Multimodal Fused Learning for Solving the Generalized Traveling Salesman Problem in Robotic Task Planning

Jiaqi Cheng[1] **Mingfeng Fan**[2*] **Xuefeng Zhang**[2] **Jingsong Liang**[2] **Yuhong Cao**[2]
Guohua Wu[1] **Guillaume Adrien Sartoretti**[2]
[1]Central South University [2]National University of Singapore

Abstract: Effective and efficient task planning is essential for mobile robots, especially in applications like warehouse retrieval and environmental monitoring. These tasks often involve selecting one location from each of several target clusters, forming a Generalized Traveling Salesman Problem (GTSP) that remains challenging to solve both accurately and efficiently. To address this, we propose a Multimodal Fused Learning (MMFL) framework that combines graph-based topology with image-based spatial representations to develop effective real-time task planning solutions. Specifically, we first introduce a novel coordinate-to-image builder that converts GTSP problem instances into spatially informative representations, complemented by an adaptive resolution scaling mechanism that ensures consistent performance across varying problem scales. We then incorporates a multimodal fusion mechanism featuring dedicated bottleneck that effectively merge topological and geometric information streams. Extensive experiments show that our MMFL approach significantly outperforms state-of-the-art methods across various GTSP instances while maintaining the computational efficiency required for real-time robotic applications. Physical robot tests further validate its practical effectiveness in real-world scenarios[†].

Keywords: Generalized Traveling Salesman Problem, Robotic Task Planning, Multimodal Learning

1 Introduction

Autonomous mobile robots have become indispensable in warehouse logistics [1, 2], healthcare, manufacturing [3], and disaster-response operations [4], where they are expected to complete complex, multi-stop missions while minimizing travel distance, energy expenditure, and makespan [5, 6, 7, 8]. A common requirement in these settings is that a robot must service exactly one location from each of several logically defined item or inspection sets. In a warehouse, for instance, a single stock-keeping unit (SKU) may reside on multiple shelves distributed throughout the facility. The robot may choose any one of those shelves as long as every requested SKU is collected [9, 10, 11]. This requirement is naturally formulated as a Generalized Traveling Salesman Problem (GTSP), in which nodes are partitioned into clusters that represent interchangeable task alternatives and the objective is to find the shortest tour visiting one node from every cluster [12], as shown in Figure 1.

While exact branch-and-bound or cutting-plane algorithms can guarantee optimal solutions to GTSP, their computational cost scales exponentially with the problem size, making them impractical for on-board execution in real-time robotic applications [13, 14, 15]. Metaheuristic techniques such as genetic algorithms [16], ant-colony optimization [17], and memetic algorithm (MA) [18] offer better scalability but require extensive parameter tuning and domain-specific adaptation, which

*Corresponding author: ming.fan@nus.edu.sg
†The source code is available at https://github.com/Carveller/MMFL-for-GTSP

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

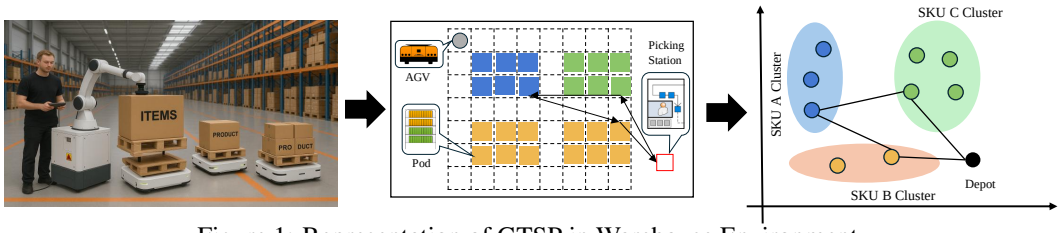


Figure 1: Representation of GTSP in Warehouse Environment.

severely limits their portability across different robots and environments. More recently, neural approaches have shown promise by learning heuristics directly from data [19]. These neural-based approaches can be categorized into graph-learning methods and hybrid learning–search methods. In neural-based approaches, most of these models only take in graph-based inputs. For example, POMO [20], which employs a Transformer-style graph encoder as our model, outperforms the edge-aware GCN [21]. In addition, several other methods [22, 23, 24, 25, 26] also adopt graph-based learning approaches; however, they overlook the global image-level information inherent to path-planning problems, leaving substantial room for improvement. Regarding hybrid learning–search methods, [21] employs supervised learning to solve path-planning problems, but it relies on high-quality datasets that are extremely difficult to obtain for path-planning tasks, which limits its scalability. [27] integrates neural networks with exact algorithms; although it can achieve high-quality solutions, it still requires substantial computational time when tackling large-scale instances. In addition, approaches such as [28, 29, 30] combine heuristic or meta-heuristic algorithms with reinforcement learning, thereby improving solution quality, yet they still incur noticeable solving times.

To overcome these limitations, we introduce a *Multimodal Fused Learning* (MMFL) framework for solving GTSP in robotic task planning applications. MMFL represents GTSP instances using both a graph structure and a constructed image that encodes the spatial layout of nodes and clusters. By combining these complementary representations, MMFL can better understand the geometric relationships between potential visitation points, leading to improved solution quality. Our MMFL framework offers a novel perspective on solving GTSP. Unlike conventional approaches that rely solely on graph-based or heuristic methods, MMFL leverages the complementary strengths of both topological and spatial representations. By constructing a rich dual representation that captures both connectivity relationships and geometric distributions of nodes, our model develops a more comprehensive understanding of the problem structure. The framework's architecture dynamically adapts to varying instance scales and spatial configurations, while its fusion mechanism intelligently integrates information streams from different modalities, enabling more context-aware decision making throughout the task planning process.

Through extensive experiments in simulated robotics environments, we demonstrate that our approach outperforms existing methods for GTSP, achieving shorter paths while maintaining computational efficiency suitable for real-time robot operation. Physical robot experiments conducted on mobile vehicle platforms validate that these improvements translate directly to real-world applications. The system's ability to efficiently process environmental information and produce high-quality paths under strict computational constraints makes MMFL a valuable tool for practical robotic applications in warehouses, healthcare facilities, and manufacturing environments.

