Factor Graph Neural Network Meets Max-Sum: A Real-Time Route Planning Algorithm for Massive-Scale Trips

Yixuan Li
Key Laboratory of New Generation Artificial Intelligence Technology and Its Interdisciplinary Applications, School of Computer Science and Engineering, Southeast University, Nanjing, China
yixuanli@seu.edu.cn

Wanyuan Wang*
Key Laboratory of New Generation Artificial Intelligence Technology and Its Interdisciplinary Applications, School of Computer Science and Engineering, Southeast University, Nanjing, China
wywang@seu.edu.cn

Weiyi Xu
Key Laboratory of New Generation Artificial Intelligence Technology and Its Interdisciplinary Applications, School of Computer Science and Engineering, Southeast University, Nanjing, China
wxu79631@gmail.com

Yanchen Deng
School of Computer Science and Engineering, Nanyang Technological University, Singapore
ycdeng@ntu.edu.sg

Weiwei Wu
School of Computer Science and Engineering, Southeast University
Nanjing, China
weiweiwu@seu.edu.cn

ABSTRACT

Global route planning (GRP) is a typical combinatorial optimization problem that has been solved for a variety of industrial purposes, such as traffic flow management, network routing, and conflict prevention. The goal of the GRP is to find a route for each trip query such that all queries have a minimum global travel time. The GRP problem is NP-hard and computationally challenging, even for medium-sized instances. However, in real-world GRP applications, such as Google Maps-based vehicle route guidance systems, there are always massive-scale trips issued simultaneously, and real-time response is required. Existing mathematical programming-based exact methods and heuristics struggle to balance the extremes of optimality and scalability. Considering that many closed-related GRP instances must be solved repeatedly, this paper explores a deep learning approach to learn real-time and efficient solutions for GRP. This paper first proposes a novel route-query factor graph (RQ-FG) to model the GRP problem, where the message-passing damped Max-sum (DMS) algorithm can be exploited to generate high-quality approximate solutions. A hybrid pruning method is proposed to accelerate solving the DMS. We further devise a route-query factor graph neural network (RQ-FGNN) based on the RQ-FG, which has the ability to return solutions in milliseconds. Experiments demonstrate that our method can generate high-quality solutions in massive-scale GRP instances in real-time.

KEYWORDS

Massive-Scale Route Planning; Factor Graph; Max-sum; Graph Neural Networks

*Corresponding author

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand. © 2024 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

ACM Reference Format:
Yixuan Li, Wanyuan Wang, Weiyi Xu, Yanchen Deng, and Weiwei Wu. 2024. Factor Graph Neural Network Meets Max-Sum: A Real-Time Route Planning Algorithm for Massive-Scale Trips. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024*, IFAAMAS, 9 pages.

1 INTRODUCTION

The global route planning (GRP) problem is defined in terms of a traffic-aware road network and a set of trip queries, each query consists of a pair of source and destination locations. The travel time on each road is related to the number of vehicles. The goal of the GRP is to generate a travel route for each query such that the sum of travel times for all queries is minimized [20, 21]. This kind of combinatorial optimization (CO) problem has a broad range of applications, including traffic management in urban road networks [13, 25], network routing in computer networks [37], and congestion reduction for warehouses with large numbers of robots [18, 33]. Nonetheless, the GRP problem in the real world is challenging for two reasons: 1) it is NP-hard to solve and computationally expensive even for medium-sized instances, and 2) with the increasing use of private cars, there might be massive-scale users issuing query trips simultaneously or within a short period of time.

The GRP problem has been extensively investigated in the Operations Research community, where mathematical programming such as Mixed Integer Programming (MIP) can be used to model GRP with different objectives, such as minimizing global travel time [20–22] and minimizing individual delay [9]. Based on mathematical programming, exact solutions by commercial solvers (e.g., Gurobi [11]) or approximate solutions by Lagrangian relaxations [23] can only apply to small-scale instances (e.g., tens of queries) within a limited time. To alleviate the computation cost, Li et al. [20, 21] recently proposed a depth-first search heuristic, and Luo et al. [24] proposed a hierarchical Monte Carlo Tree Search (MCTS) algorithm to generate routes for queries. These online search-based heuristics struggle to balance these extremes of optimality and efficiency and scale poorly to scenarios with massive-scale concurrent trips.

In practical settings, the closed-related GRP instances sharing similar patterns (e.g., the same road network and the set of candidate paths for each query is invariant) must be solved repeatedly. Therefore, the idea of learning the input-output mapping of the combinational optimization problem is promising for both optimality and scalability [3, 28]. The success of deep learning methods relies on a collection of historical data, including instance formulations and their ground truth solutions. However, the GRP problem is NP-hard, and it is a time-consuming and resource-intensive process to generate the instance data. For example, modeling the GRP problem as an MIP and generating the exact solutions for learning is infeasible because of time constraints [27, 35, 40]. On the other hand, learning the solutions returned by approximations may be inherently more difficult because these approximations may produce radically different solutions [15]. Moreover, these approximations might require domain experts, which cannot be generalized.

To this end, this paper first proposes a general route-query factor graph (RQ-FG) to model the GRP problem, in which each query is modeled as a variable and a road is modeled as a factor function. Based on the RQ-FG, the message-passing Max-sum algorithm can be used to generate high-quality solutions, which can be used directly in time-insensitive scenarios while simultaneously collecting historical data. We further develop a route-query factor graph neural network (RQ-FGNN) to represent the characters of the RQ-FG [41]. The RQ-FGNN learns the operation of the Max-sum algorithm, which can parameterize the Max-sum approximation, and is able to solve problems solvable by Max-sum in a real-time manner.

In summary, the contributions of this paper can be summarized as follows: (1) We first propose a generalized RQ-FG model for the GRP problem and extend the message-passing Max-sum algorithm to generate high-quality solutions (i.e., routes for queries). (2) By exploiting the structure of the GRP problem, an innovative hybrid pruning technique is proposed to accelerate the Max-sum algorithm. (3) We extend the Factor Graph Neural Network (RQ-FGNN) to learn representations of factors and variables in the RQ-FG. (4) Finally, experiments show that, compared to state-of-the-art benchmarks, the Max-sum method can provide better solutions after a few iterations, and RQ-FGNN can provide efficient solutions in real-time.

2 RELATED WORK

Heuristics for Global Route Planning. With the continued proliferation of GPS-enabled online map-based services (e.g., vehicle navigation systems), optimal route planning is extensively investigated to meet a user-specified preference for a single query [5, 19, 39]. Existing studies of achieving global route planning goals to reduce global traffic congestion can be regarded as the flow assignment problem by a probabilistic path choice [1, 22], where the routes are pre-defined and they search for an optimal flow assignment to each route. To minimize the global travel time, an iterative depth-first route search heuristic [20, 21] and a hierarchical Monte Carlo Tree Search (MCTS) heuristic [24] are proposed to generate routes for queries. However, these online search methods cannot apply to massive-scale scenarios with thousands of queries issued simultaneously.

