely complementary. However, in certain cases (e.g., instances 1–3 from the sophisticated EAX-LKH group), $\text{LKH}_{\text{2-opt}}$ finds the global optimum more quickly and with fewer restarts, and occasionally it explores basins not reached by standard LKH. These observations suggest that the reason LKH fails to locate basins relevant to EAX does not lie solely in the candidate set.

7 CONCLUSION

In this paper, we introduced a new visualization technique for analyzing the search behavior of TSP algorithms using PaCMAP. By preserving distance relationships between tours, this method enables a deeper structural understanding of both algorithmic behavior and instance characteristics. Our visual findings align with the theoretical descriptions of the algorithms, thereby validating the effectiveness of the approach.

Using this framework, we shed light on the complementary behaviors of the state-of-the-art heuristic TSP solvers EAX and LKH. Although we do not yet propose new algorithmic hybrids, our visualizations highlight structural gaps and overlaps in the explored search spaces – patterns that could inform and inspire future improvements. Our technique, along with the interactive dashboard, allows for qualitative evaluation of new algorithm variants and a more nuanced exploration of search dynamics.

We also examined a recent variant of LKH that integrates edge information from EAX tours. While this version occasionally explored regions previously dominated by EAX, its overall behavior remained similar to standard LKH, indicating untapped potential for further hybridization. Likewise, EAX still overlooks parts of the search space that LKH effectively explores, and has yet to be enhanced with LKH-inspired strategies. Hence, our visualization framework paves the way for such innovations by providing a clear lens through which to interpret and compare algorithmic behaviors.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the computing time made available to them on the high-performance computer at the NHR Center of TU Dresden. This center is jointly supported by the Federal Ministry of Education and Research and the state governments participating in the NHR (www.nhr-verein.de/unsere-partner).

This research received financial support from the German Academic Exchange Service (DAAD) under the Program for Project-Related Personal Exchange (PPP). Moreover, Darrell Whitley was supported by US National Science Foundation Grants, Award Number 1908866 and CCF Award Number (FAIN) 2426840.

This publication is based upon work from COST Action CA22137 "Randomised Optimisation Algorithms Research Network" (ROAR-NET), supported by COST (European Cooperation in Science and Technology).

REFERENCES

[1] Ehsan Amid and Manfred K. Warmuth. 2019. TriMap: Large-scale Dimensionality Reduction Using Triplets. *CoRR* abs/1910.00204 (2019).

[2] David L Applegate, Robert E Bixby, Vašek Chvátal, and William J Cook. 2011. *The Traveling Salesman Problem: A Computational Study*. Vol. 17. Princeton University Press.

[3] Jakob Bossek, Pascal Kerschke, Aneta Neumann, Markus Wagner, Frank Neumann, and Heike Trautmann. 2019. Evolving Diverse TSP Instances by Means of Novel and Creative Mutation Operators. In *Proceedings of the 15th ACM/SIGEVO Conference on Foundations of Genetic Algorithms (FOGA)*, Tobias Friedrich, Carola Doerr, and Dirk V. Arnold (Eds.). ACM, 58 – 71. https://doi.org/10.1145/3299904.3340307

[4] Georges A Croes. 1958. A Method for Solving Traveling-salesman Problems. *Operations Research* 6, 6 (1958), 791 – 812. https://www.jstor.org/stable/167074

[5] Jonathan P. K. Doye. 2002. Network Topology of a Potential Energy Landscape: A Static Scale-Free Network. *Physical Review Letters* 88 (2002), 238701. Issue 23. https://doi.org/10.1103/PhysRevLett.88.238701

[6] Jonathan P. K. Doye and Claire P. Massen. 2005. Characterizing the Network Topology of the Energy Landscapes of Atomic Clusters. *The Journal of Chemical Physics* 122, 8 (2005), 084105. https://doi.org/10.1063/1.1850468

[7] Jérémie Dubois-Lacoste, Holger H. Hoos, and Thomas Stützle. 2015. On the Empirical Scaling Behaviour of State-of-the-art Local Search Algorithms for the Euclidean TSP. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, Sara Silva and Anna Isabel Esparcia-Alcázar (Eds.). ACM, 377 – 384. https://doi.org/10.1145/2739480.2754747

[8] Jonathan Heins, Jakob Bossek, Janina Pohl, Moritz Seiler, Heike Trautmann, and Pascal Kerschke. 2021. On the Potential of Normalized TSP Features for Automated Algorithm Selection. In *Proceedings of the 16th ACM/SIGEVO Conference on Foundations of Genetic Algorithms (FOGA)*. ACM, 1 – 15. https://doi.org/10.1145/3450218.3477308