2 Preliminary

Problem formulation. GTSP can be formally defined on a complete graph $G = (V, E)$, where $V = \{v_1, \ldots, v_n\}$ represents the set of nodes, and $E = \{e(v_i, v_j) \mid v_i, v_j \in V, i \neq j\}$ represents the set of edges. The node set V is partitioned into m mutually exclusive and collectively exhaustive clusters $V_1,\ V_2,\ \ldots, V_m$, such that $V = \bigcup_{i=1}^{m} V_i \ and \ V_i \bigcap V_j = \emptyset, \forall i \neq j$. Each edge $e\left(v_i, v_j\right) \in E$ is associated with a non-negative cost or distance c_{ij}. The objective of GTSP is to find a minimum-cost tour that visits exactly one node from each cluster. Formally, we seek a permutation $\pi = \pi_1, \pi_2, \ldots, \pi_m$ of m nodes, where $\pi_i \in V_i$, such that the total tour cost is minimized: $\min_{\pi} f\left(\pi\right) = \sum_{i=1}^{m-1} c_{\pi_i, \pi_{i+1}} + c_{\pi_m, \pi_1}$. The complete mathematical formulation of the GTSP is provided in Appendix A.

Markov decision process(MDP). Given an GTSP instance λ, we construct a solution as an MDP. An *agent* iteratively takes the current *state* as input (e.g., the instance information and the partially constructed solution, initially empty), and outputs a probability distribution over candidate nodes belonging to unvisited clusters. The *action* is a node that is greedily selected or randomly sampled from the policy. The *transition* dynamics is joining the node to the partial solution and masking all nodes from its associated cluster to prevent revisits. We parameterize the *policy* p of the agent by a neural network p_θ, so that the probability of constructing a complete tour π to the GTSP is expressed by $p_\theta(\pi|\lambda) = \prod_{t=1}^{T} p_\theta(\pi_t|\pi_{<t}, \lambda)$, where π_t and $\pi_{<t}$ represent the selected node and partial solution at the t-th step. Typically, the *reward* is defined as the negative value of the total tour cost, e.g., $\mathcal{R}(\pi) = -f(\pi)$.

3 Multimodal Fused Learning (MMFL) for GTSP

3.1 Overview

Our framework incorporates complementary information from both graph and image modalities. The two key challenges are (1) constructing informative images from GTSP instances and (2) effectively fusing graph and image data. To tackle these challenges, we first propose an image construction method with an Adaptive Resolution Scaling (ARS) strategy, and then design specialized graph and image encoders alongside a multimodal fusion module to integrate information from both representations. As shown in Figure 2, the overall architecture of MMFL consists of a coordinate-based image builder that transforms GTSP instances into informative image representations, a graph encoder, an image encoder, a multimodal fusion module, and a multi-start decoder. We elaborate on the key components below.

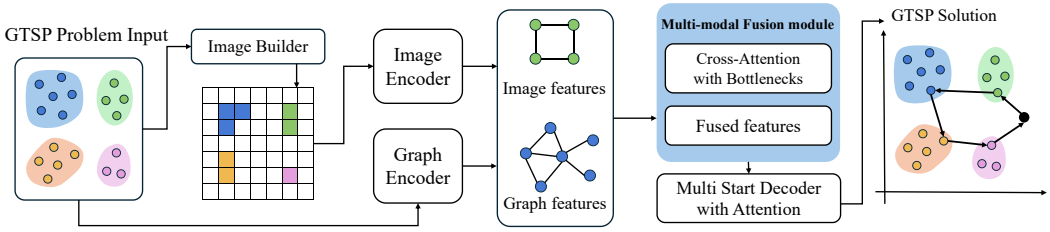


Figure 2: The overall architecture of MMFL.

3.2 Policy Network

Image builder. We construct an informative image representation by encoding both node positions and cluster memberships into a single-channel image. Each GTSP instance consists of n nodes distributed across m clusters, where each node has $2D$ coordinates and a cluster assignment. Given a node i with coordinates $(v_{i,1}, v_{i,2}) \in [0, 1]^2$ and cluster assignment $c_i \in 1, \ldots, m$, we map it to a pixel in the image representation as follows. First, we determine the image dimensions $W \times H$ based on the problem size n. We then convert the normalized node coordinates to discrete pixel positions: $v'_{i,1} = \lfloor v_{i,1} \times W \rfloor$ and $v'_{i,2} = \lfloor v_{i,2} \times H \rfloor$. The instance image I is constructed by treating each coordinate as a pixel location and the cluster index as the pixel value:

$$I(x, y) = \begin{cases} c_i + 1, & \text{if } (x, y) = (v'_{i,1}, v'_{i,2}) \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

When constructing image representations for GTSP instances, a significant challenge arises from varying problem sizes. Using a fixed image resolution would create an inconsistent information density as the number of nodes increases, potentially obscuring important spatial relationships that are crucial to effectively solving the problem. To address this challenge, we design an ARS strategy, which dynamically adjusts the image resolution based on the problem size. Our ARS strategy uses the formula $W = H = \left\lceil \frac{\alpha\sqrt{n}}{w} \right\rceil \times w$, where n is the number of nodes, w is the patch size used in the image encoder, and α is a scaling factor. This formulation ensures that the image dimensions scale

proportionally with the square root of the problem size, maintaining an approximately constant node density across different problem scales.

With our ARS strategy, larger problem instances are represented by proportionally larger images, preserving spatial relationships between nodes. For these variable-sized representations, we implement a flexible positional encoding using an MLP that maps coordinates to embeddings:

$$PosEncoder\,(i,j) = \text{MLP}\,([i/W, j/H])\,, \tag{2}$$

where (i, j) are the original patch coordinates in the image grid, and $(i/W, j/H)$ normalizes these coordinates to $[0, 1]^2$.

Image encoder. To process the constructed GTSP instance images I, we implement a specialized Vision Transformer (ViT) architecture that effectively extracts spatial features while adapting to varying image resolutions. Our image encoder consists of three primary components: patch embedding, positional encoding, and a transformer encoder stack.

The patch embedding layer transforms the instance image into a sequence of patch embeddings. Given an input image $I \in \mathbb{R}^{1\times H\times W}$, we divide it into non-overlapping patches of size $w \times w$, and project each patch into a d-dimensional embedding space using a convolutional operation.

