

CONTEXT-AWARE ADAPTIVE SELF-ORGANIZING MAP ALGORITHM FOR SOLVING THE TRAVELING SALESMAN PROBLEM

WANCHENG NIE¹, QIANQIAN DUAN^{1,*} AND MOHAMMAD KARIMI²

¹School of Electrical and Electronic Engineering
Shanghai University of Engineering Science
No. 333, Longteng Road, Shanghai 201620, P. R. China
m325123227@sues.edu.cn; *Corresponding author: 02160010@sues.edu.cn

²Department of Electrical Engineering
Tafresh University
Tafresh 39518-79611, Iran
pwr99.karimi@tafreshu.ac.ir

Received August 2025; revised December 2025

ABSTRACT. *The traditional Self-Organizing Map (SOM) algorithm encounters difficulties in solving large-scale Traveling Salesman Problem (TSP) due to inefficient convergence, susceptibility to initialization, and suboptimal termination criteria. To overcome these limitations, this paper introduces a Context-Aware Adaptive Self-Organizing Map (CASOM). CASOM leverages K-means clustering with Positional Encoding (PE) to distribute neurons based on density, ensuring structured alignment with the input space. An adaptive distance weighting strategy derives distance weights from Multi-Head Attention (MHA) between neurons and input data, dynamically modulating these weights across different training stages to balance global exploration and local refinement. During the competition phase, the Best Matching Unit (BMU) is selected by integrating distance-based and attention-based relevance. In the cooperation phase, MHA refines neuron positioning while Graph Neural Network (GNN) aggregation maintains both global topological structures and local relationships. A masking mechanism excludes converged neurons, reducing redundant updates and regulating convergence through stratified sampling and dynamic threshold checks. Experimental results demonstrate that CASOM surpasses conventional SOM and heuristic methods on several relatively large-scale TSP instances, improving optimization gaps by 2.56% to 5.00% and reducing runtime by 56.63% to 88.61%. This framework not only enhances accuracy and convergence speed but also significantly improves stability.*

Keywords: Self-organizing map, Traveling salesman problem, Positional encoding, Adaptive distance weighting, Multi-head attention, Graph neural network, Masking mechanism

1. Introduction. The Traveling Salesman Problem (TSP) is a fundamental combinatorial optimization problem that seeks the shortest route visiting each city exactly once before returning to the starting point [1]. Due to its NP-hard nature, exact algorithms become computationally infeasible for large instances, necessitating heuristic and approximation methods such as dynamic programming, branch-and-bound, genetic algorithms, simulated annealing, and k-opt heuristics [2]. Advanced metaheuristics, including tabu search, ant colony optimization, Lin-Kernighan heuristics, artificial immune systems, and deep reinforcement learning, have further improved solution efficiency [3-5].

Among these methods, the Self-Organizing Map (SOM), an unsupervised neural network introduced by Kohonen in 1982, offers a unique bio-inspired paradigm [6]. Its ability to project high-dimensional data onto a lower-dimensional space (typically a one-dimensional ring for TSP) while preserving topological relationships makes it naturally suited for the problem [7,8]. The SOM-based approach iteratively adjusts a ring of neurons through competitive learning and cooperative updates until it converges to a feasible tour. Since its adaptation for TSP in the late 1980s, it has been recognized for providing competitive solutions with relatively low computational complexity compared to many exact methods [9].

Despite its conceptual elegance, the traditional Self-Organizing Map (SOM) confronts several intertwined challenges that critically constrain its performance and practicality, especially when scaling to large or complex TSP instances. These limitations can be categorized into four primary aspects. First, its pronounced sensitivity to random initialization places the algorithm on a precarious path from the outset. Without leveraging the underlying city distribution, the initial neuron topology often deviates significantly from an optimal configuration, predisposing the model to entrapment in local optima and yielding suboptimal tour quality [10]. Second, the inherent rigidity in its update mechanism, governed by fixed neighborhood functions and learning rates, fails to accommodate the diverse spatial densities and geometric structures encountered across different TSP instances [11]. This one-size-fits-all approach lacks the contextual adaptability necessary for robust performance. Third, the use of a static network structure, where the neuron count is typically a fixed multiple of the number of cities, frequently leads to inefficient computational resource allocation. This results in an over-representation of neurons in sparse regions and an under-representation in dense clusters, ultimately impairing the solution's resolution and accuracy [12]. Finally, the dependence on predetermined termination criteria, such as a maximum iteration count or a minimum learning rate threshold, disregards the actual convergence state of the mapping process. This can cause either premature cessation before a stable solution is found or, conversely, substantial computational waste on redundant iterations that offer no meaningful improvement [13].

In response to these challenges, a substantial and diverse body of research has emerged, aiming to fortify the SOM framework through various innovative strategies. These improvements can be systematically classified into several complementary strands. Architectural modifications seek to redesign the SOM's output structure itself. For instance, Dantas introduced the Expanding SOM (ESOM) by drawing inspiration from self-organized criticality and slow expansion observed in physical models of the universe. Their model propels neurons beyond the convex hull of cities, leveraging a dynamic where microscopic interactions lead to macroscopic structural expansion, thereby better encapsulating the global layout for comprehensive path planning [14]. In parallel, to tackle complex variants such as the Single Depot-Multiple Traveling Salesman Problem (SD-MTSP) with a min-max objective, Guan et al. designed a hierarchical approach named HiSOM based on a divide-and-conquer framework. This method effectively decomposes the SD-MTSP into two subproblems: task allocation, where a novel soft label concept and an SGD-SOM framework are introduced to optimize city-salesman associations, and route planning, where a TOSOM framework generates topologically-ordered tours that adhere to the convex hull property. This hierarchical design not only manages problem complexity through layered optimization but also effectively mitigates path intersection, enhancing both solution quality and computational efficiency in multi-objective routing scenarios [15].

Algorithmic hybridizations synergize SOM with other powerful optimization paradigms to enhance solution quality and address specific limitations. Tan et al., for instance, developed a hybrid framework to overcome the slow convergence and local optima susceptibility of Ant Colony Optimization (ACO) as well as the limited local search capability of Simulated Annealing (SA). Their approach integrates an enhanced ACO with a double-improved SA, utilizing a clustering method to structure the search process, which significantly refines solution quality and accelerates convergence on various TSPLIB datasets [16]. This work exemplifies how the broad global exploration typical of SOM could be effectively augmented by embedding similarly intensive local refinement techniques. Furthermore, the integration of graph-based neural architectures extends this hybridization paradigm into the realm of deep learning. Liang et al. proposed a Bipartite Graph Neural Network (BiGNN) incorporating an attention mechanism to solve complex Multiple Traveling Salesman Problems (MTSP) in urban logistics. Their model formulates MTSP as a mixed integer linear programming problem and employs a bipartite graph structure to represent constraints and variables, using imitation learning to guide a branch-and-bound solver. This enables the model to learn a strong branching strategy, replacing complex calculations with fast approximations and effectively reducing solution time while maintaining accuracy [17]. Collectively, these hybrid strategies demonstrate a trend of replacing traditional components, such as the standard Gaussian neighborhood function in SOM, with more adaptive and learnable mechanisms to better navigate complex solution spaces [16,17].

Internally adaptive mechanisms represent another significant direction in enhancing SOM's autonomy and precision for TSP solving. Hameed et al. developed an Adaptive SOM (ASOM) model that incorporates a reweighted zero-attracting method to improve network sparsity and convergence, enabling more effective initialization based on geometric data clusters and dynamic modulation of learning parameters in response to local density variations [18]. Complementing parameter adaptation, Ding et al. introduced a fractional-order memristive Hopfield neural network framework, which employs fractional neuron labeling strategies to resolve ambiguities in city sequencing and enhance solution reversibility in complex optimization scenarios [19]. Further refining the learning dynamics, Li et al. proposed a novel fractional-order memristive Hopfield neural network with generalized competition mechanisms, establishing more equitable and precise neuron update processes within defined infiltration radii to prevent distant neurons from disrupting emergent path formation [20]. These studies collectively demonstrate how internal adaptive mechanisms – spanning parameter adjustment, labeling strategies, and competition rules – can significantly augment SOM's self-organizing capabilities in complex routing tasks.

Recent advancements in related computational intelligence fields provide valuable perspectives that can further inform SOM development. Notably, the work by Li on dynamic network representation learning, which integrates Graph Neural Networks (GNNs) with temporal information [21], underscores the profound impact of capturing dynamic, relational contexts within a data structure. This principle resonates deeply with the challenge of preserving topological relationships in TSP paths and suggests that incorporating analogous context-aware, GNN-driven aggregation mechanisms could significantly enhance the SOM's cooperative phase, ensuring both local and global topological consistency.

Simultaneously, research in audio signal processing by Ranny et al. demonstrated the efficacy of a specialized masking technique within their proposed “Separation SELDnet” framework [22]. Their work addresses the challenge of accurately localizing and detecting overlapping sound events, a scenario analogous to complex, real-life auditory environments. The core of their approach involves generating and applying a time-frequency

mask to separate mixed audio signals, thereby isolating individual sound sources for more precise identification. This methodology of using an adaptive filtering mechanism to resolve overlapping and interfering patterns offers a compelling conceptual paradigm. It inspires the consideration that an analogous logical mechanism – though distinct in technical implementation – could be designed to dynamically “separate” or identify converged neurons from those still actively adapting during SOM training. Such a mechanism would sharpen the algorithm’s focus on regions requiring further optimization, thereby improving computational efficiency.

Notwithstanding these progressive advancements, a critical observation is that many existing solutions tend to address the inherent limitations of traditional SOM in a somewhat isolated manner. It is also evident that the potential of several advanced techniques from adjacent fields – particularly the context-aware representation learning exemplified by GNNs [21] and the pattern-separation concepts inspired by advanced masking techniques [22] – has not yet been fully explored within the specific context of SOM for TSP. The pivotal inspiration drawn from these studies provides a decisive rationale for integrating analogous mechanisms to address the aforementioned limitations. Therefore, while individual improvements in architecture, hybridization, or parameter adaptation yield benefits, a cohesive and integrated framework that systematically combines structured initialization, dynamic and data-aware weight adaptation, robust topological modeling grounded in relational context, and efficient resource allocation guided by intelligent filtering remains an open challenge. The absence of such a holistic integration explains why many contemporary methods still struggle with robustness and consistently deliver high-quality solutions across the vast and diverse landscape of large-scale TSP instances. This gap in the literature precisely defines the contribution and positioning of the present study [13-22].

This study builds upon the foundational principles of adaptive SOM and introduces a novel, integrated framework termed the Context-Aware Adaptive Self-Organizing Map (CASOM) to holistically resolve these limitations. The main contributions of this work are threefold.

- 1) A structured initialization strategy is proposed, which integrates Positional Encoding (PE) via sinusoidal functions to embed rich geometric and spatial information into a high-dimensional feature space. This PE-guided representation is then utilized in a K-means clustering strategy to initialize neurons, ensuring their initial distribution is aligned with the city geometry and density, thereby mitigating the pitfalls of random initialization.

- 2) The core SOM update process is revolutionized through a Multi-Head Attention (MHA) mechanism. During the competition phase, MHA dynamically computes attention weights, creating a learnable, context-aware metric that refines Best Matching Unit (BMU) selection and update magnitudes. In the cooperation phase, these attention weights serve as priors for a Graph Neural Network (GNN) aggregation module, which propagates information between neighboring neurons. This ensures the preservation of both local and global topological relationships during updates, significantly enhancing the model’s adaptability.

- 3) Optimized convergence is achieved through a masking mechanism that dynamically identifies and excludes converged neurons based on stable MHA weights and minimal PE distance error, thereby eliminating redundant computations. This is coupled with phase-based parameter adjustments for distance weights and a cluster-sampling inspection method for termination, which together ensure efficient resource allocation and accurate convergence detection throughout the training process.

The remainder of this paper is organized as follows. Section 2 reviews the foundational concepts of SOM and related work. Section 3 elaborates on the proposed CASOM

methodology. Section 4 presents the experimental setup and discusses the results. Finally, Section 5 concludes the paper and suggests future research directions.

2. Related Work.

2.1. Self-Organizing Map (SOM). The Self-Organizing Map (SOM), proposed by Kohonen in the 1980s, is an unsupervised neural network algorithm also known as the Kohonen Map [7]. Each neuron contains a weight vector iteratively adjusted to approximate input patterns. During training, the Best Matching Unit (BMU) and its neighbors update weights proportionally to input proximity, with influence decreasing over iterations [7].

Traditional SOM initializes neuron weights via small random values or Principal Component Analysis (PCA) eigenvectors [8]. Figure 1 illustrates the basic structure, where input data are mapped onto output neurons via adjustable weight vectors, eventually forming an ordered representation of input data topology.