Deep Learning for Combinatorial Optimization (CO). By exploiting shared structure among instances in the historical data, deep learning offers to automatically construct better heuristics from the data for CO [6, 27, 40]. However, deep learning-based CO relies on a collection of pre-solved instances, which is often a time-consuming process [15]. For example, the naive supervised-learning approach [10, 16] needs the ground truth, i.e., an actual optimal solution for each instance that can be obtained by solving the problem with an optimization solver, as the labels with the objective of deep learning. Existing CO models based on mathematical programming (e.g., mixed-integer linear programming (MILP)) may not be practical since the data augmentation process that uses an optimization solver can be very time-consuming [28, 34]. In contrast, this paper proposes a factor graph to model the global route planning problem, where the Max-sum algorithm can provide high-quality approximate solutions, which will simplify the learning process.

Graph neural network (GNN), due to its favourable properties, has recently been successfully used as a suitable model to represent dependencies between variables, objectives, and constraints in CO [6]. The proposed factor graph structure (FGNN) extends GNN to capture higher-order dependencies of multiple variables [41]. FGNN has recently been proposed for marginal probability inference in Probabilistic Graph Models [17, 32], while in this paper, FGNN is extended to Max-sum for optimizing a set of factor objectives.

3 PROBLEM FORMULATION

Road Network. The road network can be represented as a connected graph $G(V, E_g)$. Here, $V = \{v_1, v_2, \ldots, v_n\}$ denotes the set of vertices in the network, each representing an intersection within a city's road system. $E_g = \{e(v_i, v_j)\}_{v_i, v_j \in V}$ represents the set of edges, each $e\left(v_i, v_j\right)$ corresponds to a road segment starting from v_i and ending at v_j. A route is denoted as π, which is defined as a finite sequence of vertices $\langle v_1, v_2, \ldots, v_k \rangle$.

Definition 1: Time flow function. Let $t(e(v_i, v_j))$ be the time required for a vehicle to traverse edge $e(v_i, v_j)$ on its route:

$$t\left(e\left(v_i, v_j\right)\right) = t_{min}(e\left(v_i, v_j\right)) \times (1 + \alpha_e \chi_e), \tag{1}$$

where t_{min} represents the minimum time to traverse a particular road, i.e., the travel time without traffic congestion. α_e is a parameter related to the width of the road segment, and χ_e denotes the total number of vehicles on this edge. To simplify the problem, the minimum travel time t_{min} is assumed to be proportional to the road length.

Definition 2: Route travel time. For a route $\pi = \langle v_1, v_2, \ldots, v_k \rangle$, the travel time $T(\pi)$ is the cumulative travel time for each road $e(v_i, v_j)$ within the route:

$$T\left(\pi\right) = \sum_{e(v_i, v_j) \in \pi} t\left(e\left(v_i, v_j\right)\right). \tag{2}$$

Global Route Planning (GRP) Problem. Massive-scale route planning aims to handle all user queries. Queries can be denoted as $Q = \{q_1, q_2, q_3, \ldots, q_m\}$. Each query can be represented as $q_i = [s_i, d_i, \tau_i]$, where s_i is the source vertex, d_i is the destination, and τ_i indicates the beginning time of the query, i.e., the moment when the vehicle is at s_i. Time is divided into equal intervals, and queries within the same interval are assumed to be requests at the same

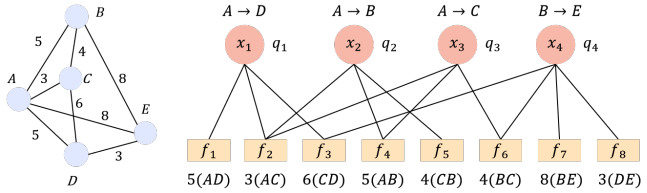

Figure 1: Visual representation of a road network (left) and a route-query factor graph corresponding to four queries on the road network (right). In the factor graph, factor nodes are shown as squares, and variable nodes are shown as circles. Each variable node represents a query, and factor nodes connected to variable nodes represent the specific roads within the candidate routes for the query. The number of candidates K in this instance is 2.

moment. The objective of the GRP problem is, given the queries Q, to generate a set of routes $\Pi = \{\pi_1, \pi_2, \pi_3, \ldots, \pi_m\}$, such that the total driving duration $GT(\Pi)$ is minimized:

$$GT(\Pi) = \sum_{i=1}^{|\Pi|} T(\pi_i). \tag{3}$$

4 FACTOR GRAPH MODEL AND MAX-SUM ALGORITHM

In this section, we present a novel graphical model for the GRP problem. Then we use the damped Max-sum algorithm [8] to solve the problem. Based on the characteristics of the GRP problem, we design an innovative hybrid pruning method to accelerate the Max-sum algorithm.

4.1 Route-Query Factor Graph

A factor graph is a bipartite graph model that represents the decomposition of a global function. Nodes can be categorized into variable nodes and function (factor) nodes. Variable nodes and factor nodes are connected if and only if the variable is in the scope of the function. To solve the GRP problem, we devise a model named route-query factor graph, where each variable node denotes a query and each function node denotes a road within the routes. The assignments of the variable nodes correspond to the selection of the routes, and the function nodes correspond to the travel cost, which provides the basis for route selection. This modeling approach has the ability to coordinate the route selection strategies for all queries.

To be specific, we model the GRP problem as a factor graph $FG = \langle X, D, F, E_{fg} \rangle$, where $X = \{x_1, \ldots, x_n\}$ are the variable nodes. These nodes represent the corresponding user queries $Q = \{q_1, \ldots, q_n\}$. $D = \{D_1, \ldots, D_n\}$ is the domain sets that the assignment of variable node x_i is selected from the domain D_i, where the assignment represents the identity number of a candidate route for the query q_i.

In order to avoid congestion, considering the massive queries, we need to select a suitable route from multiple candidates. We design a $top-K$ method to make a trade-off between the computational overhead and the solution quality of the computation. That means narrowing the list of candidates by selecting the shortest K routes for each query, where K is a manually defined hyperparameter. Since the assignment of the variable x_i is the route choice of the query q_i, it contains K candidates in domain D_i as $1, 2, \ldots, K$, where each corresponds to a candidate route $\pi_i^j, j \in 1, \ldots, K$ from the set $\Pi_i = \{\pi_i^1, \ldots, \pi_i^K\}$. That is, the domain D_i implies the selection set for the $q_i th$ query, which contains the shortest K candidate routes.

In the GRP problem, each route consists of multiple specific roads, and the travel time of the query is based on these road segments. We model the specific roads as the function nodes $F = \{f_1, \ldots, f_m\}$, where F is a finite set of constraint functions. The variable nodes connected to a function node f_j are represented as S_j. Function $f_j : \times_{x_i \in S_j} D_i \rightarrow \mathbb{R}^+ \bigcup \{0\}$ maps the queries' assignment to a non-negative real number, which is the total travel cost of all vehicles passing through the road. The value of a function node f_j depends merely on its connected variable node S_j. The undirected edges in the bipartite graph are defined by E_{fg} (e.g., $e_{i,j} \in E_{fg}$ is the edge that connects x_i and f_j). Eventually, the GRP problem is to select an assignment to the variable nodes X such that the sum of function nodes F is minimized.