After embedding the patches, we add the position encodings described in the previous section to inject spatial information. The combined embeddings then serve as input to a stack of transformer encoder layers. Each layer consists of a multi-head self-attention (MHSA) sublayer followed by a feed-forward network (FFN), with layer normalization and residual connections:

$$\widehat{z^{(l)}} = \text{LN}\left(z^{(l-1)} + \text{MHSA}\left(z^{(l-1)}\right)\right) \tag{3}$$

$$z^{(l)} = \text{LN}\left(\widehat{z^{(l)}} + \text{FFN}\left(\widehat{z^{(l)}}\right)\right), \tag{4}$$

where $z^{(l)}$ represents the patch embeddings at layer l, and LN denotes layer normalization. The final output of the image encoder is a sequence of enhanced patch embeddings $z^{(L)} \in \mathbb{R}^{N\times d}$, where N is the number of patches and L is the number of transformer layers.

Graph encoder. The graph encoder processes the node coordinates and cluster information to generate node embeddings. For each node i in the GTSP instance, we first create an initial embedding by concatenating its coordinates information and cluster index: $u_i = [v_{i,1}, v_{i,2}, c_i]$ where $v_{i,1}$ and $v_{i,2}$ represent the node's coordinates, and c_i is the cluster index to which node i belongs. These features are transformed into initial embeddings through linear projection:

$$h_i^{(0)} = W_u u_i + b_u, \forall i \in 1, \ldots, n, \tag{5}$$

where $h_i^{(0)}$ is the initial embedding for node i; W_u is a learnable weight matrix; b_u is a bias vector and n is the total number of nodes in the instance. The graph encoder consists of multiple self-attention layers. Each layer applies MHSA to capture relationships between nodes, followed by an FFN with residual connections and normalization:

$$h_i^{(l)} = \text{LayerNorm}\left(h_i^{(l-1)} + \text{MHSA}\left(h_i^{(l-1)}\right)\right) \tag{6}$$

$$h_i^{(l)} = \text{LayerNorm}\left(h_i^{(l)} + \text{FFN}\left(h_i^{(l)}\right)\right), \tag{7}$$

where $h_i^{(l)}$ is the embedding of node i at layer l. This structure enables the model to capture complex relationships between nodes and their cluster assignments.

Multimodal fusion module. To effectively integrate information from both graph and image modalities, we implement a bidirectional cross-modal fusion mechanism using learnable bottleneck tokens. This approach enables flexible information exchange while maintaining computational efficiency.

Our fusion mechanism consists of multiple layers that iteratively refine the representations from both modalities. Each fusion layer is structured as follows:

$$\mathbf{G}^{in} = [\mathbf{h}_{graph}; \mathbf{b}_{graph}] \tag{8}$$

$$\mathbf{I}^{in} = [\mathbf{h}_{image}; \mathbf{b}_{image}], \tag{9}$$

where $\mathbf{h}_{graph} \in \mathbb{R}^{b\times n\times d}$ represents the node embeddings from the graph encoder; $\mathbf{h}_{image} \in \mathbb{R}^{b\times m\times d}$ represents the patch embeddings from the image encoder; $\mathbf{b}_{graph} \in \mathbb{R}^{b\times n_b\times d}$ and $\mathbf{b}_{image} \in \mathbb{R}^{b\times n_b\times d}$ are learnable bottleneck parameters specific to each modality. These bottleneck tokens act as information conduits between modalities, with n_b being a hyperparameter controlling fusion capacity. We then apply the cross-attention to process $\mathbf{G}^{in}$ and $\mathbf{I}^{in}$:

$$\mathbf{G}^{out} = \text{MHA}(\mathbf{G}^{in}, \mathbf{I}^{in}, \mathbf{I}^{in}) \tag{10}$$

$$\mathbf{I}^{out} = \text{MHA}(\mathbf{I}^{in}, \mathbf{G}^{in}, \mathbf{G}^{in}), \tag{11}$$

where MHA denotes the multi-head attention operation. Following the attention operation, we apply layer normalization and feed-forward networks with residual connections:

$$\mathbf{h}^{norm}_{graph} = \text{LayerNorm}(\mathbf{h}_{graph} + \mathbf{G}^{out}_{:n}) \tag{12}$$

$$\mathbf{b}^{norm}_{graph} = \text{LayerNorm}(\mathbf{b}_{graph} + \mathbf{G}^{out}_{n:}) \tag{13}$$

$$\mathbf{h}^{out}_{graph} = \mathbf{h}^{norm}_{graph} + \text{FFN}(\mathbf{h}^{norm}_{graph}) \tag{14}$$

$$\mathbf{b}^{out}_{graph} = \mathbf{b}^{norm}_{graph} + \text{FFN}(\mathbf{b}^{norm}_{graph}). \tag{15}$$

Similar operations are also applied to the image features. After multiple fusion layers, the final fused representation for GTSP is computed as:

$$\mathbf{h}_{fused} = \mathbf{h}^{out}_{graph} + \alpha \cdot \text{Mean}(\mathbf{h}^{out}_{image}), \tag{16}$$

where α is a weighting factor set to 0.5 in our implementation. This fusion mechanism enables bidirectional information flow between modalities, allowing the model to leverage both the topological information from the graph and the spatial information from the image, resulting in more effective node selection decisions for GTSP.

Multi-start decoder. Our multi-start decoder constructs a tour by selecting one node from each cluster in an auto-regressive manner. For each decoding step t, we compute the agent's policy based on the embedding of the previously selected node and the fused node representations. At the initial step ($t = 0$), the model always selects the depot node (node 0), as the agent must begin there. At $t = 1$, it selects the k-nearest neighbors to the depot, enabling diverse exploration via parallel rollouts [20]. For subsequent steps ($t \geq 2$), we first retrieve the embedding of the previously selected node:

$$h_{\pi_{t-1}} = [h_{\text{fused}}, \pi_{t-1}]. \tag{17}$$

We then compute query vectors by combining node information with global context:

$$q_{\text{graph}} = W_{q_{\text{graph}}}(g), \quad q_{\text{last}} = W_{q_{\text{last}}}(h_{\pi_{t-1}}), \quad q = q_{\text{last}} + q_{\text{graph}}, \tag{18}$$

where $g = h_{\text{graph}} + 0.3 \cdot \text{Mean}(h_{\text{image}})$. Next, we compute compatibility scores for each node as follows:

$$\alpha_i = \begin{cases} -\infty & \text{if node } i \text{ is masked} \\ C \cdot \tanh\left(\frac{q_c^T W_K h_i}{\sqrt{d}}\right) & \text{otherwise,} \end{cases} \tag{19}$$

where α_i is the compatibility score for node i; h_i is the fused embedding of node i; W_K is a learnable projection matrix; d is the embedding dimension and $C = 10$ is a clipping parameter. A node is masked if it has already been visited or if its cluster has already been visited, enforcing the GTSP constraint. The policy activation for each node is finally computed using a softmax function:

$$p\left(\pi_t = i | \pi_{1:t-1}\right) = \frac{\exp\left(\alpha_i\right)}{\sum_j \exp\left(\alpha_j\right)}, \tag{20}$$

where $p\left(\pi_t = i | \pi_{1:t-1}\right)$ represents the probability of selecting node i at step t given the previously selected nodes $\pi_{1:t-1}$. During training, nodes are sampled from this policy, while during inference, we can either sample or select greedily based on the configured evaluation mode.

3.3 Training Procedure

We train our model using the REINFORCE algorithm [31], incorporating a shared baseline [20] to reduce variance, which is the average reward across all rollouts for each instance. Specifically, given a batch of N instance $\{\lambda_i\}_{i=1}^{N}$, we sample k tours (i.e., solutions) $\{\pi^{i,j}\}_{j=1}^{k}$ for each instance λ_i. The model parameters θ are then updated as: $\nabla_\theta \mathcal{L}(\theta|\lambda_i) \simeq \frac{1}{kN}\sum_{i=1}^{N}\sum_{j=1}^{k}(\mathcal{R}(\pi^{i,j}|\lambda_i) - b(\lambda_i))\nabla_\theta \log p_\theta(\pi^{i,j}|\lambda_i)$, where $b(\lambda_i)$ denotes the shared baseline for instance λ_i, computed as: $b(\lambda_i) = \frac{1}{k}\sum_{j=1}^{k}\mathcal{R}(\pi^{i,j})$. Detailed information on the training procedure, including the algorithm and dataset, is provided in Appendix B.

4 Experiments

4.1 Experimental Settings

Problems. We conduct extensive experiments to evaluate the performance of our proposed MMFL framework on GTSP. For comprehensive evaluation, we generate five scale-based GTSP instance sets: $(n = 20, m = 4), (n = 50, m = 10), (n = 100, m = 20), (n = 150, m = 30), (n = 200, m = 40)$, where n represents the number of nodes and m the number of groups. In these instances, node coordinates were uniformly sampled from the unit square $[0, 1]^2$. To assess algorithmic generalization, we create four distinct grouping configurations using $n = 100$ nodes: Random Groups (random assignment), Proximity-Based Groups (nearest centroid assignment), Density-Based Groups (DBSCAN-style assignment), and Hybrid Groups (mixed assignment strategies). Additionally, we examine four group size distributions with $n = 100$ total nodes: Uniform Groups (20 groups, 5 nodes each), Small Groups (40 groups, 2-3 nodes each), Large Groups (10-12 groups, 8-10 nodes each), and Mixed Groups (15-20 groups, 1-15 nodes each). For each setting, we generate 30 GTSP instances to form the test dataset.

Hyperparameters. For model training and evaluation, we design our neural network with an embedding dimension of 128, 3 image and graph encoder layers, and 8 attention heads. The image processing branch uses 16×16 patch size with ARS strategy. Our multimodal fusion module consists of 3 fusion layers with 10 bottleneck tokens per modality. We define the number of rollouts sampled for each instance as one quarter of the problem size (i.e., $k = \text{int}(n/4)$). The model parameters are optimized using Adam with a learning rate of 1×10^{-4} and a weight decay of 1×10^{-6}. A cosine annealing scheduler is employed to gradually decrease the learning rate during training. The model is trained for 200 epochs with gradient clipping set to 1.0. Each epoch processes 100,000 instances with a batch size of 128.

Baselines. We compare our approach against several state-of-the-art GTSP algorithms: Lin-Kernighan-Helsgaun (LKH), a heuristic typically producing near-optimal solutions for TSP variants [32]; Google OR-Tools, an efficient optimization toolkit with specialized routing capabilities [33]; MA, combining genetic algorithms with local search techniques [18]; Adaptive Large Neighborhood Search (ALNS), a destroy-and-repair metaheuristic [34]; and Policy Optimization with Multiple Optima (POMO), a reinforcement learning method exploiting problem symmetry [20] for vehicle routing problems. All methods are executed on a machine equipped with an RTX 4080 Laptop GPU and an Intel i9-13980HX CPU.

Metrics. We evaluate algorithm performance using three metrics: the average objective value (Obj.) representing the total tour length; the average optimality gap (Gap) calculated as (Obj - Best)/Best × 100%; and total run time (Time) per test dataset measured in CPU seconds. For learning-based methods, we only report inference time. The best results are highlighted in bold.

4.2 Experiment Results

Comparison analysis. Our comparison results are shown in Table 1, the results demonstrate the superior performance of our MMFL framework across all problem scales. For small instances, MMFL matches the performance of MA and POMO, but as problem size increases, MMFL's advantage be-

Table 1: Comparison of algorithms on GTSP with various problem sizes.

Algorithm	n=20, c=4			n=50, c=10			n=100, c=20			n=150, c=30			n=200, c=40		
	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time
LKH	1.18	4.42	0.60	1.98	15.79	2.10	2.73	21.33	6.90	3.33	4.06	30.30	3.83	5.51	42.00
OR-Tools	1.24	9.73	9.35	2.13	24.56	13.08	2.84	26.22	18.21	3.69	15.31	27.54	3.78	4.13	45.68
MA	**1.13**	**0.00**	1.80	1.92	12.28	9.30	2.71	20.44	106.80	3.40	6.25	655.20	3.85	6.06	1791.30
ALNS	1.18	4.42	1.20	1.94	13.45	2.10	2.64	17.33	6.30	3.23	0.94	69.60	3.69	1.65	33.00
POMO	**1.13**	**0.00**	0.11	**1.71**	**0.00**	0.12	2.72	20.89	0.12	3.32	3.75	0.13	3.77	3.86	0.13
MMFL	**1.13**	**0.00**	0.12	**1.71**	**0.00**	0.11	**2.25**	**0.00**	0.13	**3.20**	**0.00**	0.13	**3.63**	**0.00**	0.14

comes substantial, improving over LKH by 4.06 - 21.33% and OR-Tools by 4.13 - 26.22%. MMFL also maintains efficient inference times, comparable to POMO but significantly faster than MA and LKH whose computational costs increase dramatically with problem size.