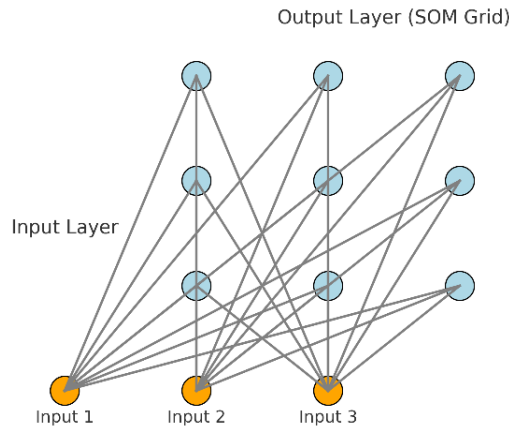


FIGURE 1. Schematic diagram of the SOM network

2.2. Application of standard SOM to the TSP problem. The Traveling Salesman Problem (TSP), an NP-hard combinatorial optimization challenge, requires identifying the minimal-weight Hamiltonian circuit traversing all cities exactly once [1]. The Self-Organizing Map (SOM) algorithm approximates solutions by iteratively optimizing neuron weights to capture city topology [14]. The fundamental process of the standard SOM for TSP is conceptually illustrated in Figure 2, which depicts the key stages of neuron evolution from initial random distribution to a final converged state representing the solution path.

For two-dimensional coordinates, standard SOM execution involves three phases:

1) Competition: For each input city X (represented as $\bullet$ in Figure 2) presented to the network during training, the algorithm searches for the most responsive neuron, known as the Best Matching Unit (BMU). This is achieved by comparing the city's coordinate vector X to the weight vector W_j of every neuron j in the network. The neuron whose weight vector most closely matches the input city, as measured by the minimum Euclidean distance, is declared the winner:

$$i(X) = \arg \min_j |X - W_j| \quad (1)$$

where $i(X)$ denotes the index of the BMU for city X [8,15]. In the context of Figure 2, this phase corresponds to associating each solid city point ($\bullet$) with its closest neuron from the ring (initially $\bullet$, later $\circ$).

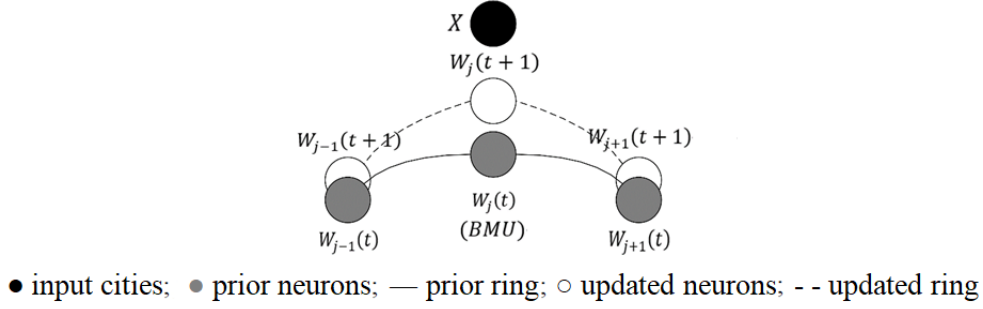


FIGURE 2. Standard SOM mechanics for TSP

2) Cooperation: Once the BMU is identified, it acts as the center of a topological neighborhood defined by a radius σ . The core idea is that not only the BMU itself (neuron j in Figure 2) but also its topologically adjacent neighbors within this radius on the ring (which, for instance, may include neurons at positions $j-1$ and $j+1$, as determined by the Gaussian kernel calculation) should be updated. This cooperative mechanism ensures the local continuity of the evolving path by synchronously adjusting a group of neurons around the winner. The strength of this cooperative influence is quantified by a neighborhood function h_{ij} , typically a Gaussian kernel:

$$h_{ij} = \exp\left(-\frac{|r_i - r_j|^2}{2\sigma^2}\right) \quad (2)$$

Here, r_i and r_j are the positions of neurons i and j on the one-dimensional ring lattice (not in the city coordinate space), and σ is the effective width of the neighborhood, which decreases monotonically over time [15]. This function ensures that neurons that are close to the BMU on the ring (i.e., have small $|r_i - r_j|$) receive a strong update signal, while those farther away are affected less, as visualized by the local adjustment of the neuron ring in Figure 2.

3) Adaptation: In this phase, the weight vectors of the BMU and its neighboring neurons are synchronously adjusted to better match the input city X . This learning rule pulls these neurons closer to the city's coordinates in the input space. The update formula is given by

$$W_j(t+1) = W_j(t) + \alpha(t) \cdot h_{ij} \cdot (X - W_j(t)) \quad (3)$$

In this equation, $W_j(t)$ represents the current weight vector (spatial coordinates) of neuron j at discrete time step (iteration) t . The term $\alpha(t)$ is the learning rate, a factor between 0 and 1 that also decreases over time, controlling the overall magnitude of the update. The difference vector $(X - W_j(t))$ points from the current neuron position towards the target city. The entire product is scaled by the neighborhood function h_{ij} from Equation (2), making the update localized both in the data space and in the topological space of the ring. Consequently, $W_j(t+1)$ denotes the new, updated position of neuron j after this single adaptive step at iteration t [8]. This process is visually depicted in Figure 2 by the movement of neurons from their old positions (●) to their new positions (○), and the corresponding deformation of the ring from solid (—) to dashed (---) lines.

As conceptually summarized in Figure 2, this cycle of Competition, Cooperation, and Adaptation is repeated for numerous iterations and across all input cities. The learning rate $\alpha(t)$ and neighborhood radius $\sigma(t)$ are gradually reduced according to a predefined schedule. This causes the initially disordered neuron ring to progressively align itself with the geometry of the city set. The sequence of BMU selections defines the visiting order,

and the final, stable configuration of the neuron ring, connecting the cities in the sequence of the final BMUs, yields an approximate solution to the TSP [8,14,15].

2.3. Analysis of existing adaptive SOM algorithms. Building upon the systematic classification of research strands outlined in the Introduction, this section provides a critical analysis of specific adaptive SOM algorithms. While these methods have incrementally addressed the inherent limitations of the traditional SOM, a thorough examination reveals persistent challenges and underscores the necessity for a more cohesive solution, thus motivating the integrated design of our CASOM framework.

Architectural modifications, such as approaches inspired by self-organizing and expanding models to improve global structure capture [14], effectively aid in capturing the global layout but often remain constrained by their reliance on the original coordinate space. Similarly, hierarchical approaches like HiSOM decouple task assignment from path planning to tackle complex variants such as multiple TSPs [15], yet they introduce increased structural complexity and may suffer from insufficient information flow between hierarchical levels. This reliance on simplistic spatial representations limits their capacity for nuanced, density-aware initialization. Adaptive SOM (ASOM) models that initialize neurons based on geometric data clusters and adjust parameters according to local density [18] represent a step forward, but their strategies for distributing neurons often lack the robustness required for highly irregular data distributions. These identified shortcomings collectively highlight the critical need for a structured, data-aware initialization process. Inspired by recent advances in representation learning, our integration of Positional Encoding (PE) with K-means clustering is motivated by the need to project cities into a high-dimensional feature space [21], while the underlying principle of component separation and focusing computational resources, as seen in advanced filtering techniques from other domains [22], informs our approach to handling complex spatial distributions. This allows us to precisely align neuron distribution with city geometry and density.

To enhance the adaptability of the neuron update process, several mechanisms have been proposed. Internally adaptive mechanisms, such as strategies that employ concepts akin to fractional labeling to refine influence distribution [19], aim to increase the influence of neurons proximal to city locations to resolve sequencing ambiguities. Furthermore, refined competition and cooperation rules, such as generalized competition within defined infiltration radii, strive to establish a more equitable and precise neuron update process, preventing distant neurons from disrupting the emerging path formation [20]. Despite their merits, these approaches are fundamentally heuristic and non-learnable. They cannot dynamically adjust the “attention” or “influence” of neuron updates based on the complex, contextual relationships within the input data. This inherent rigidity, stemming from fixed rules or radii, necessitates a paradigm shift towards a fully data-driven, context-aware weighting strategy. Consequently, we introduce a Multi-Head Attention (MHA) mechanism to revolutionize the competition phase. By employing learnable query-key-value interactions, our MHA module creates a dynamic and context-sensitive metric for Best Matching Unit (BMU) selection and update magnitudes, moving beyond fixed-distance heuristics. The integration of attention mechanisms and graph-based learning, as explored in other complex optimization contexts [23,24], supports the feasibility and value of this direction.

Algorithmic hybridizations represent another significant avenue for improvement, where the fusion of SOM with powerful optimizers like Simulated Annealing (SA) leverages SOM for broad global search and SA for intensive local refinement [16,17]. Some implementations further enhance this by refining the neighborhood mechanism. For instance, the principle of integrating complementary optimization strategies within a hybrid framework

[16] exemplifies the enhancement of traditional SOM, and the introduction of adaptive neighborhood activation helps prevent overfitting [25]. Although these hybrids boost performance, they often result in loosely coupled, multi-stage pipelines lacking a unified, end-to-end optimization perspective. The integration is typically procedural rather than architectural, which can limit the synergy between the components. More critically, the core cooperative phase in SOM, even in these hybrids, is typically still governed by a fixed or heuristically adjusted neighborhood function, which remains inherently limited in its ability to capture and preserve complex, non-local topological relationships. This gap calls for an internally integrated component capable of explicit topological modeling. In response, we incorporate a Graph Neural Network (GNN) aggregation module in the cooperation phase. The conceptual advancement of using neural networks to model relational structures, as seen in dynamic network representation learning [21], provides a foundational motivation. Utilizing the attention weights generated by MHA as prior information for edge weighting, the GNN explicitly maintains and propagates both local and global topological structure within the neuron ring, achieving a level of relational reasoning that previous methods lack.

Finally, issues of convergence and computational efficiency persist across these research strands [5]. Most existing methods, including those mentioned above, continue to depend on predetermined termination criteria, such as a maximum iteration count or a minimum learning rate [27]. This fails to address the core problem of redundant calculations performed on neurons that have already converged, leading to substantial computational waste in the later stages of training. The static resource allocation inherent in these methods means that computational effort is not dynamically redirected to the problem areas that need it most. Inspired by the efficacy of selective focusing and resource allocation mechanisms in other computational fields, such as the use of masking in signal processing [22], we designed a masking mechanism to dynamically identify and exclude converged neurons based on stable MHA weights and minimal PE distance error. This ensures efficient resource allocation by sharpening the algorithm’s focus on regions still requiring refinement, directly addressing an inefficiency that has been largely overlooked in prior SOM research.

In summary, while the body of research on adaptive SOM has produced valuable incremental advances, our analysis confirms that these approaches often tackle the fundamental limitations of SOM – initialization, update rigidity, topological preservation, and resource efficiency – in isolation. The absence of a cohesive framework that synergistically combines data-aware initialization, learnable and context-sensitive updates, explicit topological modeling, and dynamic resource allocation constitutes a significant gap in the literature. It is this very gap that justifies and motivates the integrated design of our proposed CASOM model, which systematically incorporates PE, MHA, GNN, and Masking to overcome these challenges holistically.

3. Methods.

3.1. K-means clustering with positional encoding. SOM algorithms leverage topological ordering properties to map input city data through processes of competition, cooperation, and adaptation, resulting in approximate global solutions for the TSP. However, the random distribution of initial neurons in terms of quantity and position presents the possibility of overlooking the geometric and quantitative characteristics of the input data. This randomness may overrepresent sparse regions and underrepresent dense ones. Moreover, randomly selecting starting nodes increases the possibility of the algorithm

getting trapped in local optima, restricting its scalability and adaptability to variations in problem size and distribution.

3.1.1. Geometric distribution information. This paper initializes neuron positions via K-means clustering with Positional Encoding (PE), implementing a divide-and-conquer strategy [28]. PE first embeds spatial/geometric information into features, followed by K-means clustering. While both K-means and SOM perform clustering analysis, K-means uses a different approach. While both K-means and SOM perform clustering, K-means minimizes intra-cluster squared distances using distance-based similarity metrics [9,10].

PE transforms raw coordinates into high-dimensional embeddings using periodic functions. For position t , the encoding is

$$PE_{t,i} = \begin{cases} \sin\left(\frac{t}{10000^{2i/d}}\right), & \text{if } i \text{ is odd} \\ \cos\left(\frac{t}{10000^{2i/d}}\right), & \text{if } i \text{ is even} \end{cases} \quad (4)$$

where t is the position index, i is the feature dimension index, and d is the total embedding dimension. Positional Encoding, denoted as $PE_{t,i}$, is designed to embed the position of each data point within a continuous space. These continuous embeddings preserve spatial relationships, enabling K-means to better detect patterns in irregular distributions. PE mitigates density imbalance issues in traditional SOM initialization by distinguishing geometrically proximate points [14].