We present an example of our route-query factor graph through Figure 1, which illustrates the road network and the factor graph corresponding to its queries. The left-hand side of Figure 1 shows a road network $G(V, E_g)$, where the vertices of the road network are $V = \{A, B, C, D, E\}$ and the roads E_g are shown in the graph. Assuming we have four queries simultaneously, namely $A \rightarrow D, A \rightarrow B, A \rightarrow C$ and $B \rightarrow E$. These queries are modeled as the variable nodes $X = x_1, x_2, x_3$ and x_4. We simply select two as the hyperparameter K of the $top-K$ method in this example, which means each query has two candidate routes to be chosen (the domain D_i of variable x_i is 1, 2). Take query $A \rightarrow D$ (variable node x_1) as an example, the shortest two routes of this query are $\langle A, D \rangle$ and $\langle A, C, D \rangle$. So the function nodes connected to variable node x_1 are the edges $e(A, D)$, $e(A, C)$ and $e(C, D)$. Then we can get the function nodes $F = \{f_1, \ldots, f_8\}$ and the edges E_{fg} for all the queries. Finally, we can construct the route-query factor graph $FG = \langle X, D, F, E_{fg} \rangle$, which is shown in the right-hand of Figure 1.

4.2 Damped Max-sum with Pruning

In order to solve the GRP problem, we apply the damped Max-sum algorithm to the route-query factor graph. We further devise a threshold-based hybrid pruning strategy to speed up the execution process of the algorithm.

4.2.1 Damped Max-sum Algorithm. The complete algorithms [4, 7] may take exponential time in the worst case, which is not suitable for our time-sensitive application scenarios. The Max-sum algorithm is an inference algorithm based on belief propagation that collects global information by exchanging messages with neighboring nodes on the factor graph. The message passing of Max-sum can be divided into two phases: the query message from variable nodes to function nodes and the response message in the opposite direction. In each iteration, the variable node first sends a query message to its neighboring function node:

$$Q^{\kappa}_{x_i \rightarrow f}(x_i) = \sum_{f' \epsilon N(x_i) \setminus f} R^{\kappa-1}_{f' \rightarrow x_i}(x_i) + \alpha_i \tag{4}$$

Where $Q^{\kappa}_{x_i \to f}(x_i)$ represents the message sent by the variable node x_i to the function node f at the κth iteration, and $N(x_i)$ denotes all the neighbors of the variable node x_i. $R^{\kappa-1}_{f' \to x_i}(x_i)$ is the response message sent by the neighboring nodes f' of the variable node x_i other than the target node at the $\kappa - 1th$ iteration. α_i is the regularization term used in order to prevent the message from growing unboundedly, which can be defined as:

$$\alpha_i = -\frac{1}{|D_i|} \sum_{x_i \in D_i} Q^{\kappa}_{x_i \to f}(x_i) \tag{5}$$

In the first iteration, the initial value of the query message sent from the variable node to the function node is 0. The function node f computes the response message $R^{\kappa}_{f \to x_i}(x_i)$ and sends it to the variable node. The computation process of the response message consists of summation (sum) and maximization (max)[1]. The message is the minimal marginalization of function f with respect to variable x_i. Formally, the response message $R^{\kappa}_{f \to x_i}(x_i)$ is shown in Equation (6):

$$R^{\kappa}_{f \to x_i}(x_i) = \min_{N(f) \setminus x_i} \left(\sigma(N(f)) + \sum_{x_i' \epsilon N(f) \setminus x_i} Q^{\kappa}_{x_i' \to f}\left(x_i'\right)\right) \tag{6}$$

where $\sigma(N(f))$ denotes the constraint function of the function nodes. In our scenario, the constraint function is the total travel time of all vehicles on this road $e(v_i, v_j)$ corresponding to function node f:

$$\sigma(N(f)) = \sum_{i: e(v_i, v_j) \in q_i} t(e(v_i, v_j)) \tag{7}$$

After receiving the response messages, variable node x_i will accumulate the messages of all its neighbors and calculate its beliefs. The variable node x_i will then choose the optimal assignment $\tilde{d}_i$ based on its beliefs:

$$\tilde{d}_i = \arg\min_{x_i \in D_i} \sum_{f' \epsilon N(x_i)} R^{\kappa}_{f' \to x_i}(x_i) \tag{8}$$

On tree-structured graphs, Max-sum is able to converge to the optimal solution in linear time. However, when the factor graph contains multiple cycles, Max-sum may explore low-quality solutions. Damping [29] is a method that is incorporated with belief propagation to reduce the effect of excessive loopy propagation. This is achieved by balancing the new calculation with the calculations carried out in previous iterations. The use of damping during message propagation can improve the Max-sum algorithm [8]. Damping is typically added to the message $Q^{\kappa}_{x_i \to f}(x_i)$ sent from the variable node to the function node:

$$Q^{\kappa}_{x_i \to f}(x_i) = \lambda Q^{\kappa-1}_{x_i \to f}(x_i) + (1-\lambda) \sum_{f' \epsilon N(x_i) \setminus f} R^{\kappa-1}_{f' \to x_i}(x_i) + \alpha_i \tag{9}$$

which balances the message between the previous and new iterations by the weight parameter $\lambda \in (0, 1]$. When $\lambda = 0$, the resulting algorithm is standard Max-sum.

[1]In this minimization problem, it is Mini-sum, but we still call it Max-sum because it is widely accepted

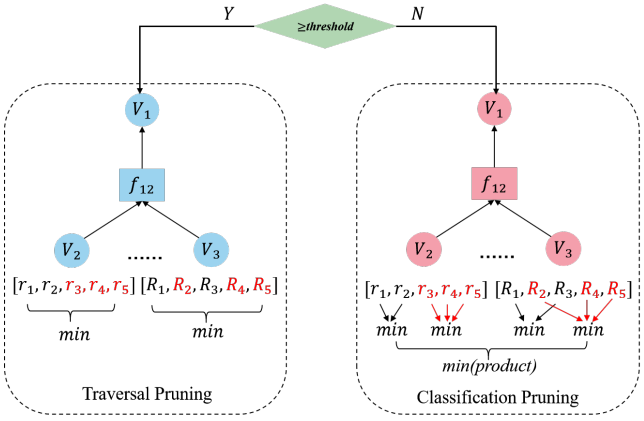

Figure 2: Threshold-based Pruning method. The function node calculates the response message with a threshold to determine which pruning method to use.

4.2.2 Threshold-based Pruning. The primary computational overhead of the Max-sum algorithm is in calculating the response messages sent from function nodes to variable nodes. As demonstrated in Equation (6), the computation overhead grows exponentially with the number of variables involved in the function node in the worst case. In our task designed for massive-scale trips, neighbors of the function node are queries that could potentially pass through this road segment. Suppose there are n variable nodes connected to a function node, and each variable node has K assignments (i.e., K candidate routes for the query). The constraint function then has K^n possible states, and the computational cost of optimization in these states will be relatively high. Thus, by exploiting the characteristics of GRP problems, we have designed two pruning strategies and employed a threshold-based method to reduce the time complexity of DMS in the worst case.