Table 2: Performance on different GTSP instance types (n=100).

Algorithm	Random			Proximity			Density			Hybrid		
	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time
LKH	3.04	28.27	6.90	3.21	1.90	6.70	**1.45**	**0.00**	9.00	2.53	17.13	9.40
OR-Tools	3.24	36.71	40.23	3.92	24.44	32.98	1.80	22.45	20.29	2.77	28.24	30.27
MA	3.29	38.82	84.30	3.24	2.86	62.70	1.46	0.69	61.50	2.58	19.44	71.10
ALNS	2.97	25.32	6.90	3.22	2.22	5.70	**1.45**	**0.00**	4.50	2.53	17.13	5.70
POMO	2.46	3.80	0.12	3.20	1.59	0.14	1.56	7.59	0.13	2.42	12.04	0.14
MMFL	**2.37**	**0.00**	0.13	**3.15**	**0.00**	0.15	1.47	1.38	0.09	**2.16**	**0.00**	0.12

Table 3: Performance on GTSP with varying group sizes (n=100).

Algorithm	Uniform			Small			Large			Mixed		
	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time
LKH	2.24	6.67	6.60	4.23	7.91	21.90	1.74	45.00	2.70	2.29	13.93	4.50
OR-Tools	2.62	24.76	38.56	4.78	21.94	62.35	1.81	50.83	20.56	2.84	41.29	40.37
MA	2.47	17.62	78.30	4.38	11.73	1026.00	1.65	37.50	15.90	2.29	13.93	44.70
ALNS	2.24	6.67	6.90	4.21	7.40	30.62	1.74	45.00	3.10	2.22	10.45	4.51
POMO	2.21	5.24	0.12	4.12	5.10	0.14	1.63	35.83	0.13	2.36	17.41	0.14
MMFL	**2.10**	**0.00**	0.12	**3.92**	**0.00**	0.16	**1.20**	**0.00**	0.12	**2.01**	**0.00**	0.13

Generalization analysis. Tables 2 and 3 demonstrate MMFL's generalization across different problem structures. MMFL achieves best performance in 3 of 4 group distributions, with only a slight 1.38% gap in Density-Based Groups where LKH and ALNS perform best. For varying group sizes, MMFL also consistently delivers superior solutions, with substantial improvements in Large Groups which is 35.83% better than POMO and 45.00% better than LKH. In Small Groups, MMFL is also 5.10% better than POMO, 7.40% better than ALNS. Figure 3a and Figure 3b show route maps for some examples, these results confirm that MMFL effectively generalizes across diverse problem structures with different spatial distributions and group configurations, while maintaining computational efficiency.

Ablation study. To validate each component's contribution, we conducted ablation studies on the image encoder and multimodal fusion module. Figure 4 shows our complete MMFL model consistently maintains lower length values than ablated versions. Without our image encoder, convergence is slower in early epochs, while removing multimodal fusion causes higher initial values and instability. Even after convergence, MMFL still retains a consistent advantage of approximately 0.1 in tour length, confirming both components are essential for optimal performance. Detailed results are provided in Appendix C.

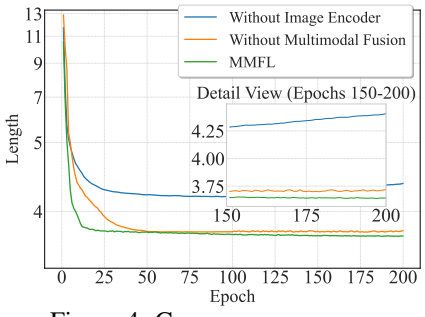
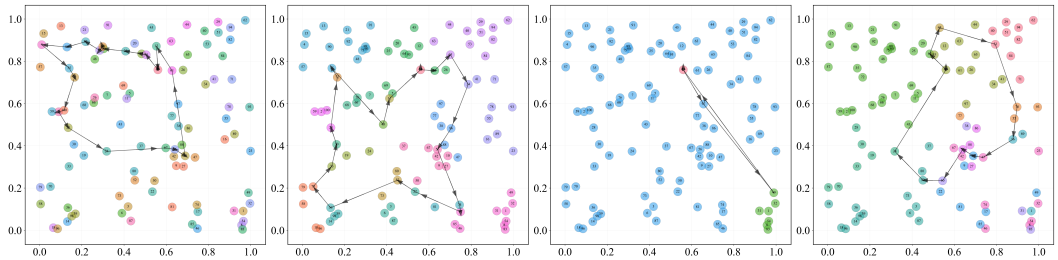
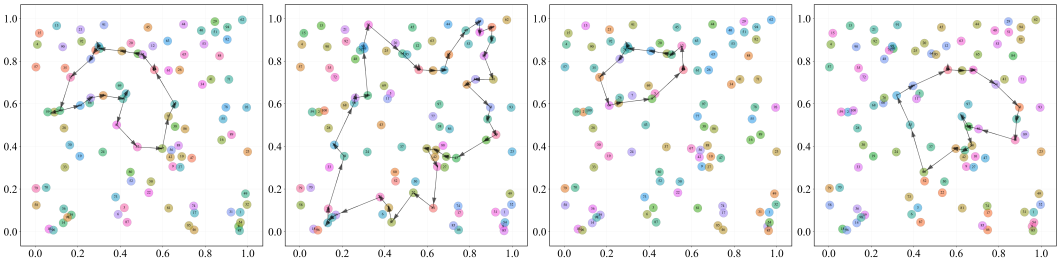


Figure 4: Convergence curves.

Experimental Validation. We implemented our proposed MMFL framework on a physical robot platform to validate its effectiveness in real-world task planning scenarios. We deployed our algorithm on a mobile robot operating in a complex indoor environment where the robot needed to



a: GTSP instances with different grouping distributions.



b: GTSP instances with varying group sizes.

Figure 3: GTSP instance visualizations under different configurations.