The optimal number of clusters, k , is determined using the Elbow Method by evaluating the total Sum of Squared Errors (SSE) for different cluster counts [29]. SSE quantifies cumulative point-centroid distances within clusters.

Sum of Squared Errors (SSE) quantifies clustering quality by calculating the sum of squared distances between each positional encoding vector $PE_{t,i}$ and its respective cluster centroid. A smaller SSE indicates higher clustering quality, implying that data points embedded by positional encodings are grouped tightly within clusters, thereby clearly distinguishing clusters from each other. For a given number of clusters k , the SSE is computed as follows:

$$SSE = \sum_{i=1}^k \sum_{x \in C_i} |PE_{t,i} - \mu_i|^2 \quad (5)$$

Here, C_i represents the i -th cluster, $PE_{t,i}$ denotes the positional encoding for the data point x within C_i , and μ_i is the centroid of C_i . The symbol $|\cdot|$ denotes Euclidean distance. SSE is computed for each k , ranging from 1 to the maximum number of clusters. By plotting SSE against the number of clusters, an Elbow plot is generated. In high-dimensional data analysis, particularly when dealing with spatial or sequential data, the use of Positional Encoding (PE) allows the model to incorporate spatial or geometric information about each data point relative to other points in the feature space. This is particularly important when traditional data points x (e.g., raw feature vectors) do not fully capture the underlying structure, such as in the case of irregularly distributed data or when there is a need to preserve relative spatial relationships.

The main advantage of using $PE_{t,i}$ instead of x is that it enhances the ability of the model to maintain relative spatial relationships across data points by creating smooth, continuous embeddings. This preserves both local and global geometric relationships. As shown in Figure 3, the elbow point – where the rate of SSE reduction slows – indicates

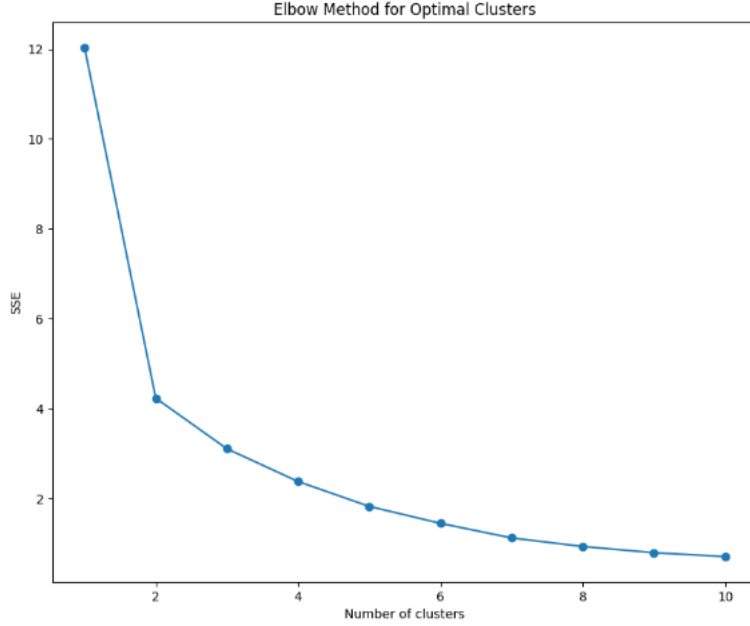


FIGURE 3. Elbow Method for determining optimal number of clusters

the optimal number of clusters:

$$optimal_clusters = \arg \max \left(\frac{d^2}{dk^2} SSE(k) \right) + 2 \quad (6)$$

In this formula, “*optimal_clusters*” refers to the ideal number of clusters, $SSE(k)$ is the sum of squared errors for different cluster numbers k , $\frac{d^2}{dk^2}$ represents the second derivative of SSE with respect to the number of clusters, and $\arg \max$ identifies the cluster number where the maximum value occurs. The second derivative reveals the rate of change in SSE, with the elbow point marking the optimal cluster number. This method reduces the need for manual parameter selection, increases adaptability, and ensures consistent performance across various datasets.

After identifying the optimal number of clusters, the neuron distribution becomes more structured. Based on the Expanding Self-Organizing Map (ESOM) approach [16], an inward expansion from the convex hull is applied. This method leverages convex hull data from each cluster to maintain consistent topological relationships among neurons from the start. It eliminates the need for a transition from disorder to order, allowing SOM to adjust neuron positions and connections more efficiently. This reduces the number of iterations and improves search speed.

3.1.2. Quantity information. Self-Organizing Maps (SOMs) perform well in sparse regions but struggle in dense areas due to uniform neuron allocation. Increasing neurons improves data capture but increases time complexity and computational cost [12]. Uniform allocation causes overrepresentation in sparse regions and underrepresentation in dense areas.

A density-based dynamic neuron allocation method addresses this by integrating Positional Encoding (PE). PE first embeds spatial/geometric information into input data, enabling accurate density estimation for irregular distributions. Following PE-enhanced K-means clustering, neurons are allocated proportionally:

$$num_neurons = \left\lfloor \frac{total_neurons}{optimal_clusters} \times (1 + \alpha \cdot PE_{t,i}) \right\rfloor \quad (7)$$

where *optimal_clusters* is the optimal number of clusters derived from Equation (6), *total_neurons* represents the total number of neurons in the SOM network, and $PE_{t,i}$ is the Positional Encoding (PE) of each neuron, capturing the spatial relationships between neurons. PE embeds spatial relationships between neurons, with higher PE values in dense regions reflecting central positions. This allocates more neurons to high-density clusters for finer representation, while sparse regions receive fewer neurons. PE captures subtle density variations within regions through embedded geometric features.

Neuron counts adjust dynamically to cluster density. High-density regions receive more neurons, low-density regions fewer, aligning allocation with data distribution. This optimizes computational resources by reducing neurons in sparse areas while improving dense region resolution. The approach balances training, prevents over/undertraining, and maintains computational efficiency across problem scales.

3.2. Multi-head attention-enhanced adaptive distance weights. Traditional SOM algorithms adjust neuron positions using fixed neighborhood functions and learning rates (Equations (2) and (3)). These methods consider only 1D inter-neuron distances, ignoring 2D neuron-city distances. Early-stage rapid adjustments risk network destabilization when distant neurons move abruptly, causing premature convergence and reduced refinement capability. Conversely, insufficient late-stage adjustments near input points limit precision [30].

This study introduces adaptive distance weights with Multi-Head Attention (MHA) to address these issues. Adaptive weights dynamically regulate updates using neuron-city distances, while MHA captures global/local neuron-data relationships across feature dimensions, ensuring accurate spatial and high-dimensional modeling.

In the adaptive strategy, distance weights are modulated by nonlinear parameters across training stages. For iterations $< \textit{check_start}$, global alignment dominates with rapid adjustments. Beyond *check_start*, convergence-phase refinements prioritize precision. This staging mitigates early disruptions while ensuring sufficient late updates.

The incorporation of MHA further refines this process by dynamically adjusting neuron updates based on attention-weighted relationships. Input data features serve as query vectors Q , while neuron weights act as keys K and values V . Attention scores are calculated as

$$\textit{Attention}(Q, K, V) = \textit{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V \quad (8)$$

where d_k is the dimensionality of the key vector. MHA divides these computations into multiple parallel “heads”, with each head capturing relationships across different feature dimensions. Outputs from all heads are concatenated and linearly transformed. This process enables MHA to model complex feature interactions that fixed neighborhood functions cannot address. The attention-weighted updates refine distance weights dynamically, ensuring adjustments remain effective throughout training. In MHA, the attention score for each query-key pair is calculated using the dot product between query vectors (Q) and key vectors (K). The query vectors represent the input data, and the key vectors represent neuron-specific weights. Each query vector is compared with all key vectors to compute attention scores. These attention scores are then normalized using a *softmax* function to ensure that they sum to one, producing a set of *attention weights*.

During competition, MHA improves winning neuron identification by modeling spatial and feature similarity. During cooperation, attention-weighted adjustments preserve topology while aligning with input distributions, enhancing precision for high-dimensional data.

The parameter *check_start* divides the process into an initial global alignment phase and a later convergence phase: The following two subsections elaborate on the distinct strategies employed in each of these critical phases. Section 3.2.1 details the initial stage, where the primary objective is to achieve a stable global topological mapping by suppressing large, disruptive updates to distant neurons. Section 3.2.2 then describes the convergence stage, which shifts the focus to high-precision, localized refinements by intensifying the updates for neurons in close proximity to the input cities. The dynamic parameterizations of the adaptive distance weights for both stages are explicitly formulated and explained.

3.2.1. Distance weights in the initial stage. During initial iterations ($t < \text{check_start}$), the algorithm balances neuron-input alignment with network stability by minimizing adjustments for distant neurons. The competition phase identifies the Best Matching Unit (BMU) via Multi-Head Attention (MHA), replacing Euclidean distance with relevance scores computed across feature dimensions. Inputs serve as queries (Q), neuron weights as keys/values (K/V), with scores calculated per Equation (8), capturing spatial-feature similarity. In cooperation, MHA dynamically guides neighbor updates: high-attention neurons near BMUs align precisely with inputs, while low-attention distant neurons undergo minimal adjustments, preserving topology. Adaptive distance weights, governed by nonlinear parameter p , further regulate updates:

$$\text{adaptive_weights} = \frac{1}{1 + \left(\text{attention_weights} \cdot \left(1 + \frac{\text{iteration}}{\text{check_start}} \right) \right)^p} \quad (9)$$

where *attention_weights* reflect neuron-input relevance. The parameter p evolves dynamically as

$$p = p_0 \cdot \left(1 + \left(\frac{\text{iteration}}{\text{check_start}} \right)^2 \right) \quad (10)$$

with initial value p_0 . MHA bridges competition and cooperation, weighting neuron relevance to refine BMU selection and neighbor updates. This ensures stable global alignment during initial optimization while preparing for convergence.

3.2.2. Distance weights in the convergence stage. During the convergence phase ($t \geq \text{check_start}$), a diminished nonlinear parameter q augments the adjustment velocity of neurons proximal to input cities, facilitating high-precision optimization. Multi-Head Attention (MHA) allocates elevated weight coefficients to BMU-adjacent neurons to enable fine-grained positional refinement. The adaptive weight integrates MHA-derived relevance metrics:

$$\text{adaptive_weights} = \frac{1}{1 + \text{attention_weights}^q} \quad (11)$$

where *attention_weights* quantify neuron-input affinity. The parameter q undergoes dynamic evolution per

$$q = q_0 \cdot \left(1 + \left(\frac{\text{total_iterations} - \text{iteration}}{\text{total_iterations} - \text{check_start}} \right)^2 \right) \quad (12)$$

with q_0 denoting its initial value. MHA intrinsically encodes feature-spatial correlations, supplanting static Gaussian kernels via the update operator:

$$W_j(t+1) = W_j(t) + \alpha(t) \cdot h_{ij} \cdot \text{distance_weights} \cdot \text{Attention}(Q, K, V) \cdot (X_j - W_j(t)) \quad (13)$$

Here, $Attention(Q, K, V)$ replaces the fixed Gaussian function h_{ij} , refining weight updates by combining geometric proximity and feature saliency. Consequently, BMU-vicinal neurons undergo pronounced positional refinement, whereas distal neurons exhibit attenuated adjustments to conserve topological integrity. Figure 4 delineates this dichotomy: the initialization phase (4(a)) imposes motion constraints on distal neurons (via reduced weights) to stabilize emergent topology and avert premature convergence; the convergence phase (4(b)) prioritizes BMU-centric neurons (via amplified weights) to minimize spatial residuals. This dual-phase paradigm equilibrates global structural preservation with local precision enhancement, culminating in optimized path accuracy.