Classification pruning. When computing the response message $R^{\kappa}_{f \to x_i}(x_i)$ that the function node f sends to the variable node x_i, Max-sum needs to traverse all combinations of variable nodes $x_i' \in N(f) \setminus x_i$ connected to function node f except x_i. For the GRP problem, the function node f corresponds to a specific road, while the variable node x_i corresponds to a user query. Inspired by the "Fast Max-sum" [26, 38], considering the properties of queries, the values of variable nodes x_i, linked to function nodes f can be naturally divided into two categories. The first includes values for queries that will pass through this road, while the second comprises those that will not. Since the constraint function calculation is solely related to the number of vehicles passing through the edge, each category impacts the constraint function identically. Therefore, we only need to compute $Q_{x_i' \to f}(x_i)$ for each category and choose the smaller one. With classification pruning, the computational time complexity of the response messages is reduced from $O(K^n)$ to $O(2^n)$. This computational overhead is acceptable for function nodes with fewer neighbors, making it applicable to areas like suburban roads.

Traversal pruning. While the classification pruning method can significantly reduce computational time complexity, it's still resource-intensive for highly trafficked central road segments. Hence,

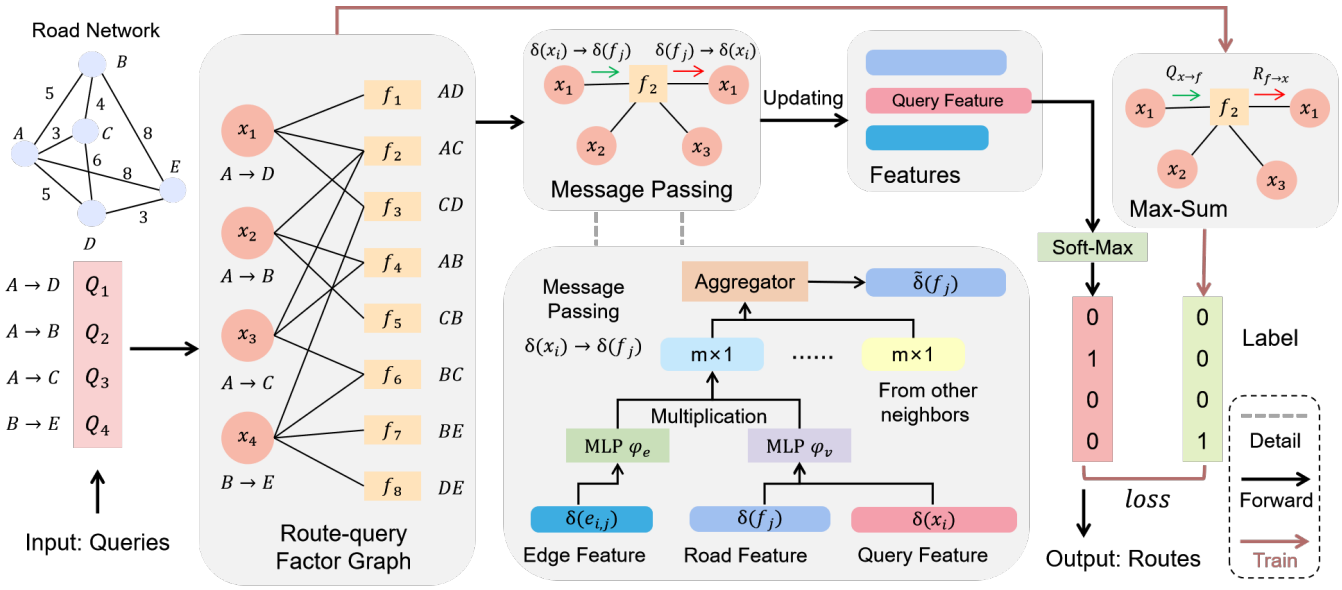

Figure 3: Overview of our proposed RQ-FGNN. The query at the same moment is taken as input and combined with the road network to model the route-query factor graph. The features of the nodes in the factor graph are updated by message passing, and the output is obtained from the features of the variable nodes. The results of Max-sum are used as labels during training.

we propose a traversal pruning method inspired by greedy approaches. When the function node f sends the response message $R^{\kappa}_{f\to x_i}(x_i)$, it no longer scans the entire space. Instead, it sequentially traverses the node set $N(f)\backslash x_i$. For each variable node x_i in $N(f)\backslash x_i$, it chooses a value that minimizes $Q_{x_i'\to f}(x_i)$ under the current state. This search strategy only considers the locally optimal values up to the current variable node, disregarding subsequent variable nodes' influence. Using this traversal concept, each variable node decides its value based on previously determined values, requiring just a single traversal through all x_i in $N(f)\backslash x_i$, reducing the time complexity to $O(Kn)$.

Threshold-based pruning. The two pruning methods can be applied to different road segments. We set a threshold ρ to strike a balance between the two methods. Specifically, when the number of neighboring nodes of a function node is less than or equal to ρ, we opt for the classification pruning method for precise computation. If the count exceeds ρ, we use traversal pruning for a rapid computation. By employing this threshold-based pruning optimization, we can ensure a balance between accuracy and efficiency, tailoring acceleration solutions flexibly for different road conditions. The threshold-based pruning optimization is depicted in Figure 2. In addition to our work, there are other runtime reduction methods [30, 31, 38] that have the potential to be applied to this scenario, where the variables can be divided into groups and the decision is only dependent on the number of variables in each group.

5 ROUTE-QUERY FACTOR GRAPH NEURAL NETWORK (RQ-FGNN)

In massive-scale route planning scenarios, a large number of repeated queries are generated every day. To quickly get a feasible

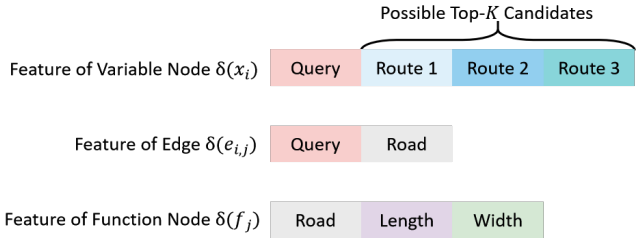

Figure 4: Node and edge features in RQ-FGNN.

solution in real-time scenarios, we design an end-to-end model route-query Factor Graph Neural Network (RQ-FGNN), which utilizes historical data and simulates the Max-sum process based on the route-query factor graphs to obtain the solution. This algorithm uses historical query data to train the model offline and makes fast online inferences using node representations and dependencies. Zhang et al. [41] proposed a Factor Graph Neural Network (FGNN), a neural network for message passing over factor graphs, and demonstrated that factor graph neural networks can accurately parameterize Max-Product belief propagation [2]. Our approach essentially parameterized Max-sum, a variant algorithm of Max-Product. In our scenario, we improve the FGNN and design a suitable framework, route-query Factor Graph Neural Network (RQ-FGNN), for our route-query factor graph.