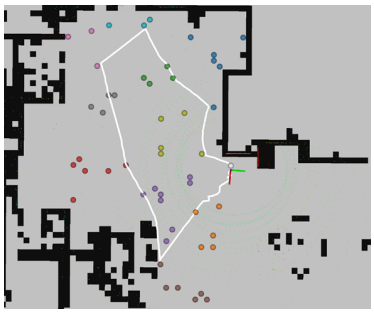


Figure 5: MMFL solution path for a multi-zone exploration task.

Figure 6: Locomotion in a real-world environment.

explore different regions, each containing multiple possible observation points. Figures 5 and 6 show our experimental results, where the robot successfully navigated through multiple zones (indicated by different colored nodes). The white path represents the solution generated by our algorithm, demonstrating MMFL's ability to effectively solve GTSP instances in practical settings.

5 Conclusion

In this paper, we introduce MMFL, a novel framework for solving GTSP in robotic task planning scenarios. Our method uniquely integrates both graph-based and spatial representation learning through a carefully designed fusion architecture incorporating ARS strategy and dedicated bottleneck mechanisms. Extensive experimental results across multiple problem sizes and configurations demonstrate that MMFL consistently outperforms state-of-the-art algorithms, achieving optimal or near-optimal solutions with significantly shorter tour lengths while maintaining competitive computational efficiency. Our performance advantages are particularly pronounced for complex instances with diverse spatial distributions and group configurations, highlighting MMFL's strong generalization capabilities. Physical robot experiments further validate the practical effectiveness of our approach in real-world navigation scenarios. While some challenges remain for extreme problem distributions, we believe that MMFL represents a significant advancement in solving GTSP efficiently for robotic applications, with potential extensions to other combinatorial optimization problems involving spatial reasoning.

6 Limitation

While our MMFL framework demonstrates strong performance in solving GTSP instances for robotic task planning, several limitations should be acknowledged:

(1) **Limited generalization to novel distributions.** The current model is trained on a fixed distribution of problem instances and may exhibit degraded performance when faced with significantly different distributions, such as asymmetric node layouts, highly imbalanced cluster densities, or non-uniform spatial patterns. In the future, we will boost the robustness of MMFL via robust learning technologies (e.g., meta-learning, curriculum learning, and few-shot fine-tuning).

(2) **Computational scalability challenges.** Although MMFL incorporates our ARS strategy to improve efficiency, solving very large-scale GTSP instances remains computationally expensive. Our multimodal fusion module, while effective, introduces additional overhead compared to graph-only baselines, particularly when scaling to instances with thousands of nodes and clusters. Future work will explore integrating a divide-and-conquer architecture to maintain both high solution quality and computational efficiency in large-scale task planning scenarios.

(3) **Environmental complexity limitations.** Our current framework assumes a static environment with known node positions. However, in real-world scenarios, robots often operate in dynamic and uncertain environments with moving obstacles, temporal constraints, and localization uncertainty caused by challenging terrain or infrastructure limitations. Future work will focus on incorporating such dynamic elements into the planning process.

References

[1] Z. Qiu, J. Long, Y. Yu, and S. Chen. Integrated task assignment and path planning for multi-type robots in an intelligent warehouse system. *Transportation Research Part E: Logistics and Transportation Review*, 194:103883, 2025.

[2] S. Zhang, Q. Han, H. Zhu, H. Wang, H. Li, and K. Wang. Real time task planning for order picking in intelligent logistics warehousing. *Scientific Reports*, 15(1):7331, 2025.

[3] S. Sandrini, M. Faroni, and N. Pedrocchi. Learning and planning for optimal synergistic human–robot coordination in manufacturing contexts. *Robotics and Computer-Integrated Manufacturing*, 95:103006, 2025.

[4] Y. Huang, Y. Wang, Z. Li, H. Zhang, and C. Zhang. A hierarchical multi robot coverage strategy for large maps with reinforcement learning and dense segmented siamese network. *IEEE Robotics and Automation Letters*, 2024.

[5] H. Bai, P. Yang, Z. Qin, M. Qi, and W. Xiong. Order sequencing, tote scheduling, and robot routing optimization in multi-tote storage and retrieval autonomous mobile robot systems. *International Journal of Production Research*, 63(1):314–341, 2025.

[6] D. Zhang, Y. Yin, G. Xiong, and S. Zou. A bi-level hybrid algorithm for solving multi-target inspection path planning problem of mobile robot in complex radioactive indoor environment. *Expert Systems with Applications*, 266:126095, 2025.

[7] K. Kim, J. Park, and J. Kim. Optimized area partitioning for cooperative underwater search using multiple autonomous underwater vehicles. *International Journal of Control, Automation, and Systems*, 23(2):392–404, 2025.

[8] Y. Yu, Q. Tang, Q. Jiang, and Q. Fan. A deep reinforcement learning-assisted multimodal multi-objective bi-level optimization method for multi-robot task allocation. *IEEE Transactions on Evolutionary Computation*, 2025.

[9] N. Adachi and H. Date. Path planning considering energy consumption of crawler robots in mountain environments. In *2024 IEEE/SICE International Symposium on System Integration (SII)*, pages 79–84. IEEE, 2024.

[10] L. Zahorán and A. Kovács. Proseqqo: A generic solver for process planning and sequencing in industrial robotics. *Robotics and Computer-Integrated Manufacturing*, 78:102387, 2022.

[11] S. Bock, S. Bomsdorf, N. Boysen, and M. Schneider. A survey on the traveling salesman problem and its variants in a warehousing context. *European Journal of Operational Research*, 322(1):1–14, 2025.

[12] P. C. Pop, O. Cosma, C. Sabo, and C. P. Sitar. A comprehensive survey on the generalized traveling salesman problem. *European Journal of Operational Research*, 314(3):819–835, 2024.

[13] J. Deckerová, P. Váňa, and J. Faigl. Combinatorial lower bounds for the generalized traveling salesman problem with neighborhoods. *Expert Systems with Applications*, 258:125185, 2024.

[14] M. Fischetti, J. J. Salazar González, and P. Toth. A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research*, 45(3):378–394, 1997.

[15] Y. Yuan, D. Cattaruzza, M. Ogier, and F. Semet. A branch-and-cut algorithm for the generalized traveling salesman problem with time windows. *European Journal of Operational Research*, 286(3):849–866, 2020.