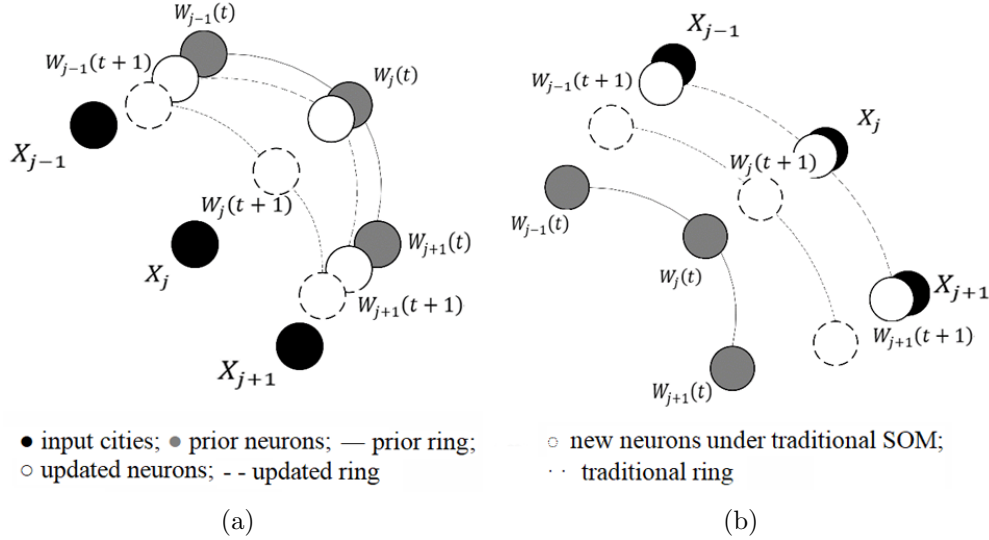


FIGURE 4. (a) Initial stage adaptation; (b) convergence stage

3.3. Graph Neural Network (GNN) aggregation mechanism. To resolve limitations of traditional neuron addition/deletion mechanisms – including local optima, inefficient resource utilization, and inflexibility in large-scale scenarios (e.g., ASOM’s fixed neurons requiring excessive iterations [17], and ISOM’s inadvertent removal of valuable neurons [31]) – this paper integrates a Graph Neural Network (GNN) aggregation mechanism into SOM’s cooperation phase, directly utilizing MHA-derived attention weights $Attention(Q, K, V)$ (Equation (13)) as feature inputs. GNN dynamically aggregates neighbor neuron features to update weights, optimizing topological adaptability and distribution. During cooperation, for each neuron i , its weight update incorporates aggregated neighbor features via

$$h_i^{(l)} = \sigma \left(W_1 h_i^{(l-1)} + W_2 \sum_{j \in \mathcal{N}(i)} \frac{h_j^{(l-1)}}{|\mathcal{N}(i)|} \right) \quad (14)$$

where $h_i^{(l)}$ represents the feature of neuron i at layer l , $\mathcal{N}(i)$ is the set of neighboring neurons for i , W_1 and W_2 are learnable weight matrices, and σ is an activation function. The input features $h_j^{(l-1)}$ implicitly encode MHA-computed affinities $Attention(Q, K, V)$, replacing static Gaussian functions. Post-BMU identification, GNN layers aggregate neighbor features; the resultant vector dynamically updates weights per Equation (13). GNN resolves uneven distribution: neurons in sparse regions leverage MHA-weighted global context for alignment, while those in dense regions refine positions locally without overfitting.

Figure 5 details this mechanism: neurons (nodes i) gather features from neighbors j , weighted by topological distance and MHA-computed affinities $Attention(Q, K, V)$. Features pass through a shared Meta-Policy module (learnable W) to fuse global distances and MHA-enhanced local features. Aggregated features $\mathcal{N}(i)$ update neuron weights, enabling sparse-region neurons to assimilate broader topological contexts and dense-region neurons to enhance resolution. Iterative GNN updates thus balance global structural consistency with MHA-guided local precision, mitigating local optima and resource inefficiency for relatively large-scale TSP. Neuron i aggregates features from neighbors j , weighted by topological distance and MHA-derived affinities $Attention(Q, K, V)$. The Meta-Policy module (transformation W) processes features, propagating aggregated outputs $\mathcal{N}(i)$ back to neurons. Sparse-region neurons utilize MHA-weighted global context; dense-region neurons refine locally [39].

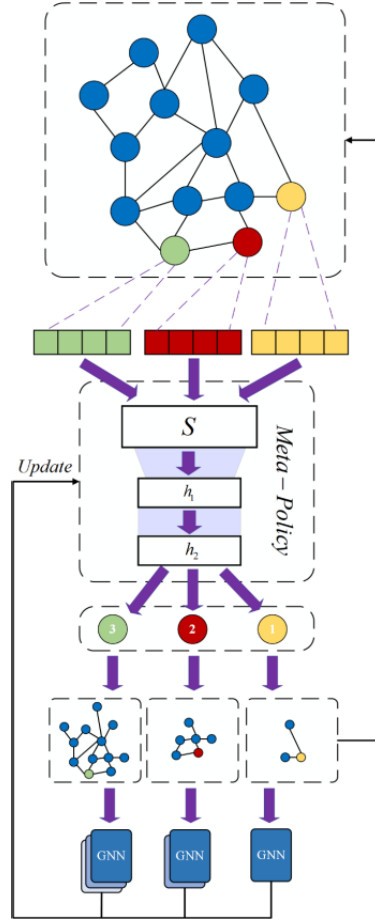


FIGURE 5. GNN aggregation mechanism

3.4. Optimized convergence criteria with masking mechanism. To overcome the limitations of fixed convergence thresholds in SOM (e.g., maximum iterations, minimal learning rate/neighborhood radius), a masking mechanism dynamically excludes converged neurons, reducing redundancy and enhancing efficiency. Converged neurons are identified via distance error to nearest inputs and negligible weight changes over iterations, with thresholds adapting to data distribution shifts. Convergence checks commence post-stabilization, governed by a dynamic threshold:

$$check_start = \max(p_{\min}, \alpha \cdot N_{net}) \cdot \left(1 - \frac{t}{T_{\max}}\right) \quad (15)$$

where $p_{\min}$ prevents premature checks, α is a scaling factor, N_{net} represents the network size, t current iteration, and $T_{\max}$ total iteration limit. Stratified sampling optimizes checks: cities are K-means clustered, with representative subsets preserving data integrity while lowering computational load. Masked neurons are excluded from path improvement calculations:

$$\Delta d = L_{\text{prev}} - L_{\text{curr}} \quad (16)$$

where L_{prev} is prior path length and L_{curr} current length. The check interval Δt_{i+1} adapts to improvement rate:

$$\Delta t_{i+1} = \max(100, \min(2000, \Delta t_i \cdot (1 + \gamma \cdot (\text{improvement} - \theta)))) \quad (17)$$

where γ controls the adjustment range, θ is a preset threshold, and ΔL is computed as

$$\Delta L = \frac{\Delta d}{L_{\text{prev}}} \quad (18)$$

($\Delta t = 0$ if $\Delta L \leq 0$). Active neurons update per decaying learning rate $\eta(t_{i+1})$ and neighborhood radius $\sigma(t_{i+1})$:

$$\eta(t_{i+1}) = \eta(t_i) * 0.99997 \quad (19)$$

$$\sigma(t_{i+1}) = \sigma(t_i) * 0.99997 \quad (20)$$

Termination occurs when (i) maximum iterations reached; (ii) $\eta(t)$ and $\sigma(t)$ decay to minima; (iii) masking confirms neuron stabilization per cluster; or (iv) path length reduction plateaus.

Figure 6 illustrates masking: initially, all neurons participate for global alignment (top). Inputs X transform via adaptive weights W_{adapt} , with Masked Attention (calculated via

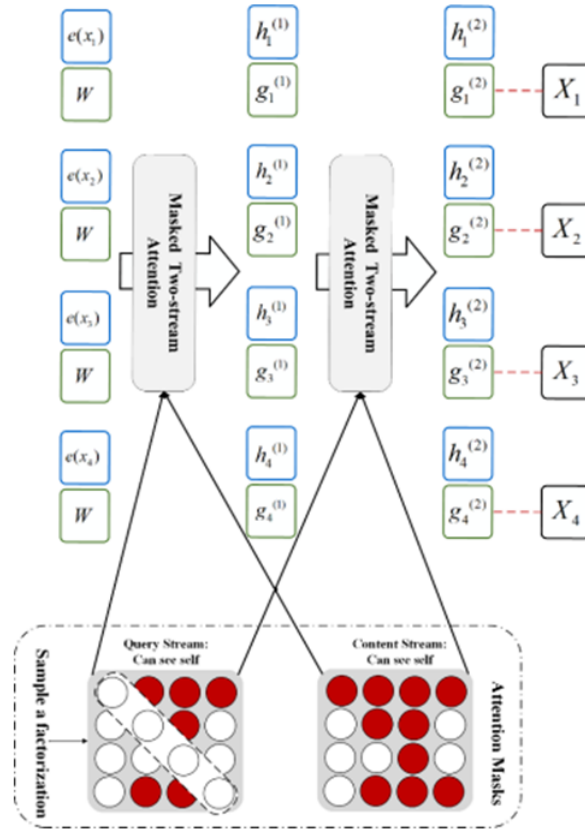


FIGURE 6. (color online) Masked attention mechanism

Equation (8)) excluding converged neurons (middle, red regions). Finally, only active neurons undergo refinement for local precision (bottom), balancing global exploration and local optimization while minimizing overhead.

The integrated methods described above can be summarized in Algorithm 1 as follows, with the corresponding algorithmic workflow illustrated in Figure 7: The proposed Context-Aware Adaptive Self-Organizing Map (CASOM) implements a hierarchical

Algorithm 1: CASOM

Input: $m_f, m_1, m_2, \dots, m_n$ (city data)

Output: $e_r^N, e_1^N, e_2^N, \dots, e_n^N$ (final set of neurons after training)

1: **procedure** Initialize Neurons

2: Perform K-means clustering to initialize neuron distribution based on city data $m_f, m_1, m_2, \dots, m_n$, according to Equations (5) and (6)

3: Apply Position Encoding (PE) to neurons and combine it with input features for spatial alignment, as shown in Equation (4)

4: Use density-based neuron allocation with PE to adjust neuron count based on cluster density, as per Equation (7)

5: **end procedure**

6: $t = 1$

7: **while** convergence criteria not met **do**

8: **procedure** Competition Phase

9: Select winner neuron based on the distance (Euclidean distance) to input data, as calculated in Equations (1) and (8)

10: **end procedure**

11: **procedure** Cooperation Phase with GNN and MHA

12: Apply Graph Neural Network (GNN) to adjusting adaptive distance weights based on relationships between neurons and their neighbors, as represented in Equations (9)-(12) and (14)

13: Apply Multi-Head Attention (MHA) to dynamically adjusting attention weights for neuron updates, allowing more relevant neurons to influence the update process, according to Equations (13) and (14)

14: **end procedure**

15: **procedure** Masking

16: Mask converged neurons (neurons with minimal weight changes or small distance errors from input points), and skip their updates, as outlined in Equations (15)-(20)

17: **end procedure**

18: $t = t + 1$

19: **end while**

20: **procedure** Training Termination

21: Terminate training based on optimized convergence condition (e.g., when weight changes are minimal or error falls below threshold), as per Equations (15)-(20)

22: If the termination condition met, proceed to return final neuron set

23: **end procedure**

24: **procedure** Final Path Formation

25: Return final neuron set representing the optimized path after training: $e_r^N, e_1^N, e_2^N, \dots, e_n^N$ (final set of neurons in the optimized path)

26: **end procedure**

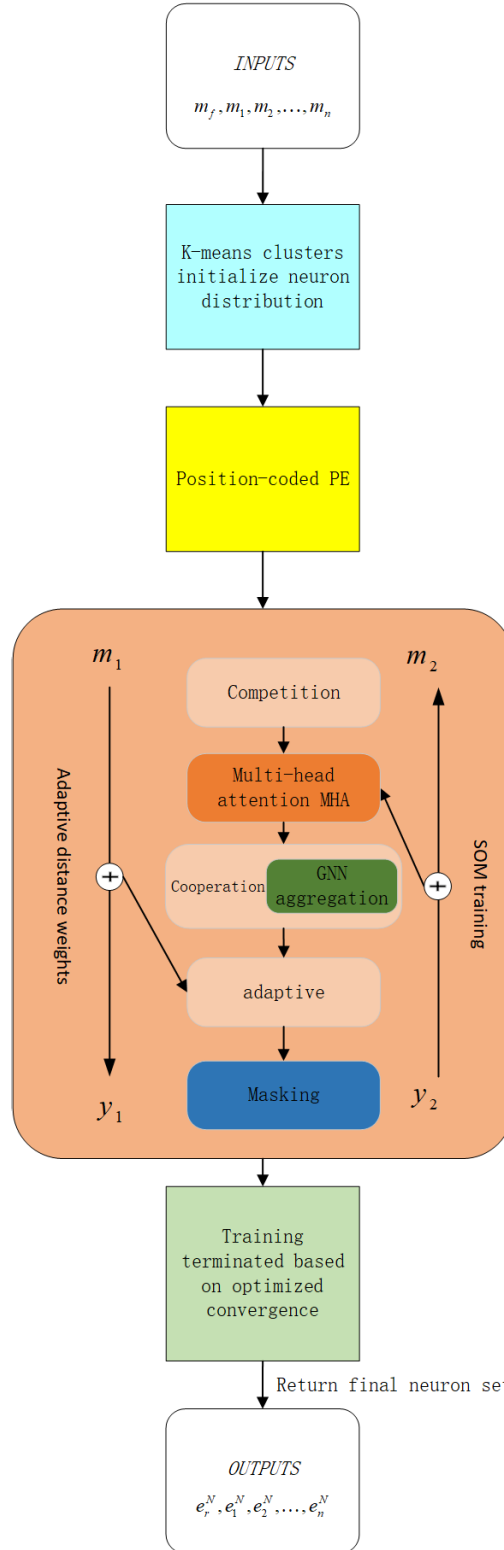


FIGURE 7. Flowchart of the algorithm

multi-head attention-based adaptive graph neural network model. The process begins by initializing the neuron ring through K-means clustering combined with Positional Encoding (PE), which embeds spatial information to ensure a structured alignment with the input city nodes $m_f, m_1, m_2, \dots, m_n$ and dynamically allocates neurons based on data density (Algorithm 1, lines 1-5). The core training loop then alternates between key phases.