[9] Jonathan Heins, Jakob Bossek, Janina Pohl, Moritz Seiler, Heike Trautmann, and Pascal Kerschke. 2023. A Study on the Effects of Normalized TSP Features for

Automated Algorithm Selection. *Theoretical Computer Science (TCS)* 940 (2023), 123 – 145. https://doi.org/10.1016/j.tcs.2022.10.019

[10] Jonathan Heins, Lennart Schäpermeier, Pascal Kerschke, and Darrell Whitley. 2024. Dancing to the State of the Art? How Candidate Lists Influence LKH for Solving the Traveling Salesperson Problem. In *Proceedings of the International Conference on Parallel Problem Solving from Nature (PPSN)*. Springer, 100 – 115. https://doi.org/10.1007/978-3-031-70055-2_7

[11] Jonathan Heins, Darrell Whitley, and Pascal Kerschke. 2025. To Repair or Not to Repair? Investigating the Importance of AB-Cycles for the State-of-the-Art TSP Heuristic EAX. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*. ACM. https://doi.org/10.48550/arXiv.2505.00803

[12] Michael Held and Richard M. Karp. 1970. The Traveling-Salesman Problem and Minimum Spanning Trees. *Operations Research* 18, 6 (1970), 1138 – 1162. https://doi.org/10.1287/opre.18.6.1138

[13] Michael Held and Richard M. Karp. 1971. The Traveling-Salesman Problem and Minimum Spanning Trees: Part II. *Mathematical Programming* 1, 1 (1971), 6 – 25. https://doi.org/10.1007/BF01584070

[14] Keld Helsgaun. 2000. An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic. *European Journal of Operational Research (EJOR)* 126, 1 (2000), 106 – 130. https://doi.org/10.1016/S0377-2217(99)00284-2

[15] Keld Helsgaun. 2009. General k-opt Submoves for the Lin-Kernighan TSP Heuristic. *Mathematical Programming Computation* 1 (2009), 119 – 163. https://doi.org/10.1007/s12532-009-0004-6

[16] Pascal Kerschke, Lars Kotthoff, Jakob Bossek, Holger H Hoos, and Heike Trautmann. 2018. Leveraging TSP Solver Complementarity through Machine Learning. *Evolutionary Computation (ECJ)* 26, 4 (2018), 597 – 620. https://doi.org/10.1162/evco_a_00215

[17] Lars Kotthoff, Pascal Kerschke, Holger Hoos, and Heike Trautmann. 2015. Improving the State of the Art in Inexact TSP Solving using Per-instance Algorithm Selection. In *Proceedings of the 9th International Conference on Learning and Intelligent Optimization (LION)*. Springer, 202 – 217. https://doi.org/10.1007/978-3-319-19084-6_18

[18] Yuri Lavinas, Claus Aranha, and Gabriela Ochoa. 2022. Search Trajectories Networks of Multiobjective Evolutionary Algorithms. In *Applications of Evolutionary Computation (EvoApplications)*, Juan Luis Jiménez Laredo, J. Ignacio Hidalgo, and Kehinde Oluwatoyin Babaagba (Eds.). Springer, 223 – 238.

[19] Tianyu Liang, Zhize Wu, Matthias Thürer, Markus Wagner, and Thomas Weise. 2024. Generating Small Instances with Interesting Features for the Traveling Salesperson Problem. In *Proceedings of the 16th International Joint Conference on Computational Intelligence - ECTA*. INSTICC, SciTePress, 173–180. https://doi.org/10.5220/0012888800003837

[20] Shen Lin and Brian Wilson Kernighan. 1973. An Effective Heuristic Algorithm for the Traveling-Salesman Problem. *Operations Research* 21, 2 (1973), 498–516. https://doi.org/10.1287/opre.21.2.498

[21] Laurens van der Maaten and Geoffrey E. Hinton. 2008. Visualizing Data Using t-SNE. *Journal of Machine Learning Research (JMLR)* 9 (2008), 2579 – 2605.

[22] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. 2018. UMAP: Uniform Manifold Approximation and Projection. *Journal of Open Source Software (JOSS)* 3 (2018), 861.