[16] Z. Ahmed, M. Choudhary, and I. Al-Dayel. Effects of crossover operator combined with mutation operator in genetic algorithms for the generalized travelling salesman problem. *International Journal of Industrial Engineering Computations*, 15(3):18, 2024.

[17] E. R. R. Kato, R. S. Inoue, and L. dos Santos Franco. A method for planning multirotor unmanned aerial vehicle flight paths to cover areas using the ant colony optimization metaheuristic. *Computers and Electronics in Agriculture*, 231:109983, 2025.

[18] O. Cosma, P. C. Pop, and L. Cosma. A novel memetic algorithm for solving the generalized traveling salesman problem. *Logic Journal of the IGPL*, 32(4):576–588, 2024.

[19] K. Chung, C. Lee, and Y. Tsang. Neural combinatorial optimization with reinforcement learning in industrial engineering: a survey. *Artificial Intelligence Review*, 58(5):130, 2025.

[20] Y.-D. Kwon, J. Choo, B. Kim, I. Yoon, Y. Gwon, and S. Min. Pomo: Policy optimization with multiple optima for reinforcement learning. *Advances in Neural Information Processing Systems*, 33:21188–21198, 2020.

[21] C. K. Joshi *et al.* An efficient graph convolutional network technique for the travelling salesman problem. *arXiv preprint arXiv:1906.01227*, 2019.

[22] Z. Fang, J. Chen, Z. Zhang, and D. Su. Learning node-pair insertion for the pickup and delivery problem with time windows. In *2024 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1678–1683. IEEE, 2024.

[23] D. Kong, Y. Ma, Z. Cao, T. Yu, and J. Xiao. Efficient neural collaborative search for pickup and delivery problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[24] W. Kool, H. Van Hoof, and M. Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2018.

[25] J. Luo and C. Li. An efficient encoder-decoder network for the capacitated vehicle routing problem. *Expert Systems with Applications*, 278:127311, 2025.

[26] R. N. Monemi, S. Gelareh, P. H. González, L. Cui, K. Bouamrane, Y.-H. Dai, and N. Maculan. Graph convolutional networks for logistics optimization: A survey of scheduling and operational applications. *Transportation Research Part E: Logistics and Transportation Review*, 197:104083, 2025.

[27] V. Nair, S. Bartunov, Gimeno, et al. Solving mixed integer programs using neural networks. *arXiv preprint arXiv:2012.13349*, 2020.

[28] H. Ye, J. Wang, Z. Cao, H. Liang, and Y. Li. Deepaco: Neural-enhanced ant systems for combinatorial optimization. *Advances in neural information processing systems*, 36:43706–43728, 2023.

[29] Y. Ma, J. Li, Z. Cao, W. Song, L. Zhang, Z. Chen, and J. Tang. Learning to iteratively solve routing problems with dual-aspect collaborative transformer. *Advances in Neural Information Processing Systems*, 34:11096–11107, 2021.

[30] Y. Wu, W. Song, Z. Cao, J. Zhang, and A. Lim. Learning improvement heuristics for solving routing problems. *IEEE transactions on neural networks and learning systems*, 33(9):5057–5069, 2021.

[31] J. Zhang, J. Kim, B. O'Donoghue, and S. Boyd. Sample efficient reinforcement learning with reinforce. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 10887–10895, 2021.

[32] K. Helsgaun. Solving the equality generalized traveling salesman problem using the lin–kernighan–helsgaun algorithm. *Mathematical Programming Computation*, 7:269–287, 2015.

[33] T. Cuvelier, F. Didier, V. Furnon, S. Gay, S. Mohajeri, and L. Perron. Or-tools' vehicle routing solver: a generic constraint-programming solver with heuristic search for routing problems. In *24e congrès annuel de la société française de recherche opérationnelle et d'aide à la décision*, 2023.

[34] S. L. Smith and F. Imeson. Glns: An effective large neighborhood search heuristic for the generalized traveling salesman problem. *Computers & Operations Research*, 87:1–19, 2017.

Appendix

A Mathematical model of GTSP

This appendix provides the detailed integer linear programming(ILP) formulation of the Generalized Traveling Salesman Problem (GTSP) referened in the main body of this paper.

Definitions and notation. The GTSP is defined on a graph where nodes are partitioned into a number of predefined, mutually exclusive clusters. The objective is to find a minimum-cost tour that visits exactly one node from each cluster.

Let $V = v_1, v_2, \ldots, v_n$ be the set of n nodes. The set of nodes V is partitioned into m mutually exclusive and collectively exhaustive clusters $V_1, V_2, \ldots, V_m$, such that $V = \bigcup_{i=1}^{m} V_i \; and \; V_i \bigcap V_j = \emptyset, \forall i \neq j$. Let c_{ij} be the non-negative cost (e.g., distance or travel time) associated with traversing the arc from node $i \in V$ to node $j \in V$.

The following decision variables are used in the model:

x_{ij}: A binary variable. $x_{ij} = 1$ if the tour travels directly from node i to node j, and 0 otherwise.

y_i: A binary variable. $y_i = 1$ if node i is included in the tour, and 0 otherwise.

u_i: An auxiliary continuous variable used for Miller-Tucker-Zemlin (MTZ) subtour elimination. If node i is visited, it represents the position of node i in the sequence of the tour.

Objective Function The objective is to minimize the total cost of the tour, which is the sum of the costs of all selected arcs:

$$\min Z = \sum_{i \in V} \sum_{j \in V, i \neq j} c_{ij} x_{ij} \tag{21}$$

Constraints The minimization of the objective function is subject to the following constraints.