In the competition phase, the Best Matching Unit (BMU) is selected by fusing distance-based metrics with attention-based relevance scores computed by the Multi-Head Attention (MHA) mechanism (Algorithm 1, lines 8-10). In the subsequent cooperation phase, the MHA-derived attention weights guide the Graph Neural Network (GNN) aggregation, which propagates PE information to update and preserve both global and local topological structures (Algorithm 1, lines 11-14). Concurrently, a masking mechanism is applied to excluding converged neurons, and stratified sampling with dynamic threshold checks is employed to minimize redundant updates and standardize the convergence process (Algorithm 1, lines 15-17). This cycle of SOM training continues until the optimized termination criteria are met. The final neuron ring, comprising neurons $e_r^N, e_1^N, e_2^N, \dots, e_n^N$, represents the approximate solution to the TSP (Algorithm 1, lines 20-26).

4. Experiments and Results. This section provides a detailed description of the experimental setup, datasets, comparison benchmarks, evaluation metrics, and experimental results. By comparing CASOM with several classical SOM-based algorithms, heuristic algorithms, and deep reinforcement learning methods in solving the Traveling Salesman Problem (TSP), CASOM's performance advantages in terms of solution accuracy, efficiency, and error reduction are validated. Through performance analysis and comparison, CASOM's advantages in achieving higher accuracy and efficiency are demonstrated.

4.1. Experimental setup. The experiments were conducted on an AMD Ryzen 5 7530U processor with Radeon Graphics @ 2.00 GHz, and 16 GB of physical memory. The development environment was PyCharm, using Python as the programming language. As shown in Table 1, the key initial parameters for CASOM are summarized.

To validate CASOM's effectiveness, comparisons were made with algorithms including SOM, KNIES, ESOM, ASOM, SA, GA, PSO, ACO, and TRANSFORMER. Each algorithm was tested 100 times on TSP instances sourced from TSPLIB [32]. The results

TABLE 1. Key initial parameter settings in CASOM

Parameter	Description	Value
Learning rate	Step size for weight updates, gradually decreasing	0.8
n_heads	Number of attention heads	8.0
p_{min}	Start checking after sufficient iterations	1000
Max. iterations	Maximum number of iterations	200000
α	Adjusts check threshold and interval dynamically	5
β	Controls dynamic adjustment of check interval	0.01
Improvement threshold	Threshold for significant distance improvement	1e-4
No improvement threshold	Max. non-improvement iterations before termination	5
Max. clusters (K)	Max. number of clusters in K-means	10
K-means iterations (m)	Iterations for K-means clustering	10
batch_size	Number of instances per batch	256
Sample ratio	Proportion for stratified sampling	0.1
Tint	Cycle for neuron adjustment and deletion	50
p_0	Nonlinear adjustment during the initial phase	2
q_0	Nonlinear adjustment during the convergence phase	2
Check interval	Interval for checking improvements	1000

obtained by CASOM were compared both with the optimal solutions published in TSPLIB and the best-known solutions reported in [12,13,16,17,33-40] for similar methods. This dual comparison approach rigorously benchmarks CASOM's performance against both established datasets and previously documented results, thereby enhancing the robustness of the conclusions. All data used in the experiments were two-dimensional, and distances were calculated using the Euclidean metric.

4.2. Datasets and selection rationale. This study employs the TSPLIB benchmark, the predominant standard dataset in the TSP domain, to ensure a fair and reproducible evaluation of the proposed CASOM algorithm. TSPLIB, established and maintained by the University of Heidelberg, is widely recognized as an international standard for comparing TSP and related combinatorial optimization algorithms [32]. The selection of specific instances from TSPLIB was guided by a multifaceted strategy aimed at a comprehensive and scientifically rigorous assessment, based on the following deliberate considerations.

First, regarding scalability and computational burden, to systematically evaluate the algorithm's performance scaling with problem size, we selected instances forming a progressive size gradient. This range extends from small-scale (e.g., *eil51* with 51 nodes) to large-scale (e.g., *pr2392* with 2,392 nodes). Such a progression is critical for objectively assessing the CASOM's computational efficiency and its ability to handle increasing problem complexity, a common challenge in NP-hard problems.

Second, concerning the diversity of instance characteristics, the chosen set encompasses instances with heterogeneous spatial distributions and structural features to test the robustness and generalizability of CASOM. This includes real-world geographic layouts, where instances like *berlin52*, *ch130*, and *pcb442* are derived from real-world city coordinates or circuit board drilling problems, embedding practical, irregular spatial patterns. It also includes synthetic but structured networks, where instances such as *kroA100* and *lin105* provide examples of structured but synthetically generated challenges. This mix ensures the algorithm is tested against a spectrum of spatial data distributions, from clustered points to more uniform dispersions, preventing overfitting to a single problem type.

Third, for established benchmarking and comparability, all selected instances are extensively used and well-established within the operational research and computer science literature. For instance, the *eil51*, *berlin52*, and *kroA100* instances are frequently cited in numerous seminal and contemporary studies on TSP. This common ground allows for a direct and unequivocal comparison of CASOM's performance against a wide array of established methods, including other SOM-based variants, classical heuristics (e.g., Genetic Algorithms, and Ant Colony Optimization), and modern learning-based approaches. The use of these universally accepted benchmarks solidifies the validity and contribution of our experimental conclusions.

Based on the above rationale, the final selection comprises the following TSPLIB instances: *eil51*, *berlin52*, *st70*, *kroA100*, *lin105*, *ch130*, *ch150*, *kroa200*, *pcb442*, *pr1002*, *vm1748*, and *pr2392*. This curated portfolio covers a wide range of problem sizes, from 51 to 2392 nodes, and includes both real-world geographical layouts and randomly generated complex network paths, thereby establishing a balanced and challenging testing framework for benchmarking the CASOM algorithm against the state-of-the-art.

4.3. Evaluation metrics. The performance of the model is evaluated using the following metrics: best path length, worst path length, average path length, average processing time, and the optimality gap [33,34].

The best path length represents the shortest path across all test instances. The worst path length indicates the longest path. The average path length is the mean over 100

tests per instance, calculated by Equation (21). The average processing time denotes the mean runtime per instance for computational efficiency assessment. The optimality gap, computed via Equation (22), quantifies the deviation from the theoretical optimum.

$$\text{Average Length} = \frac{1}{n} \sum_{i=1}^n \text{length}_i \quad (21)$$

$$\text{Optimality gap (\%)} = \left(\frac{\text{average length}}{\text{optimal length}} - 1 \right) \times 100\% \quad (22)$$

The *optimality gap* indicates the quality of the current solution; a smaller gap indicates that the solution is closer to the optimum, while a larger gap indicates that the solution significantly deviates from the optimal solution.

These metrics provide a comprehensive evaluation of the algorithm’s performance across different problem instances, offering valuable insights for further algorithm improvement and optimization.

4.4. Experimental results. The experimental results for different TSP problems across the tested algorithms are summarized in Table 2. In Table 2, the best results for each metric are highlighted in bold, while the second-best results are indicated with underlines. The results in Table 2 show that CASOM achieves the optimal path in 9 out of 12 TSP instances and finds the suboptimal path in the remaining 3 instances. For smaller-scale instances, CASOM performed comparably to KNIES, ESOM, and ASOM, and in some cases, it even outperformed TRANSFORMER. For example, CASOM found the optimal path in the berlin52 instance, and its results on the eil51 instance were nearly identical to those of other SOM-based algorithms. Additionally, for the st70 instance, CASOM’s average path length was only 0.43% longer than that of ACO, while its processing time was significantly faster.

As the instance size increases, particularly when the number of nodes exceeds 100, the advantages of CASOM become increasingly pronounced. For example, in the kroA100 instance, CASOM achieved an optimality gap of 0.99%, and it also found the optimal paths in the lin105, ch130, ch150, and kroa200 instances. Moreover, its average runtime was significantly shorter than that of other SOM-based algorithms. Although ACO demonstrated superior solution quality, its processing time was considerably longer. As the problem size increased, the efficiency of ACO lagged behind that of CASOM and other SOM-based algorithms.

In relatively large-scale instances, the prolonged runtime of certain heuristic algorithms diminishes their practicality for meaningful comparisons. Consequently, the focus in these instances shifted towards evaluating CASOM against other SOM-based algorithms. The results indicate that CASOM delivers significant enhancements in both solution quality and stability, alongside marked improvements in computational efficiency. Specifically, in the pcb442, pr1002, vm1748, and pr2392 instances, CASOM outperformed comparable SOM algorithms, achieving an average optimality gap improvement ranging from 2.56% to 5.00%, while reducing runtime by 56.63% to 88.61%. Based on the simulation results presented in Table 2, Figures 8 and 9 illustrate comparison curves for the average optimality gap and average runtime across different city scales for various algorithms. These figures provide a more intuitive visualization of CASOM’s advantages, especially in larger-scale datasets.

To further assess the contributions of key components in CASOM, an ablation study was conducted by individually removing Positional Encoding (PE), Multi-Head Attention (MHA), Graph Neural Network (GNN), and the Masking mechanism, evaluating their impact across problem instances of size 50, 100, and 200. The results demonstrate that

the absence of any of these components leads to performance degradation, confirming their essential role in improving convergence and solution quality.

TABLE 2. Comparative experimental results of CASOM and various TSP algorithms

Instances	Optimal length	Algorithm	Best length	Worst length	Average length	Average processing time	Optimality gap
eil51	426	SOM	437.56	444.89	441.51	12.08	3.64%
		KNIES	430.66	449.94	437.97	<u>0.64</u>	2.81%
		ESOM	430.36	444.4	439.12	0.58	3.08%
		ASOM	430.11	440.32	435.16	5.78	2.15%
		SA	438.97	463.03	451.00	60.37	5.87%
		GA	435.19	457.27	445.87	145.67	4.68%
		PSO	450.08	491.66	469.57	12.75	10.21%
		ACO	441.4	993.71	447.96	502.03	5.17%
		TRANSFORMER	435.24	448.69	438.00	\	2.82%
		CASOM	<u>430.24</u>	444.71	<u>437.47</u>	3.96	<u>2.69%</u>
berlin52	7542	SOM	7629.74	8344.72	7963.04	12.81	5.58%
		KNIES	7545.24	8374.45	7958.84	\	5.53%
		ESOM	7545.16	8370.56	7808.99	\	3.54%
		ASOM	<u>7544.75</u>	7970.4	<u>7756.2</u>	<u>5.58</u>	<u>2.88%</u>
		SA	7623.45	9585.8	8604.62	67.34	14.09%
		GA	7847.06	8897.46	8257.3	154.32	9.48%
		PSO	7894.25	8821.26	8218.71	14.56	8.97%
		ACO	7545.6	7680.8	7611.4	560.35	0.92%
		TRANSFORMER	7802.00	8001.43	8212.56	\	6.10%
		CASOM	7544.36	8064.84	7794.6	3.94	3.35%
st70	675	SOM	684.92	697.95	690.48	12.39	2.43%
		KNIES	683.01	698.86	690.93	<u>1.15</u>	2.36%
		ESOM	682.99	695.22	689.11	1.13	2.09%
		ASOM	683.12	696.85	689.92	5.33	2.21%
		SA	688.98	689.90	689.45	\	2.14%
		GA	688.14	773.21	719.93	210.75	6.66%
		PSO	745.95	837.86	779.11	15.5	15.42%
		ACO	682.99	686.11	684.55	1087.2	1.41%
		TRANSFORMER	681.00	740.45	710.34	\	5.19%
		CASOM	<u>682.98</u>	693.87	<u>687.42</u>	4.93	<u>1.84%</u>
kroA100	21282	SOM	21322.73	23062.51	22690.62	14.82	6.62%
		KNIES	\	\	\	\	\
		ESOM	21320.59	22594.18	21952.38	\	3.15%
		ASOM	21321.32	22627.75	21971.54	<u>9.34</u>	3.24%
		SA	22128.87	23261.81	22695.34	117.75	6.65%
		GA	\	\	\	\	\
		PSO	\	\	\	\	\
		ACO	22347.07	57559.49	22533.73	3410.47	5.89%
		TRANSFORMER	<u>21320.31</u>	22176.69	<u>21747.00</u>	\	<u>2.18%</u>
		CASOM	21285.44	21725.18	21495.31	6.76	0.99%
lin105	14379	SOM	14524.51	14855	14788.76	15.46	2.85%
		KNIES	<u>14403.51</u>	14981.41	14692.46	3.2	2.18%
		ESOM	14454.58	14961.98	14708.28	<u>3.35</u>	2.29%
		ASOM	14418.97	14537.53	14478.22	10.77	0.69%
		SA	\	\	\	\	\
		GA	\	\	\	\	\
		PSO	\	\	\	\	\
		TRANSFORMER	14932.00	15225.64	15519.29	\	5.89%
		ACO	14413.25	14656.75	<u>14535</u>	3563.23	<u>1.08%</u>
		CASOM	14382.99	14616.04	14539.51	6.41	1.16%