5.1 Features

In the route-query factor graph, function nodes correspond to roads, and variable nodes correspond to queries. We denote the feature of a function node f_i as $\delta(f_i)$, which consists of the number of start and end, length, and width of the road. In this case, the length of a road

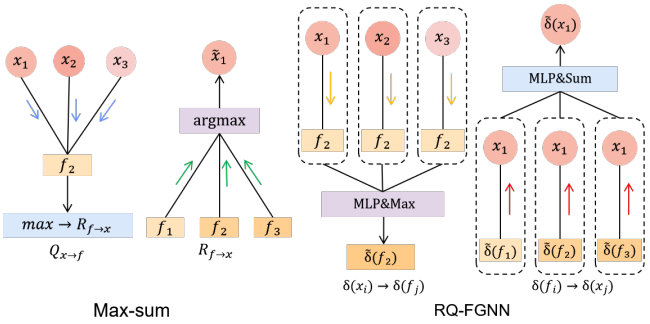

Figure 5: Message passing for Max-sum and RQ-FGNN.

determines the travel time of a single vehicle, and the width, such as the α_e in Equation (1), affects the travel time for multiple vehicles. For the edge $e_{i,j}$ between the variable node x_i and the function node f_j, the feature $\delta(e_{i,j})$ contains the identical number of the variable node and the function node. For variable node x_i, its feature $\delta(x_i)$ contains the origin and destination of the query and the $top-K$ candidate routes. Our feature setup is shown in Figure 4. We then encode the features through the network to propagate messages between nodes and obtain higher-order features.

5.2 Message Passing

In the Max-sum algorithm, the message passing can be divided into the query message sent by the variable node to the function node and the response message by the function node. Given a route-query factor graph $FG = \langle X, D, F, E \rangle$, for the query message sent from variable node x_i to function node f_j, the inputs are the features $\delta(x_i)$ of the variable node, the features $\delta(f_j)$ of the function node, and the features $\delta(e_{i,j})$ of the edge $e_{i,j}$. Concatenate $\delta(x_i)$ with $\delta(f_j)$, then input $\delta(e_{i,j})$ and $[\delta(x_i), \delta(f_j)]$ into the fully-connected layer, respectively, to obtain a $m \times n$ matrix and a $n \times 1$ vector. Then we multiply the two, and their product yields the query message $\delta(Q_i)$. After that, the function node f_j aggregates the query message $\{\delta(Q_i) | x_i \in N(f_j)\}$ from all the neighbors to update its feature $\delta(f_j)$. Specifically, the feature $\tilde{\delta}(f_j)$ of the function node is shown in Equation (10).

$$\tilde{\delta}(f_j) = \max_{x_i \epsilon N(f_j)} \varphi_e(\delta(e_{i,j}) | \Theta_{VF}) \varphi_v([\delta(x_i), \delta(f_j)] | \Phi_{VF}) \quad (10)$$

Where φ_e maps features to a $m \times n$ matrix and φ_v maps features to a $m \times 1$ vector. Θ_{VF} and Φ_{VF} represent the parameters of the model in variable-to-function message passing. Taking $[\Theta_{FV}, \Phi_{FV}]$ to denote the parameters of the model from the function-to-variable module, the updating process of the feature $\delta(x_i)$ of the variable node can be expressed as:

$$\tilde{\delta}(x_i) = \sum_{f_j \epsilon N(x_i)} \varphi_e(\delta(e_{i,j}) | \Theta_{FV}) \varphi_v([\delta(x_i), \delta(f_j)] | \Phi_{FV}) \quad (11)$$

Figure 5 illustrates the message passing process of RQ-FGNN and Max-sum.

5.3 Route Inference

We take the queries at the same moment as inputs to the model and construct a route-query factor graph according to the road network. After that, the model will perform message passing on the factor graph to update the features of each node and edge. After the features are updated, we input $\delta(x_i)$ of the variable nodes into the fully connected layer, and through the softmax layer, we can map the features of the variable nodes to a value $\hat{x_i}$.

$$\hat{x_i} = softmax(\varphi_c(\delta(x_i) | \theta)) \quad (12)$$

Where φ_c maps the feature $\delta(x_i)$ to a $k \times 1$ vector, and θ is the network parameter of the module. In the training phase, the result x_i^* is obtained by Max-sum calculation, and the model can be trained by the cross-entropy loss.

$$Loss(\hat{x_i}, x_i^*) = -\sum_{i=1}^{|X|} \hat{x_i} log(x_i^*) \quad (13)$$

The loss of each item is shown in Equation 13. In the inference phase, the output of the model $\hat{x_i}$ is taken as the corresponding route selection for each query. The overall framework of the method is shown in figure 3.

6 EXPERIMENT

6.1 Datasets and Baselines

We construct a fully connected graph with 2000 vertices and 3771 edges as the synthetic road network (SG). We also chose the San Joaquin County Road Network (TG)[2] as the real road network. We select an area from the entire urban area that has the same count of vertices as the SG and add the necessary edges to make it a fully connected graph, which has 2000 vertices and 4958 edges. We randomly generate queries, and to prevent queries that are too close in distance, we set the minimum length for query generation to 10 road segments. We compare our method with the individual-based search algorithm (IND) and the self-aware batch process (SBP). The IND algorithm finds the shortest path for each query separately under the traffic situation [36]. In [21], the state-of-the-art method named SBP algorithm consists of an initial greedy-based search with subsequent refining steps.

6.2 Metrics

There are two widely used metrics for evaluation: the computation time and the global travel time $GT(\Pi)$ in Equation (3), which are also used in [21]. The $GT(\Pi)$ metric Calculate the time taken to pass through each road in real time, and count the total travel time of all the queries under expectations. However, the time taken for vehicles to traverse each road segment depends on the travel time on the preceding roads, so the traffic volume on each edge is uncertain. Existing methods for solving the GRP assume that traffic generated by queries will arrive at the next intersection on time, yet they overlook delays caused by factors such as traffic lights and unexpected incidents. Therefore, we are the first to propose the "worst-case travel time" metric $GT_{worst}(\Pi)$ as an auxiliary evaluation metric. Compared to the $GT(\Pi)$, the $GT_{worst}(\Pi)$ considers the impact of queries on road congestion in terms of an entire route rather than decomposing this route into individual roads over time. That is, when a query is assigned to route π, the count of vehicles (χ_e in Equation 1) on all road segments along this route increments

[2]https://users.cs.utah.edu/ lifeifei/SpatialDataset.htm

by one. This metric removes the effect of uncertainty in arrival times and represents the upper bound of the total travel time for all vehicles caused by queries.