$$\sum_{i \in V_p} y_i = 1, \forall p \in \{1, 2, \ldots, m\} \tag{22}$$

$$\sum_{j \in V, j \neq i} x_{ji} = y_i, \forall i \in V \tag{23}$$

$$\sum_{j \in V, j \neq i} x_{ij} = y_i, \forall i \in V \tag{24}$$

$$y_i \leq u_i \leq m \cdot y_i, \forall i \in V \tag{25}$$

$$u_i - u_j + m \cdot x_{ij} \leq m - 1, \forall i, j \in V, i \neq j \tag{26}$$

$$x_{ij} \in \{0, 1\}, \forall i, j \in V, i \neq j \tag{27}$$

$$y_i \in \{0, 1\}, \forall i \in V \tag{28}$$

$$u_i \geq 0, \forall i \in V \tag{29}$$

Constraint(22) ensures that exactly one node is selected and included in the final tour from each of the m clusters. Constraint(23) and (24) ensure that if a node i is selected for the tour (i.e., $y_i = 1$), then exactly one arc must enter that node, and exactly one arc must leave it. If a node is not selected ($y_i = 0$), no arcs can be incident to it, this guarantees proper connectivity at the node level. Constraints(25) and (26) are the MTZ subtour elimination constraints, designed to ensure a single, continuous tour. Constraint(25) represents that if node i is visited ($y_i = 1$), u_i takes a value between 1 and m (representing its position in the m-node tour), and $u_i = 0$ if node i is not visited. Constraint(26) then imposes a sequential order: if the tour travels from node i to node j

($x_{ij} = 1$), this implies $u_j \geq u_i + 1$, meaning node j must appear after node i in the sequence, thereby preventing the formation of premature cycles or subtours before all m clusters have been visited. Constraints(27 - 29) are the variable constraints.

The GTSP is an NP-hard combinatorial optimization problem. This implies that finding a provably optimal solution using exact algorithms based on the above ILP formulation can be computationally intractable for large-scale instances. This intractability underscores the necessity for advanced approaches, such as Deep Reinforcement Learning, to effectively tackle such complex problems by learning high-quality solution policies.

B Training Procedure

The training algorithm for our MMFL framework is summarized in Algorithm 1. We employ the REINFORCE algorithm to train our model. To reduce the variance of the policy gradient and stabilize the training process, we incorporate a shared baseline. This baseline is computed for each problem instance by averaging the rewards obtained from multiple rollouts (i.e., multiple generated solution tours). For each GTSP instance λ_i, we sample k different routes $\pi^{i,j}$ using the SAMPLEROLLOUT function (line 4), which utilizes the multi-start decoder to generate feasible solutions from different starting nodes based on the k-nearest neighbors principle. We then compute the shared baseline $b(\lambda_i)$ for each instance, which is the average reward across the k solutions (line 5).

In line 7, we calculate the policy gradient $\nabla_\theta \mathcal{L}(\theta|\lambda_i)$ using the core REINFORCE formula. Here, $\mathcal{R}(\pi^{i,j}|\lambda_i) - b(\lambda_i)$ serves as an advantage function, reducing the variance of the gradient estimates. The $\nabla_\theta \mathcal{L}(\theta|\lambda_i)$ term indicates increasing the probability of producing solutions with high rewards while decreasing the probability of solutions with low rewards. Finally, line 8 updates the model parameters θ using the Adam optimizer, adjusting the weights through the policy gradient and the learning rate α.

Algorithm 1: Training Algorithm for MMFL

Input: Initialize policy network p_θ with random weights θ, number of training epochs E, number of rollouts k, number of batches B per epoch, batch size N, learning rate α

1 **for** *epoch = 1, . . . , E* **do**
2 **for** *b = 1, . . . , B* **do**
3 **for** *i = 1, ..., N* **do**
4 $\pi^{i,j} \leftarrow \text{SAMPLEROLLOUT}(\lambda_i) \quad \forall j \in \{1, 2, \ldots, k\}$;
5 $b(\lambda_i) \leftarrow \frac{1}{k} \sum_{j=1}^{k} \mathcal{R}(\pi^{i,j})$;
6 **end**
7 $\nabla_\theta \mathcal{L}(\theta|\lambda_i) \simeq \frac{1}{kN} \sum_{i=1}^{N} \sum_{j=1}^{k} (\mathcal{R}(\pi^{i,j}|\lambda_i) - b(\lambda_i)) \nabla_\theta \log p_\theta(\pi^{i,j}|\lambda_i)$;
8 $\theta \leftarrow \theta + \alpha \nabla_\theta \mathcal{L}(\theta|\lambda_i)$;
9 **end**
10 **end**

C Ablation Study

Our ablation study demonstrates the critical contribution of each component in the MMFL framework for solving GTSP. The results are shown in Table 4. For smaller problem instances ($n = 20, c = 4$ and $n = 50, c = 10$), all model variants perform similarly, but as problem complexity increases, the advantages of our complete architecture become evident. Without the Image Encoder, performance gaps range from 1.59% to 10.00%, while removing the Multimodal Fusion module causes even larger degradation (up to 35.83% for Large group distributions).

The most significant performance differences appear in complex spatial configurations and heterogeneous structures (Hybrid, Mixed, and Large groups), confirming that MMFL's strength lies in effectively integrating both topological and geometric information. Importantly, all model variants maintain comparable inference times, demonstrating that our approach achieves superior solution

quality without sacrificing the computational efficiency required for real-time robotic task planning.

Table 4: Ablation study.

Instances	MMFL		w/o Multimodal Fusion			w/o Image Encoder		
	Obj.	Time	Obj.	Gap	Time	Obj.	Gap	Time
$n = 20, c = 4$	**1.13**	0.12	1.13	0.00	0.11	1.13	0.00	0.12
$n = 50, c = 10$	**1.71**	0.11	1.71	0.00	0.12	1.71	0.00	0.09
$n = 100, c = 20$	**2.25**	0.13	2.31	2.67	0.11	2.69	19.56	0.11
$n = 150, c = 30$	**3.20**	0.13	3.32	3.75	0.13	3.53	10.31	0.12
$n = 200, c = 40$	**3.63**	0.14	3.74	3.03	0.13	3.78	4.13	0.12
Random	**2.37**	0.13	2.41	1.69	0.12	2.47	4.22	0.12
Proximity	**3.15**	0.15	3.20	1.59	0.13	3.22	2.22	0.12
Density	**1.47**	0.09	1.50	2.04	0.10	1.56	6.12	0.08
Hybrid	**2.16**	0.12	2.27	5.09	0.11	2.42	12.04	0.10
Uniform	**2.10**	0.12	2.23	6.19	0.12	2.31	10.00	0.10
Small	**3.92**	0.16	3.98	1.53	0.14	4.12	5.10	0.14
Large	**1.20**	0.12	1.32	10.00	0.11	1.63	35.83	0.10
Mixed	**2.01**	0.13	2.21	9.95	0.13	2.36	17.41	0.11