(Continued)

Instances	Optimal length	Algorithm	Best length	Worst length	Average length	Average processing time	Optimality gap
ch130	6110	SOM	6362.96	6410.72	6384.84	16.48	4.52%
		KNIES	\	\	\	\	\
		ESOM	6346.91	6481.65	6414.28	\	4.98%
		ASOM	6291.96	6360.63	6326.29	11.06	3.54%
		SA	6379.01	6758.53	6568.77	167.87	7.51%
		GA	\	\	\	\	\
		PSO	\	\	\	\	\
		ACO	6335.71	15651.72	6435.65	7855.7	5.32%
		TRANSFORMER	6203.04	6850.98	6569.01	\	7.51%
ch150	6528	CASOM	6179.31	6358.31	6248.81	6.71	2.27%
		SOM	6756.55	6964.55	6859.55	16.75	5.08%
		KNIES	\	\	\	\	\
		ESOM	6742.37	6891.03	6816.7	\	4.43%
		ASOM	6734.58	6812.18	6773.38	12.06	3.77%
		SA	\	\	\	\	\
		GA	\	\	\	\	\
		PSO	40082	46396	43239	170.34	562.10%
		ACO	6699.85	6647.15	6673.5	9426.84	2.23%
kroa200	29368	TRANSFORMER	6825.73	8045.91	7390.32	\	13.20%
		CASOM	6604.01	6766.92	6675.46	9.3	2.26%
		SOM	30459.12	33733.44	31595.28	16.85	7.58%
		KNIES	30005.42	33122.58	31564.03	21.8	5.66%
		ESOM	30438.39	31564.58	30785.05	23.66	2.91%
		ASOM	29852.78	30481.8	30147.29	9.19	2.65%
		SOM	55143.11	62857.46	59000.28	21.99	16.20%
		KNIES	54975.32	56782.48	55878.9	49.84	9.86%
		ESOM	54912.24	56538.2	55756.76	54.78	9.79%
pcb442	50778	ASOM	53934.37	55453.62	55214.47	16.74	8.62%
		SA	\	\	\	\	\
		GA	102433.37	117131.78	113772.13	\	123.90%
		PSO	204008.95	220222.3	211268.05	\	316.00%
		ACO	\	\	\	\	\
		CASOM	53511.76	56164.03	54307.89	15.33	6.96%
		SOM	278616.04	301070.62	289842.33	38.99	11.90%
		KNIES	277536.29	278850.99	278400.62	170.86	7.47%
		ESOM	277248.17	301135.33	289542.79	168.18	11.64%
pr1002	259045	ASOM	276198.25	286723.29	282336.77	28.75	8.97%
		CASOM	275328.93	279957.33	276903.13	27.14	6.89%
		SOM	373329.09	378108.8	375716.94	61.44	11.63%
		KNIES	362586.49	369853.23	366268.37	461.71	8.84%
		ESOM	354501.97	364560.25	362226.11	505.05	7.63%
		ASOM	361950.57	376728.39	368339.32	43.01	9.50%
		CASOM	356928.23	364894.16	359911.19	40.28	6.94%
		SOM	423453.32	437742.36	428871.18	80.32	13.48%
		KNIES	419950.25	423225.11	421587.68	823.76	11.50%
pr2392	378032	ESOM	414568.78	434739.11	427344.68	910.84	13.09%
		ASOM	413458.75	418059.69	419004.97	56.09	9.53%
		CASOM	400111.57	408925.8	404518.68	53.21	7.00%

Figure 10 presents the comparative results of the ablation study. When PE was removed, the initial neuron distribution lacked geometric information embedding, leading to sub-optimal neuron placement in large-scale TSP instances. This resulted in a 3.41%-5.02% decline in optimization quality and an increase in convergence time by 18.27%-25.36%. The absence of MHA reduced local search capability, limiting the model's ability to escape local optima. Consequently, the optimality gap increased by 2.68%-4.91%, while runtime extended by 15.12%-22.43%. The removal of GNN impaired neighborhood aggregation, causing suboptimal neuron distribution, especially in complex topologies, increasing the

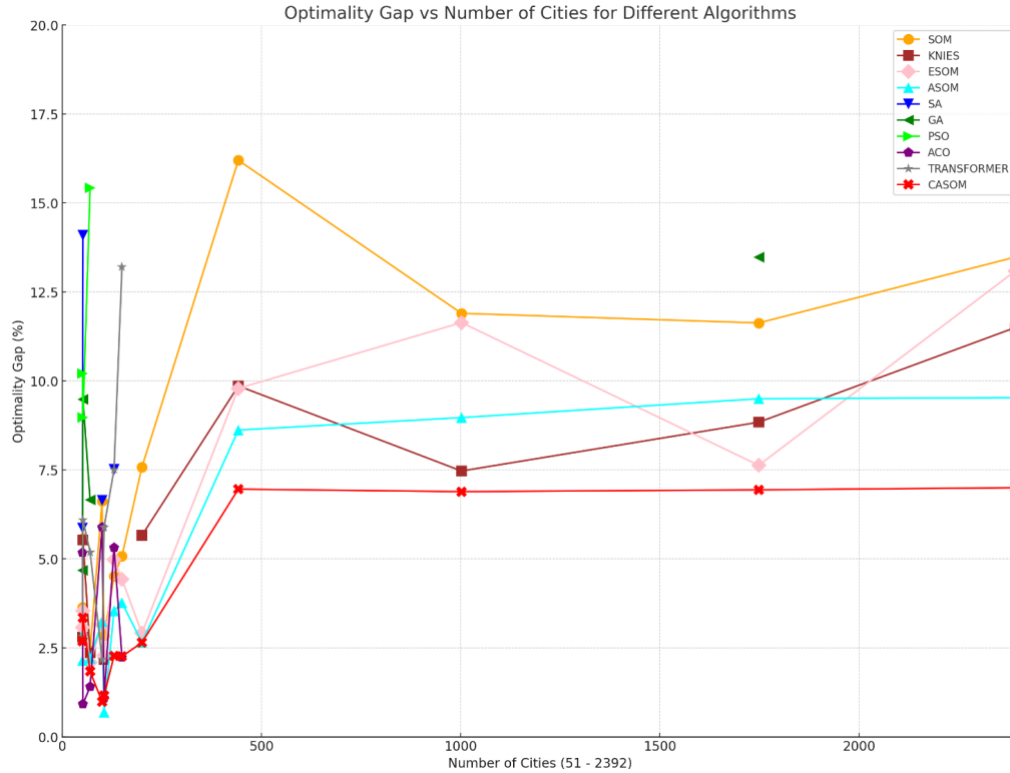


FIGURE 8. Comparative curves of average optimality gaps across various algorithms

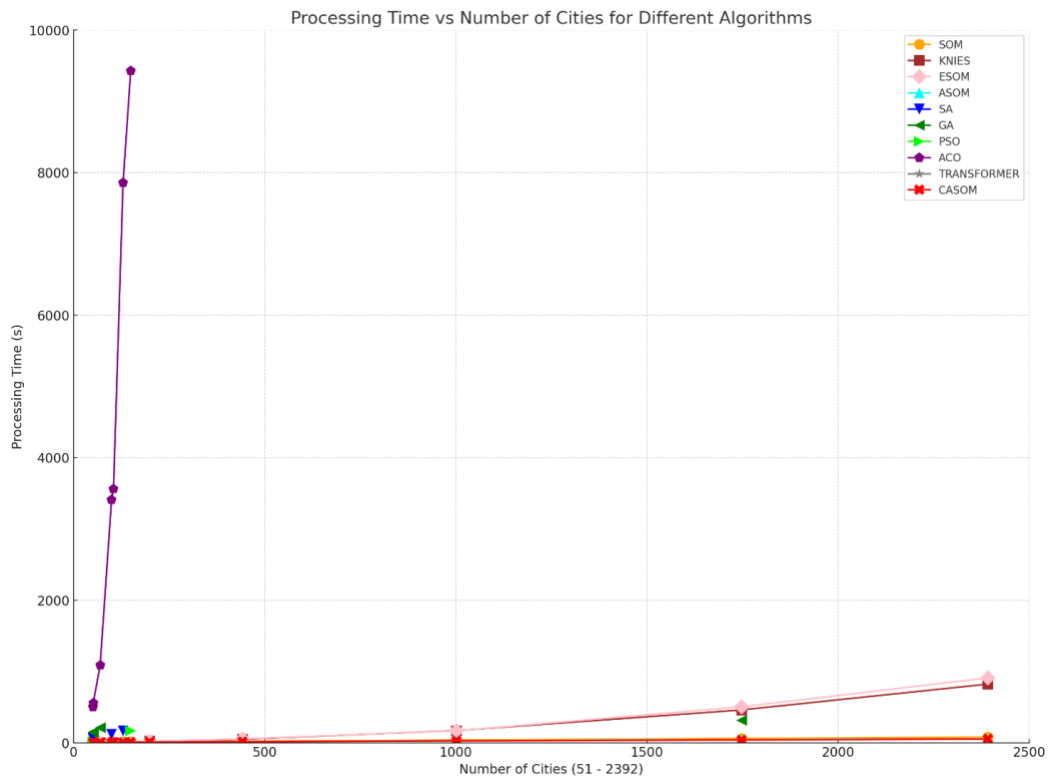


FIGURE 9. Comparative curves of average processing times across various algorithms

optimality gap by 3.52%-6.89%. Finally, eliminating Masking led to inefficient computational resource allocation, significantly increasing redundant calculations and prolonging convergence, with an average runtime extension of 23.74%-39.15%.