[23] Mario A. Muñoz Acosta and Kate A. Smith-Miles. 2017. Performance Analysis of Continuous Black-box Optimization Algorithms via Footprints in Instance Space. *Evolutionary Computation (ECJ)* 25, 4 (2017), 529 – 554.

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

[25] Yuichi Nagata and Shigenobu Kobayashi. 2013. A Powerful Genetic Algorithm Using Edge Assembly Crossover for the Traveling Salesman Problem. *INFORMS Journal on Computing* 25, 2 (2013), 346 – 363. https://doi.org/10.1287/IJOC.1120.0506

[26] Gabriela Ochoa, Arnaud Liefooghe, Yuri Lavinas, and Claus Aranha. 2023. Decision/Objective Space Trajectory Networks for Multi-objective Combinatorial Optimisation (EvoCOP). In *Evolutionary Computation in Combinatorial Optimization*, Leslie Pérez Cáceres and Thomas Stützle (Eds.). Springer, 211 – 226.

[27] Gabriela Ochoa, Katherine M. Malan, and Christian Blum. 2020. Search Trajectory Networks of Population-Based Algorithms in Continuous Spaces. In *Applications of Evolutionary Computation (EvoApplications)*, Pedro A. Castillo, Juan Luis Jiménez Laredo, and Francisco Fernández de Vega (Eds.). Springer, 70 – 85.

[28] Gabriela Ochoa, Katherine M. Malan, and Christian Blum. 2021. Search Trajectory Networks: A Tool for Analysing and Visualising the Behaviour of Metaheuristics. *Applied Soft Computing (ASOC)* 109 (2021), 107492. https://doi.org/10.1016/j.asoc.2021.107492

[29] Gabriela Ochoa, Marco Tomassini, Sebástien Vérel, and Christian Darabos. 2008. A Study of NK Landscapes' Basins and Local Optima Networks. In *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation (GECCO)*. ACM, 555 – 562. https://doi.org/10.1145/1389095.1389204

[30] Gabriela Ochoa and Nadarajen Veerapen. 2018. Mapping the Global Structure of TSP Fitness Landscapes. *Journal of Heuristics* 24, 3 (2018), 265 – 294. https://doi.org/10.1007/s10732-017-9334-0

[31] Gabriela Ochoa and Nadarajen Veerapen. 2022. Neural Architecture Search: A Visual Analysis. In *17th International Conference on Parallel Problem Solving from Nature (PPSN)*, Günter Rudolph, Anna V. Kononova, Hernán Aguirre, Pascal Kerschke, Gabriela Ochoa, and Tea Tušar (Eds.). Springer, 603 – 615. https://doi.org/10.1007/978-3-031-14714-2_42

[32] Gabriela Ochoa, Nadarajen Veerapen, Fabio Daolio, and Marco Tomassini. 2017. Understanding Phase Transitions with Local Optima Networks: Number Partitioning as a Case Study. In *Evolutionary Computation in Combinatorial Optimization (EvoCOP)*, Bin Hu and Manuel López-Ibáñez (Eds.). Springer, 233 – 248.

[33] Gerhard Reinelt. 1991. TSPLIB – A Traveling Salesman Problem Library. *ORSA Journal on Computing* 3, 4 (1991), 376–384. https://doi.org/10.1287/ijoc.3.4.376

[34] Éric D Taillard and Keld Helsgaun. 2019. POPMUSIC for the Travelling Salesman Problem. *European Journal of Operational Research (EJOR)* 272, 2 (2019), 420 – 429. https://doi.org/10.1016/J.EJOR.2018.06.039

[35] Swetha Varadarajan and L. Darrell Whitley. 2019. The Massively Parallel *Mixing Genetic Algorithm* for the Traveling Salesman Problem. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, Anne Auger and Thomas Stützle (Eds.). ACM, 872 – 879. https://doi.org/10.1145/3321707.3321772

[36] Sébastien Verel, Gabriela Ochoa, and Marco Tomassini. 2011. Local Optima Networks of NK Landscapes With Neutrality. *IEEE Transactions on Evolutionary Computation (TEVC)* 15, 6 (2011), 783–797. https://doi.org/10.1109/TEVC.2010.2046175

[37] Yingfan Wang, Haiyang Huang, Cynthia Rudin, and Yaron Shaposhnik. 2021. Understanding How Dimension Reduction Tools Work: An Empirical Approach to Deciphering t-SNE, UMAP, TriMAP, and PaCMAP for Data Visualization. *Journal of Machine Learning Research (JMLR)* 22, 201 (2021), 1 – 73.