6.3 Experimental Settings

The minimum travel time $t_{min}(e(v_i, v_j))$ is a randomly generated value ranging from 10 to 200 (seconds) for each edge. The parameter α_e in Equation (1) measures the travel time, represents parameters such as the width of the road, and is randomly set between 0.1 and 2. A larger α_e makes it more likely that congestion will occur, allowing a better assessment of the method's ability to avoid congestion. We assume that the non-query traffic is relatively stable over a time period. Specific road conditions and traffic flow simulation experiments are out of the scope of our study. For the damping factor λ in Equation (9), we adjust it between 0 and 0.9 during training. The threshold parameter ρ for pruning can be adjusted depending on the road network, and our default setting is 10. We use the networkx [12] to construct the road network graph, calculate and store the shortest K distance between any two nodes in advance. The experiments are conducted on an AMD R9 3950X processor (3.5 GHz) and GeForce RTX 2080ti GPUs. Note that the Max-sum algorithm naturally supports parallel computation. Therefore, we parallelize the message-passing process during the training process.

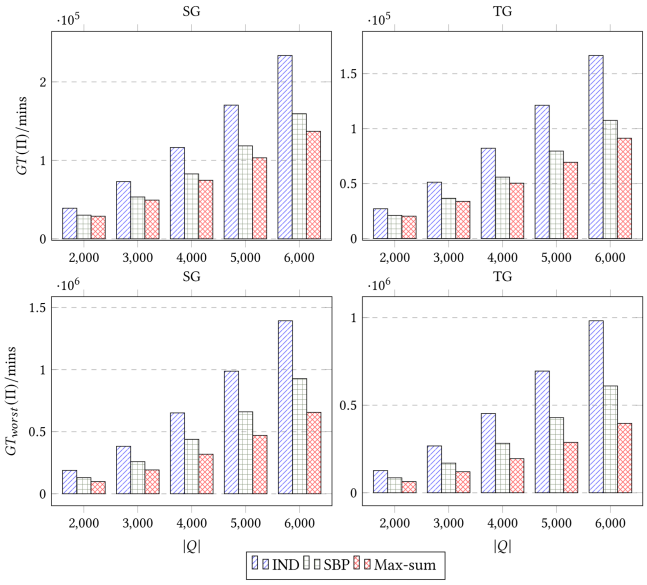

Figure 6: The performance of the Max-sum after 20 iterations and the baselines under different numbers of queries.

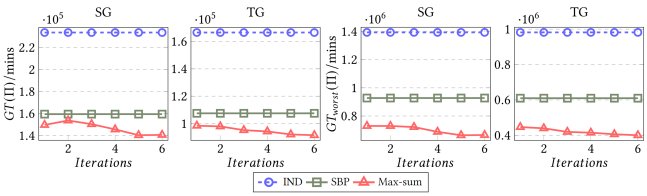

Figure 7: Performance of Max-sum after each iteration when query count is 6000.

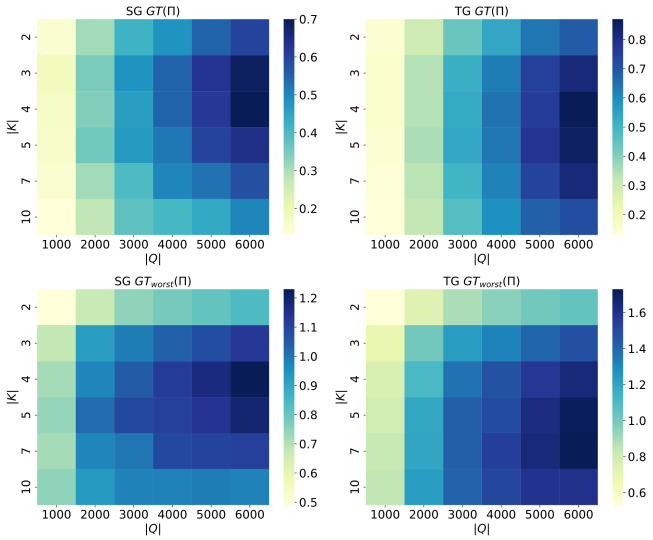

Figure 8: The performance of Max-sum under different numbers of candidate routes and queries after 20 iterations.

6.4 Experimental Results

The Performance of the Max-sum. We first present the performance of the Max-sum and analyze some factors that affect the performance. We make the number of simultaneous arrivals of query 100 and set the batch size of sbp to 100 and ϵ to 0.001, which are the optimal hyperparameters found in [21]. Figure 6 shows the results achieved by Max-sum after 20 iterations when $K = 3$. It is evident that the Max-sum algorithm outperforms all baseline methods in both metrics, effectively reducing the total travel time caused by the massive-scale queries. As the number of queries increases, the performance of Max-sum improves more significantly than the baseline methods. As Max-sum makes decisions based on the selection of an entire route, it shows more improvements over baseline methods in the $GT_{worst}(\Pi)$ metric.

Figure 7 illustrates the performance of Max-sum during iterations when $K = 3$ and query count is 6000. As can be seen from the figure, both metrics are trending down as the number of iterations of the Max-sum algorithm increases. It is observable that Max-sum had already surpassed the performance of IND and SBP in both metrics after the first iteration. In fact, for $K < 5$, the time taken for each iteration of Max-sum is only on the order of seconds even without parallel computation, indicating that the Max-sum algorithm without neural networks is already capable of real-time applications in some scenarios.

Hyperparametric analysis. We analyze how the candidate routes K affect the performance of the Max-sum method. We fix the number of iterations as 20 for Max-sum and obtain the results shown in Figure 8. The values in the graph are the percentage improvement relative to the IND algorithm (i.e., $\frac{GT(\Pi)_{IND} - GT(\Pi)_{Max-sum}}{GT(\Pi)_{Max-sum}}$), with darker colours meaning greater improvement. A suitable K may yield better performance in some scenarios due to the number of iterations and pruning threshold ρ. A larger K usually makes the model more difficult to converge, but the ability of the Max-sum algorithm may be enhanced as a trade-off, which should be considered in real-world applications.

Table 1: Effect of training set size

Maps	*Methods*	*Number of samples*				
		200	*400*	*600*	*800*	*1000*
SG	GNN	61.35%	73.20%	80.56%	79.29%	82.70%
	RQ-FGNN	64.19%	77.25%	85.47%	85.72%	88.76%
TG	GNN	63.74%	74.39%	84.45%	85.13%	85.78%
	RQ-FGNN	67.02%	78.91%	89.02%	90.20%	92.53%
Average Gain		**3.06%**	**4.29%**	**4.74%**	**5.75%**	**6.41%**

Table 2: Performance of models under different K

Maps	*Methods*	*Value of K*				
		2	*3*	*4*	*5*	*6*
SG	GNN	85.24%	82.70%	79.71%	77.34%	71.39%
	RQ-FGNN	89.32%	88.76%	85.12%	82.41%	77.82%
TG	GNN	87.05%	85.78%	80.43%	76.58%	73.31%
	RQ-FGNN	92.11%	92.53%	87.62%	83.36%	78.14%
Average Gain		**4.57%**	**6.41%**	**6.30%**	**5.93%**	**5.63%**

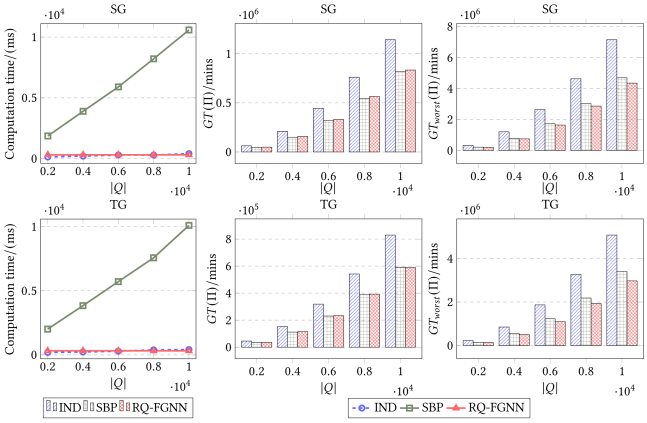

Figure 9: Performance and the computational overhead of methods corresponding to different numbers of queries.