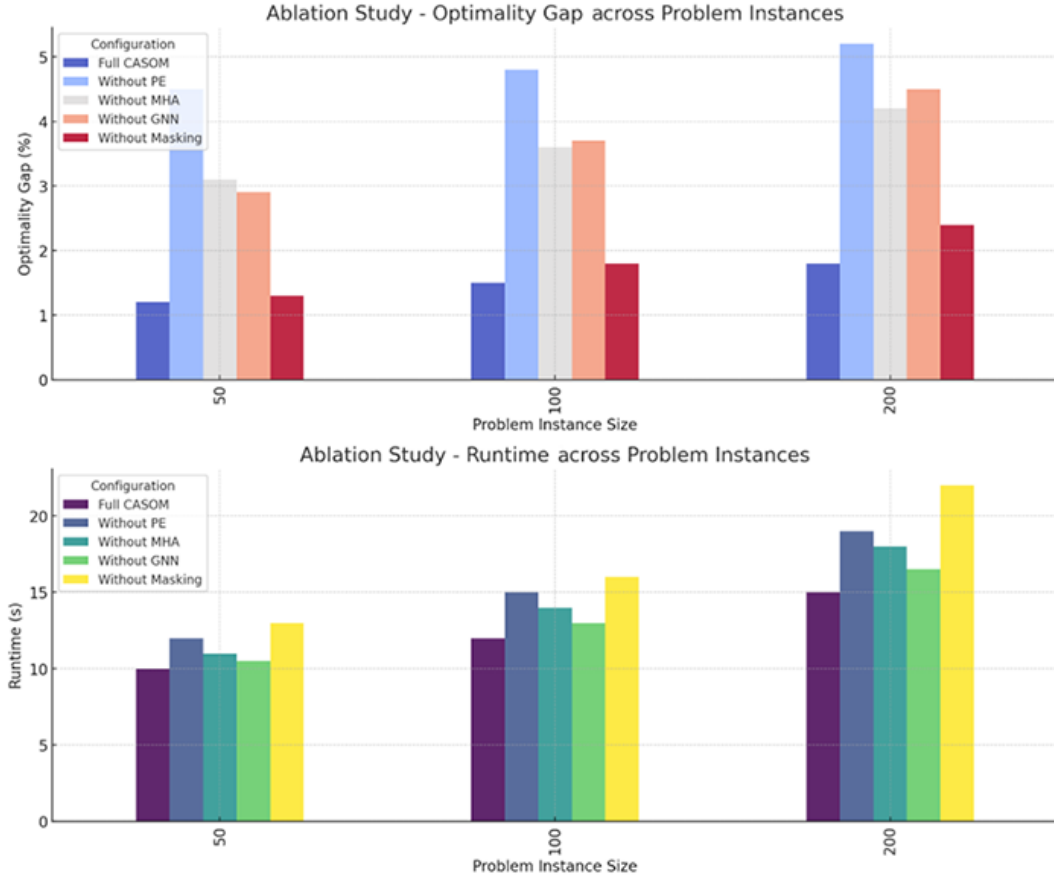


FIGURE 10. Ablation study on CASOM components across different problem sizes

The ablation study further confirms that PE optimizes spatial distribution during neuron initialization, MHA dynamically adjusts weights through attention mechanisms, GNN preserves topological relationships, and Masking reduces computational redundancy to accelerate convergence.

Taking the kroA100 instance as an example, Figure 11 illustrates the process of solving the TSP problem using CASOM. The algorithm begins by initializing neurons based on clustered convex hull properties, optimizing the initial neuron distribution with Positional Encoding (PE). This structured initialization leverages K-means clustering enhanced by PE, where the Elbow Method determines the optimal number of clusters to be two, thereby achieving a density-aware spatial allocation of neurons aligned with the underlying city distribution. As shown in Figure 11(a), this PE-guided strategy effectively embeds the underlying city geometry, positioning the neurons within the two primary data clusters from the outset, which mitigates the pitfalls of random initialization and provides a structured starting point for convergence.

At iteration 3000, GNN aggregation is applied to dynamically adjusting the neighborhood information. The GNN operates by aggregating Positional Encoding (PE) information from neighboring nodes, which facilitates a more stable and efficient topological organization, effectively bypassing the traditional SOM's chaotic transition from disorder to order. The result (Figure 11(b)) demonstrates the GNN's role in topological smoothing, where the neuron ring is coherently pulled towards the city points while preserving its global continuity, thereby accelerating the transition from a disordered state.

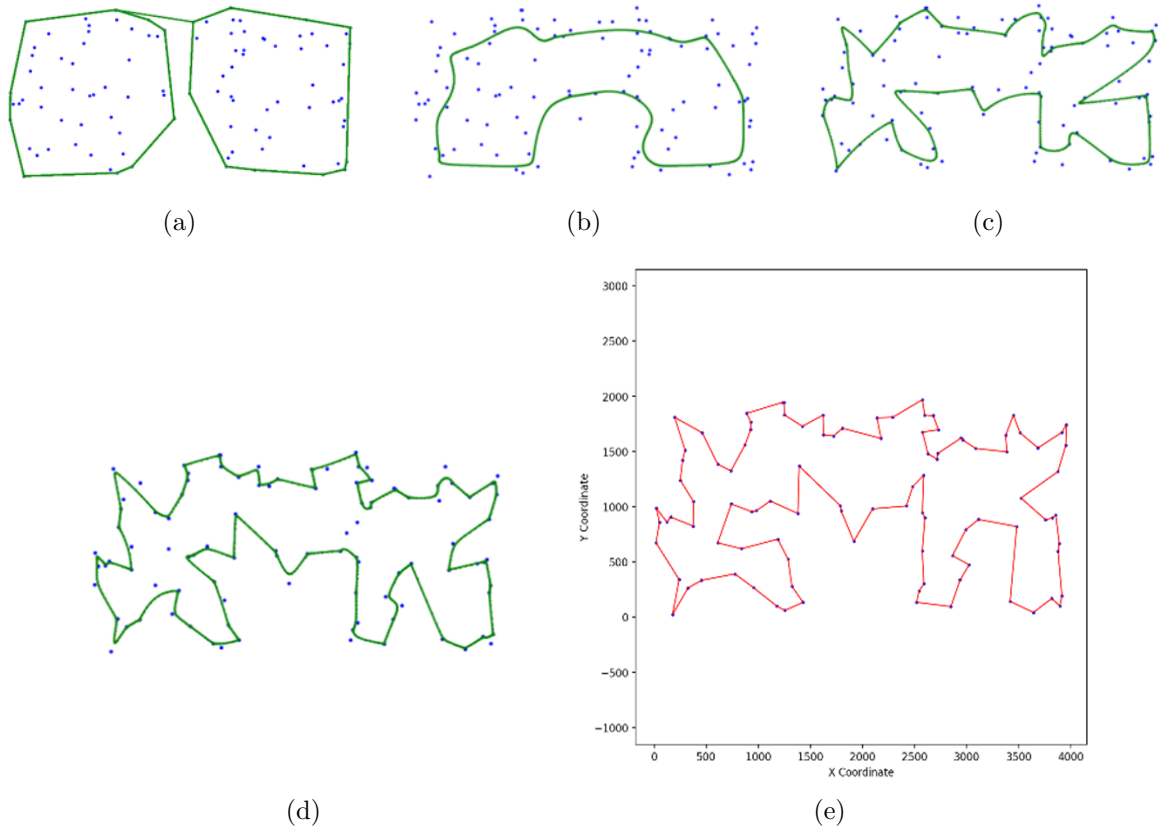


FIGURE 11. (a) Initial neural network for kroA100 at iteration 0; (b) neural network for kroA100 after 3000 iterations; (c) neural network for kroA100 after 6000 iterations; (d) neural network for kroA100 after 7000 iterations in the convergence phase; (e) final TSP solution for kroA100

By iteration 6000, the Multi-Head Attention (MHA) mechanism becomes instrumental in guiding the neuron updates. During this phase, MHA dynamically computes attention weights to guide the granularity of distance control in neuron updates. This context-aware adjustment ensures effective updates in critical regions while minimizing redundant computations. This phase (Figure 11(c)) showcases MHA's ability to refine the mapping through context-aware, localized adjustments, sharpening the path definition without disrupting the established global topology.

By iteration 7000, the network enters its convergence phase, where the masking mechanism excludes converged neurons. Converged neurons are identified by assessing stable MHA weights and minimal PE distance errors to input points, enabling the masking mechanism to dynamically exclude them from subsequent updates. This prevents both over-fitting and under-fitting, facilitating an adaptive and efficient convergence. The final state (Figures 11(d) and 11(e)) highlights the synergy of all components: the initial structure is preserved, the path is topologically sound, and precise local refinements have eliminated crossovers, resulting in a clean, high-quality tour.

Figure 12 displays CASOM's final results on selected instances, providing crucial visual evidence that complements the quantitative metrics in Table 2 and Figures 8 and 9. The generated paths are not only shorter in length but also visually superior in quality compared to those typically produced by traditional SOMs or heuristic methods. For instance, the solutions for berlin52 (Figure 12(a)) and st70 (Figure 12(b)) are exceptionally

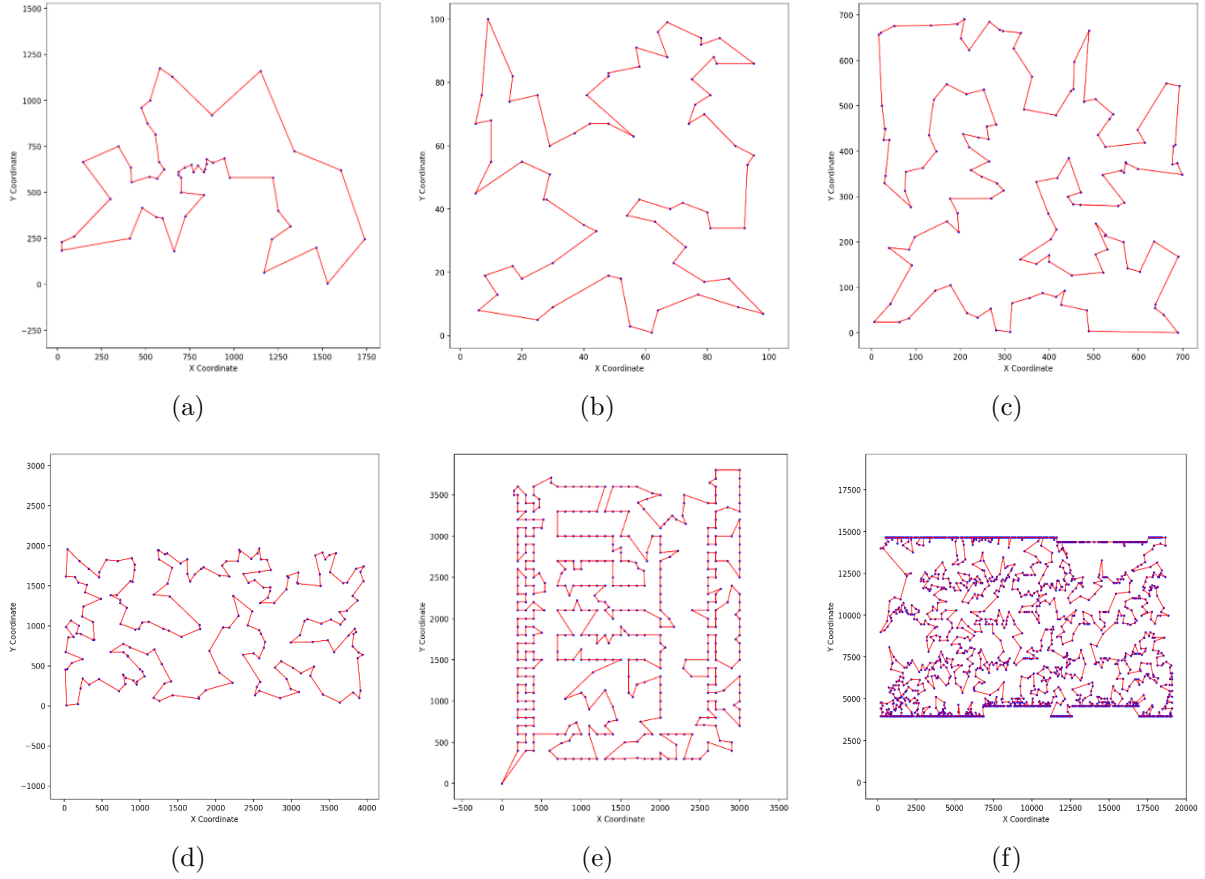


FIGURE 12. (a) CASOM solution for berlin52; (b) CASOM solution for st70; (c) CASOM solution for ch130; (d) CASOM solution for kroA200; (e) CASOM solution for pcb442; (f) CASOM solution for vm1748

smooth and adhere to the known optimal tour shapes. More importantly, for larger-scale instances like pcb442 (Figure 12(e)) and vm1748 (Figure 12(f)), CASOM successfully constructs clean, plausible routes without the visible crossovers or topological irregularities that often plague other methods in complex layouts. This visual integrity across diverse problem scales and distributions underscores CASOM's enhanced robustness and explains its superior performance in the quantitative comparisons. CASOM adapts more effectively to varying problem scales and distributions than other algorithms, as quantitatively demonstrated in previous sections, with these visualizations serving as direct proof of that capability.

Figure 8 illustrates the performance comparison of different algorithms in terms of average optimal gap. These curves show how each algorithm performs in various TSP instances, highlighting the relative efficiency of CASOM in maintaining a low optimal gap compared to other methods.

Figure 9 presents the comparative analysis of average processing times for different algorithms. The curves demonstrate how each algorithm performs in terms of computational efficiency across various TSP instances, highlighting the relatively faster processing speed of CASOM compared to other methods.

Figure 10 presents the results of the ablation study evaluating the impact of key components – Positional Encoding (PE), Multi-Head Attention (MHA), Graph Neural Network (GNN), and the Masking mechanism – on CASOM's performance across problem instances of sizes 50, 100, and 200.