Learning ability of RQ-FGNN. Then we analyze the ability of RQ-FGNN to fit the Max-sum algorithm. We compare RQ-FGNN with commonly used GNN, whose structure consists of a stack of linear, convolutional, and attention convolutional layers. We maintained the number of iterations at 20 for the Max-Sum algorithm to generate training datasets. Unifying different labels associated with the same query into the most frequent label leads to more stable network convergence without significantly reducing algorithm performance. We conducted experiments by controlling the number of potential query-generating vertices to 200. In order to make the problem scenario more realistic and reduce the sample space for the neural network, we obtain the query through weighted sampling. That means downtown areas with heavy traffic are more likely to generate queries than remote areas. We randomly assign each vertex a weight between 1 and 10, with the weight determining the vertex's probability of generating a query.

To explore how neural networks' dependency on prior solutions affects performance, we first examine the effect of varying training set sizes. We report the optimal network performance achieved with consistent hyperparameters. The experiments were conducted with 100 simultaneous queries, making each sample a factor graph comprising 100 variable nodes. We use Adam [14] as the optimizer. Table 1 shows the fitting performance under testing sets with 100 samples, revealing that RQ-FGNN outperforms conventional GNNs across various dataset sizes, with fitting performance gradually improving as the training set enlarges. The commonly used GNN can also achieve good accuracy, which demonstrates the effectiveness of our factor graph structure and feature design. As 200 query-generating vertices are sufficient for application in real scenarios (e.g., the maps in [24] contain hundreds of vertices), we believe that adequate training can leverage our algorithms to handle scenarios with more query-generating vertices and larger map scales.

Then we present the effect of different K values. As can be seen from Table 2, RQ-FGNN fits the Max-sum better under every K, which shows that the RQ-FGNN structure can effectively learn the message propagation of Max-sum. Because K has an effect on the size of the factor graph and the size of the features, increasing K may require networks with more parameters. Note that there may be a sample imbalance problem with increasing values of K, which can affect the learning ability of the neural network.

The performance of RQ-FGNN. Finally, we compare our RQ-FGNN with the baselines of performance and computational overhead. Considering the effect of K on Max-sum and RQ-FGNN, the experiment was conducted in the case of $K = 3$. We selected the model with the highest accuracy in our previous experiments. The results are shown in Figure 9, where the horizontal coordinate is the number of accumulated queries. It can be seen that RQ-FGNN requires less time than the SBP method, within one second. Because the neural networks can perform batch processing, the computation time does not obviously increase as the number of queries grows. Meanwhile, RQ-FGNN outperforms the IND method in both datasets and metrics, significantly cutting global travel time, which proves the effectiveness of RQ-FGNN in avoiding congestion. Compared to the SBP method, our method achieves similar performance while consuming significantly less time.

7 CONCLUSION

In this paper, we propose a novel graph model named the route-query factor graph for the GRP problem. We apply a damped Max-sum method and design a hybrid pruning approach based on the characteristics of the GRP problem, which can return high-quality solutions. We further devised a message-passing route-query factor graph neural network (RQ-FGNN) to represent the route-query factor graph. Our experimental results show that the RQ-FGNN is able to generate plausible solutions in a real-time manner. Our factor graph modeling approach with DMS computation also supports deployment in distributed scenarios, which provides insights into research trends in privacy protection. Besides, RQ-FGNN requires a data labeling process where the label encoding, especially searching the ground truth marginals for hard instances, can be time-consuming. We will consider such factors and explore the theoretical guarantee of RQ-FGNN in our future work.

ACKNOWLEDGMENTS

This research was supported by the National Natural Science Foundation of China (62076060, 62072099, 61932007, and 61806053).

REFERENCES

[1] Abbas Babazadeh Babak Javani and Avishai (Avi) Ceder. 2019. Path-based capacity-restrained dynamic traffic assignment algorithm. *Transportmetrica B: Transport Dynamics* 7, 1 (2019), 741–764.

[2] Mohsen Bayati, Devavrat Shah, and Mayank Sharma. 2008. Max-product for maximum weight matching: Convergence, correctness, and lp duality. *IEEE Transactions on information theory* 54, 3 (2008), 1241–1251.

[3] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. 2021. Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research* 290, 2 (2021), 405–421.

[4] Dingding Chen, Yanchen Deng, Ziyu Chen, Wenxing Zhang, and Zhongshi He. 2020. HS-CAI: A hybrid DCOP algorithm via combining search with context-based inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 7087–7094.

[5] Lisi Chen, Shuo Shang, and Tao Guo. 2020. Real-Time Route Search by Locations. In *AAAI'20*. 574–581.

[6] Ziang Chen, Jialin Liu, Xinshang Wang, and Wotao Yin. 2023. On Representing Mixed-Integer Linear Programs by Graph Neural Networks. In *ICLR'23*.

[7] Ziyu Chen, Wenxin Zhang, Yanchen Deng, Dingding Chen, and Qing Li. 2020. RMB-DPOP: refining MB-DPOP by reducing redundant inferences. *arXiv preprint arXiv:2002.10641* (2020).

[8] Liel Cohen, Rotem Galiki, and Roie Zivan. 2020. Governing convergence of Max-sum on DCOPs through damping and splitting. *Artificial Intelligence* 279 (2020), 103212.

[9] Mathijs Michiel de Weerdt, Sebastian Stein, Enrico H. Gerding, Valentin Robu, and Nicholas R. Jennings. 2016. Intention-Aware Routing of Electric Vehicles. *IEEE Transactions on Intelligent Transportation Systems* 17, 5 (2016), 1472–1482.

[10] Ferdinando Fioretto, Terrence W. K. Mak, and Pascal Van Hentenryck. 2020. Predicting AC Optimal Power Flows: Combining Deep Learning and Lagrangian Dual Methods. In *AAAI'20*. 630–637.

[11] LLC Gurobi Optimization. 2021. Gurobi optimizer reference manual.

[12] Aric Hagberg and Drew Conway. 2020. Networkx: Network analysis with python. *URL: https://networkx. github. io* (2020).

[13] Devansh Jalota, Kiril Solovey, Matthew Tsao, Stephen Zoepf, and Marco Pavone. 2023. Balancing fairness and efficiency in traffic routing via interpolated traffic assignment. *Autonomous Agents and Multi-Agent Systems* 37, 2 (2023), 32.

[14] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[15] James Kotary, Ferdinando Fioretto, and Pascal Van Hentenryck. 2021. Learning Hard Optimization Problems: A Data Generation Perspective. In *NeurIPS'21*. 24981–24992.

[16] James Kotary, Ferdinando Fioretto, Pascal Van Hentenryck, and Bryan Wilder. 2021. End-to-End Constrained Optimization Learning: A Survey. In *IJCAI'21*. 4475–4482.

[17] Jonathan Kuck, Shuvam Chakraborty, Hao Tang, Rachel Luo, Jiaming Song, Ashish Sabharwal, and Stefano Ermon. 2020. Belief Propagation Neural Networks. In *NeurIPS'20*. 685–693.

[18] Jiaoyang Li, Andrew Tinka, Scott Kiesel, Joseph W. Durham, T. K. Satish Kumar, and Sven Koenig. 2021. Lifelong Multi-Agent Path Finding in Large-Scale Warehouses. In *AAAI'21*. 11272–11281.

[19] Jing Li, Yin David Yang, and Nikos Mamoulis. 2013. Optimal Route Queries with Arbitrary Order Constraints. *IEEE Transactions on Knowledge and Data Engineering* 25, 5 (2013), 1097–1110.

[20] Ke Li, Lisi Chen, and Shuo Shang. 2020. Towards Alleviating Traffic Congestion: Optimal Route Planning for Massive-Scale Trips. In *IJCAI'20*. 3400–3406.

[21] Ke Li, Lisi Chen, Shuo Shang, Panos Kalnis, and Bin Yao. 2021. Traffic Congestion Alleviation over Dynamic Road Networks: Continuous Optimal Route Combination for Trip Query Streams. In *IJCAI'21*. 3656–3662.

[22] Sejoon Lim and Daniela Rus. 2012. Stochastic distributed multi-agent planning and applications to traffic. In *ICRA'12*. 2873–2879.

[23] Meghna Lowalekar, Pradeep Varakantham, and Patrick Jaillet. 2018. Online spatio-temporal matching in stochastic and dynamic domains. *Artificial Intelligence* 261 (2018), 71–112.

[24] Guiyang Luo, Yantao Wang, Hui Zhang, Quan Yuan, and Jinglin Li. 2023. AlphaRoute: Large-Scale Coordinated Route Planning via Monte Carlo Tree Search. In *AAAI'23*. 12058–12067.

[25] Renshi Luo, Ton J. J. van den Boom, and Bart De Schutter. 2018. Multi-Agent Dynamic Routing of a Fleet of Cybercars. *IEEE Transactions on Intelligent Transportation Systems* 19, 5 (2018), 1340–1352.

[26] Kathryn Macarthur, Ruben Stranders, Sarvapali Ramchurn, and Nicholas Jennings. 2011. A distributed anytime algorithm for dynamic task allocation in multi-agent systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 25. 701–706.

[27] Vinod Nair, Sergey Bartunov, Felix Gimeno, Ingrid von Glehn, Pawel Lichocki, Ivan Lobov, Brendan O'Donoghue, Nicolas Sonnerat, Christian Tjandraatmadja, Pengming Wang, Ravichandra Addanki, Tharindi Hapuarachchi, Thomas Keck, James Keeling, Pushmeet Kohli, Ira Ktena, Yujia Li, Oriol Vinyals, and Yori Zwols. 2021. Solving Mixed Integer Programs Using Neural Networks. arXiv:2012.13349 [math.OC]

[28] Seonho Park and Pascal Van Hentenryck. 2023. Self-Supervised Primal-Dual Learning for Constrained Optimization. In *AAAI'23*. 4052–4060.

[29] Marco Pretti. 2005. A message-passing algorithm with damping. *Journal of Statistical Mechanics: Theory and Experiment* 2005, 11 (2005), P11008.

[30] Marc Pujol-Gonzalez, Jesus Cerquides, Gonzalo Escalada-Imaz, Pedro Meseguer, and Juan A Rodriguez-Aguilar. 2013. On binary max-sum and tractable hops. In *European Workshop on Multi-agent Systems (EUMAS)*.

[31] Marc Pujol-Gonzalez, Jesus Cerquides, Pedro Meseguer, Juan Antonio Rodríguez-Aguilar, and Milind Tambe. 2013. Engineering the decentralized coordination of UAVs with limited communication range. In *Advances in Artificial Intelligence: 15th Conference of the Spanish Association for Artificial Intelligence, CAEPIA 2013, Madrid, Spain, September 17-20, 2013. Proceedings 15*. Springer, 199–208.

[32] Victor Garcia Satorras and Max Welling. 2021. Neural Enhanced Belief Propagation on Factor Graphs. In *AISTATS'21*, Vol. 130. 685–693.

[33] Arambam James Singh and Akshat Kumar. 2019. Graph Based Optimization for Multiagent Cooperation. In *AAMAS'19*. 1497–1505.

[34] Zhihai Wang, Xijun Li, Jie Wang, Yufei Kuang, Mingxuan Yuan, Jia Zeng, Yongdong Zhang, and Feng Wu. 2023. Learning Cut Selection for Mixed-Integer Linear Programming via Hierarchical Sequence Model. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=Zob4P9bRNcK

[35] Yaoxin Wu, Wen Song, Zhiguang Cao, and Jie Zhang. 2021. Learning Large Neighborhood Search Policy for Integer Programming. In *NeuIPS'21*. 30075–30087.

[36] Jiajie Xu, Limin Guo, Zhiming Ding, Xiling Sun, and Chengfei Liu. 2012. Traffic aware route planning in dynamic road networks. In *Database Systems for Advanced Applications: 17th International Conference, DASFAA 2012, Busan, South Korea, April 15-19, 2012, Proceedings, Part I 17*. Springer, 576–591.

[37] Zhiyuan Xu, Jian Tang, Jingsong Meng, Weiyi Zhang, Yanzhi Wang, Chi Harold Liu, and Dejun Yang. 2018. Experience-driven Networking: A Deep Reinforcement Learning based Approach. In *INFOCOM'18*. 1871–1879.

[38] Harel Yedidsion, Roie Zivan, and Alessandro Farinelli. 2018. Applying max-sum to teams of mobile sensing agents. *Engineering Applications of Artificial Intelligence* 71 (2018), 87–99.

[39] Yifeng Zeng, Xuefeng Chen, Xin Cao, Shengchao Qin, Marc Cavazza, and Yanping Xiang. 2015. Optimal Route Search with the Coverage of Users' Preferences. In *IJCAI'15*. 2118–2124.

[40] Jiayi Zhang, Chang Liu, Xijun Li, Hui-Ling Zhen, Mingxuan Yuan, Yawen Li, and Junchi Yan. 2023. A survey for solving mixed integer programming via machine learning. *Neurocomputing* 519 (2023), 205–217.

[41] Zhen Zhang, Fan Wu, and Wee Sun Lee. 2020. Factor Graph Neural Networks. In *NeurIPS'20*.