5. Conclusion. This paper proposes a Context-Aware Adaptive Self-Organizing Map (CASOM) algorithm for solving relatively large-scale Traveling Salesman Problem (TSP). CASOM addresses key limitations of traditional SOM methods, such as random initialization, neuron allocation inefficiency, and fixed convergence conditions. A density-based neuron allocation strategy integrating Positional Encoding (PE) is introduced, optimizing initial neuron distribution by embedding spatial geometry and dynamically adjusting neuron counts based on cluster density.

During training, adaptive neuron updates are driven by Multi-Head Attention (MHA) and Graph Neural Network (GNN) aggregation. MHA dynamically calculates attention-weighted neuron relevance scores, refining neuron selection and update magnitudes. Simultaneously, GNN aggregation incorporates neighborhood information, ensuring neuron adjustments maintain global topological consistency and local precision. A masking mechanism further optimizes computational efficiency by dynamically excluding neurons that have converged, identified through stratified sampling and dynamic threshold checks.

Experimental evaluation demonstrates that CASOM significantly improves neuron alignment accuracy, computational efficiency, and convergence stability compared to traditional SOM methods. Future research directions include scaling CASOM to larger combinatorial optimization tasks such as Vehicle Routing Problem (VRP), exploring advanced GNN architectures, and potential integration with Deep Reinforcement Learning (DRL) for enhanced performance and broader applicability.

Acknowledgement. This research was supported by the Yunnan Provincial Science and Technology Talent and Platform Program (Academician Expert Workstation) (Grant/Award Number: 202405AF140077).

REFERENCES

- [1] Y. Shi and Y. Zhang, The neural network methods for solving traveling salesman problem, *Procedia Computer Science*, vol.199, pp.681-686, DOI: 10.1016/J.PROCS.2022.01.084, 2022.
- [2] F. J. Gil-Gala, M. Durasević, M. R. Sierra and R. Varela, Evolving ensembles of heuristics for the travelling salesman problem, *Natural Computing*, vol.22, no.4, pp.671-684, DOI: 10.1007/S11047-023-09968-9, 2023.
- [3] W. Qian, R. A. M. Basili, M. M. Eshaghian-Wilner et al., Comparative study of variations in quantum approximate optimization algorithms for the traveling salesman problem, *Entropy*, vol.25, no.8, 1238, DOI: 10.3390/E25081238, 2023.
- [4] J. Sui, S. Ding, X. Huang et al., A survey on deep learning-based algorithms for the traveling salesman problem, *Frontiers of Computer Science*, vol.19, no.6, 196322, DOI: 10.1007/S11704-024-40490-Y, 2024.
- [5] M. Çmen, M. Soysal, S. Belbağ et al., Dynamic open time-dependent traveling salesman problem with speed optimization, *International Transactions in Operational Research*, vol.32, no.6, pp.3316-3346, DOI: 10.1111/ITOR.13595, 2024.
- [6] P. Sinha, An observation on the repetitive nearest neighbour heuristic for the travelling salesman problem, *Abhigyan*, vol.42, no.2, pp.145-149, DOI: 10.1177/09702385241239481, 2024.
- [7] Z. Chen, H. Zhu, J. Wu, L. Fan et al., Health indicator construction for degradation assessment by embedded LSTM-CNN autoencoder and growing self-organized map, *Knowledge-Based Systems*, vol.252, DOI: 10.1016/J.KNOSYS.2022.109399, 2022.
- [8] S. Srinivasan and K. Rajakumar, Dimensionality reduction using optimized self-organized map technique for hyperspectral image classification, *Computer Systems Science and Engineering*, vol.47, no.2, pp.2481-2496, DOI: 10.32604/CSSE.2023.040817, 2023.
- [9] L. Jing, Parameter estimation of quantized DARMA systems using weighted least squares, *IET Control Theory & Applications*, vol.17, no.12, pp.1732-1738, DOI: 10.1049/CTH2.12507, 2023.
- [10] M. Hadavand and C. V. Deutsch, Spatial multivariate data imputation using deep learning and lambda distribution, *Computers and Geosciences*, vol.177, DOI: 10.1016/J.CAGEO.2023.105376, 2023.

- [11] N. A. Romli, N. F. S. Zulkepli, M. S. M. Kasihmuddin et al., An optimized logic mining method for data processing through higher-order satisfiability representation in discrete Hopfield neural network, *Applied Soft Computing*, vol.184, 113759, DOI: 10.1016/J.ASOC.2025.113759, 2025.
- [12] G. Avatharam and R. Sheela, Self-organizing map with granular competitive learning: Application to microarray clustering, *Intelligent Decision Technologies*, vol.16, no.3, pp.505-521, DOI: 10.3233/IDT-210264, 2022.
- [13] J. Zhu, H. Ye, L. Yao et al., Algorithm for solving traveling salesman problem based on self-organizing mapping network, *Journal of Shanghai Jiaotong University (Science)*, vol.29, no.3, pp.463-470, DOI: 10.1007/S12204-022-2517-3, 2022.
- [14] C. C. Dantas, Expanding and self-organizing 2D universe models emerging from frozen trivalent spin networks, *Classical and Quantum Gravity*, vol.38, no.20, DOI: 10.1088/1361-6382/AC25E1, 2021.
- [15] Q. Guan, H. Cao, X. Zhong, D. Yan and S. Xue, HiSOM: Hierarchical self-organizing map for solving multiple traveling salesman problems, *Networks*, vol.85, no.4, pp.368-381, 2025.
- [16] H. Tan, Y. Wu, J. Zhang et al., Study on a hybrid algorithm combining enhanced ant colony optimization and double improved simulated annealing via clustering in the Traveling Salesman Problem (TSP), *Peer J. Computer Science*, vol.9, e1609, DOI: 10.7717/PEERJ-CS.1609, 2023.
- [17] H. Liang, S. Wang, H. Li et al., BiGNN: Bipartite graph neural network with attention mechanism for solving multiple traveling salesman problems in urban logistics, *International Journal of Applied Earth Observation and Geoinformation*, vol.129, 103863, DOI: 10.1016/J.JAG.2024.103863, 2024.
- [18] A. A. Hameed, A. Jamil, M. E. Alazzawi et al., Improving the performance of self-organizing map using reweighted zero-attracting method, *Alexandria Engineering Journal*, vol.106, pp.743-752, DOI: 10.1016/J.AEJ.2024.08.081, 2024.
- [19] D. Ding, X. Liu, H. Zhang et al., Reversible image encryption and hiding algorithm based on fractional-order memristive Hopfield neural network, *Chaos, Solitons and Fractals: The Interdisciplinary Journal of Nonlinear Science, and Nonequilibrium and Complex Phenomena*, vol.199, 116757, DOI: 10.1016/J.CHAOS.2025.116757, 2025.
- [20] X. Li, X. Yang and X. Ju, A novel fractional-order memristive Hopfield neural network for traveling salesman problem and its FPGA implementation, *Neural Networks*, vol.179, 106548, DOI: 10.1016/J.NEUNET.2024.106548, 2024.
- [21] Z. Li, Dynamic network representation learning method combining graph neural networks and temporal information, *International Journal of Innovative Computing, Information and Control*, vol.20, no.6, pp.1803-1817, DOI: 10.24507/ijicic.20.06.1803, 2024.
- [22] Ranny, D. P. Lestari and T. L. E. R. Mengko, Separation sound event localization and detection using neural network and time frequency masking, *International Journal of Innovative Computing, Information and Control*, vol.20, no.3, pp.709-723, DOI: 10.24507/ijicic.20.03.709, 2024.
- [23] A. M. Shoukat, B. A. Sargano, A. Malyshev et al., SS3DNet-AF: A single-stage, single-view 3D reconstruction network with attention-based fusion, *Applied Sciences*, vol.14, no.23, 11424, DOI: 10.3390/APP142311424, 2024.
- [24] G. Fellek, A. Farid, S. Fujimura et al., G-DGNet: Gated deep graph attention network with reinforcement learning for solving traveling salesman problem, *Neurocomputing*, vol.579, 127392, DOI: 10.1016/J.NEUCOM.2024.127392, 2024.
- [25] Y. Doğan, SOMTreeNet: A hybrid topological neural model combining self-organizing maps and birch for structured learning, *Mathematics*, vol.13, no.18, 2958, DOI: 10.3390/MATH13182958, 2025.
- [26] M. Jared and A. Ali, Construction of trail networks based on growing self-organizing maps and public GPS data, *International Journal of Knowledge-Based and Intelligent Engineering Systems*, vol.28, no.2, pp.247-262, DOI: 10.3233/KES-230153, 2024.
- [27] A. Jamil, A. A. Hameed and Z. Orman, A faster dynamic convergency approach for self-organizing maps, *Complex & Intelligent Systems*, vol.9, no.1, pp.677-696, 2023.
- [28] A. Yatabe, Partitioning QUBO with two-way one-hot conditions on traveling salesman problems for city distributions with multiple clusters, *Frontiers in Computer Science*, vol.6, DOI: 10.3389/FCOMP.2024.1285244, 2024.
- [29] K. Tripathi, The novel hierarchical clustering approach using self-organizing map with optimum dimension selection, *Health Care Science*, vol.3, no.2, pp.88-100, DOI: 10.1002/HCS2.90, 2024.
- [30] Z. Liu, P. Zhao, J. Cao et al., A constrained multi-objective evolutionary algorithm with Pareto estimation via neural network, *Expert Systems with Applications*, vol.237, DOI: 10.1016/J.ESWA.2023.121718, 2024.
- [31] T. Zhang, *Task Planning for Multi-AUV Systems Based on SOM*, Ph.D. Thesis, Yanshan University, 2023.

- [32] F. Uddin, N. Riaz, A. Manan et al., An improvement to the 2-opt heuristic algorithm for approximation of optimal TSP tour, *Applied Sciences*, vol.13, no.12, 7339, DOI: 10.3390/APP13127339, 2023.
- [33] Y. Qiu, Y. Liu and Z. Yu, GCN-pointransformer model for traveling salesman problem, *Journal of Frontiers of Computer Science and Technology*, pp.1-14, 2024.
- [34] K. Li, Y. Xu, Z. Zhang and Y. Xi, Improved max-min ant system with visual correction for TSP, *Computer Applications and Software*, vol.40, no.10, pp.279-284+328, 2023.
- [35] J. Wu and Q. Duan, An algorithm for solving travelling salesman problem based on improved particle swarm optimisation and dynamic step Hopfield network, *International Journal of Vehicle Design*, vol.91, pp.208-231, DOI: 10.1504/IJVD.2023.131053, 2023.
- [36] A. Nayak and S. Rathinam, Heuristics and learning models for Dubins minmax traveling salesman problem, *Sensors (Basel, Switzerland)*, vol.23, no.14, DOI: 10.3390/S23146432, 2023.
- [37] Z. Liu, Application of improved ant colony algorithm in TSP, *Technology and Innovation*, vol.23, pp.166-168+171, 2023.
- [38] Y. Ahsini, D. P. Masa, B. Inglés et al., The electric vehicle traveling salesman problem on digital elevation models for traffic-aware urban logistics, *Algorithms*, vol.16, no.9, 402, DOI: 10.3390/A16090402, 2023.
- [39] H. Y. Jeong and B. D. Song, Meta-learning-based adaptive operator selection for traveling salesman problem, *Applied Soft Computing*, vol.185, 113930, DOI: 10.1016/J.ASOC.2025.113930, 2025.
- [40] A. S. Pon and V. Jeyalakshmi, A novel test pattern optimization method using recurrent neural network, *Journal of Circuits, Systems and Computers*, vol.33, no.4, DOI: 10.1142/S0218126624500762, 2023.

Author Biography



Wancheng Nie received his B.E. degree in Computer Science and Technology from Anhui Jianzhu University, China, in 2022 and is currently pursuing the M.E. degree in Electronic Information at Shanghai University of Engineering Science, China, with a curriculum focused on Machine Learning, Optimization Methods, and Data Structures and Algorithms. His research interests lie in neural network approaches to combinatorial optimization, particularly for solving the Traveling Salesman Problem and other path planning challenges, aiming to enhance the efficiency of intelligent decision-making systems in real-world scenarios.



Qianqian Duan received the Ph.D. degree in Control Science and Control Engineering from Shanghai Jiao Tong University, China, in 2016. She is currently a lecturer with the School of Electrical and Electronic Engineering, Shanghai University of Engineering Science, China. Her current research includes optimal operation.



Mohammad Karimi received the B.Sc. degree in Electrical Engineering from the Islamic Azad University, South Tehran Branch, Iran, in 2015, and the M.Sc. degree in Control Systems from the same university in 2017. He is currently pursuing the Ph.D. degree in Electrical Engineering (Control) at Tafresh University, Iran. His current research interests include robust control systems, resilience control, networked control systems, time-delay systems, fuzzy systems and neural networks, with applications to automotive systems and wind turbines.