A Graph Neural Network Approach for Solving the Ranked Assignment Problem in Multi-Object Tracking

Robin Dehler, Martin Herrmann, Jan Strohbeck, and Michael Buchholz

This paper has been accepted for presentation and publication at the 2024 IEEE Intelligent Vehicles Symposium (IV), June 02 - 05, 2024, Jeju Island, Republic of Korea. This is the accepted version of the paper, which has not been fully edited and the layout may differ from the original publication.

Citation information of the original publication:

R. Dehler, M. Herrmann, J. Strohbeck and M. Buchholz, "A Graph Neural Network Approach for Solving the Ranked Assignment Problem in Multi-Object Tracking," 2024 IEEE Intelligent Vehicles Symposium (IV), Jeju Island, Korea, Republic of, 2024, pp. 2646-2652, doi: 10.1109/IV55156.2024.10588674.

A Graph Neural Network Approach for Solving the Ranked Assignment Problem in Multi-Object Tracking

Robin Dehler[1], Martin Herrmann[1], Jan Strohbeck[1] and Michael Buchholz[1]

***Abstract*— Associating measurements with tracks is a crucial step in Multi-Object Tracking (MOT) to guarantee the safety of autonomous vehicles. To manage the exponentially growing number of track hypotheses, truncation becomes necessary. In the δ-Generalized Labeled Multi-Bernoulli (δ-GLMB) filter application, this truncation typically involves the ranked assignment problem, solved by Murty's algorithm or the Gibbs sampling approach, both with limitations in terms of complexity or accuracy, respectively. With the motivation to improve these limitations, this paper addresses the ranked assignment problem arising from data association tasks with an approach that employs Graph Neural Networks (GNNs). The proposed Ranked Assignment Prediction Graph Neural Network (RAPNet) uses bipartite graphs to model the problem, harnessing the computational capabilities of deep learning. The conclusive evaluation compares the RAPNet with Murty's algorithm and the Gibbs sampler, showing accuracy improvements compared to the Gibbs sampler.**

I. Introduction

Reliable multi-object tracking (MOT) is an important component for many technical applications, e.g., for robotics or autonomous driving. As part of environmental awareness modeling, MOT is faced with the task of tracking multiple objects over time given uncertainty in the detections [1], [2]. Different classical approaches to handle the tracking problem can be summarized into the categories Joint Probabilistic Data Association (JPDA), Multi-Hypothesis Tracking (MHT) [1] and approaches using Random Finite Sets (RFS) and the Finite Set Statistics (FISST) framework [2].

Within the RFS approach, a Bayes optimal MOT filter was introduced using Generalized Labeled Multi-Bernoulli (GLMB) RFSs [3]. For a real-time capable implementation, the GLMB filter was extended to the δ-Generalized Labeled Multi-Bernoulli (δ-GLMB) filter [4]. In the update step of the δ-GLMB filter, it is necessary to associate detections with objects or tracks [4]. When organizing both the detections and tracks in a cost matrix, the association task results in a ranked assignment problem that is solved optimally with Murty's algorithm [5]. Solving the ranked assignment problem poses high computational complexity. The association task is NP-hard when dealing with Multi-Sensor MOT (MS-MOT), where the association involves multiple tracks, measurements, and sensors [6]. Multi-sensor setups are crucial for creating a comprehensive environment model in autonomous driving, hence the importance of MS-MOT [7].

The δ-GLMB filter offers a mathematically optimal solution for MOT [4]. Because of the filter's complexity, however, the literature includes various mathematically suboptimal approaches, where the entire tracking task is executed using Deep Learning (DL) models, e.g., SORT and StrongSORT [8], [9], MOTS [10] or SMILEtrack [11].

Motivated by the promising results of DL approaches for tracking applications, this work aims to combine the advantages of the δ-GLMB filter with a GNN model that solves the computationally expensive data association task. Although being especially challenging for MS-MOT, this paper focuses on solving two-dimensional assignment problems, to form the basic theory for an entire class of DL methods for solving ranked assignment problems in single- and multi-sensor applications. Moreover, we propose a suitable DL model based on Graph Neural Networks (GNNs). For that, we represent the ranked assignment problem using bipartite graphs, where the source and target nodes align with the rows and columns of the cost matrix, respectively. The objective is then framed as a weighted bipartite matching problem [12].

Our main contributions are as follows:

- An introduction to the fundamental theory of describing assignment problems in MOT that can be used to solve the problem with a new class of DL-based algorithms.
- A specific GNN framework to predict ranked assignments denoted Ranked Assignment Prediction Graph Neural Network (RAPNet).
- A new score called *weighted position* (wp) for evaluating the performance of approximation algorithms for the ranked assignment problem focusing on the position of the assignments.

II. Background

For the considered MOT use case, the notion of RFSs is used to model the state of multiple objects and multiple measurements.

A. Multi-Object Generalized Labeled Multi-Bernoulli Filter

Taking the common notations from [3], a labeled state of an object is given by $\mathbf{x} = (x, l) \in \mathbb{X} \times \mathbb{L}$ with the state space $\mathbb{X}$ and label space $\mathbb{L}$ containing state vectors x and labels l, respectively. Then, the labeled states of all

Parts of this work have been financially supported by the Federal Ministry of Education and Research (project AUTOtech.*agil*, FKZ 01IS22088W). Parts of this research have been conducted as part of the PoDIUM project and other parts as part of the EVENTS project, which both are funded by the European Union under grant agreement No. 101069547 and No. 101069614 respectively. Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or European Commission. Neither the European Union nor the granting authority can be held responsible for them.

[1]The authors are with the Institute of Measurement, Control and Microtechnology, Ulm University, D-89081 Ulm, Germany.
E-mail addresses: {firstname}.{lastname}@uni-ulm.de

objects can be combined to the labeled multi-object state $\mathbf{X} = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(n)}\}$. Similarly, m measurements z form the multi-measurement state $Z = \{z^{(1)}, \dots z^{(m)}\}$. With $\mathcal{L}(\mathbf{X})$ as the set of labels of $\mathbf{X}$ and the distinct label operator $\Delta(\mathbf{X})$ that removes infeasible multi-object states and label sets, the multi-object probability density function of a GLMB RFS is defined by

$$\boldsymbol{\pi}(\mathbf{X}) = \Delta(\mathbf{X}) \sum_{c \in \mathbb{C}} w^{(c)}(\mathcal{L}(\mathbf{X})) \left[p^{(c)}\right]^{\mathbf{X}}, \tag{1}$$

where $c \in \mathbb{C}$ systematically lists the GLMB hypotheses consisting of the product of single-object densities $p^{(c)}(\mathbf{x})$ with corresponding weights $w^{(c)}(\mathcal{L}(\mathbf{X}))$. In (1), multiple copies of identical spatial distributions are considered. For a δ-GLMB, the probability density function in (1) is adjusted so that these duplicate copies are removed [3].[1]

Given a GLMB or δ-GLMB filtering density $\boldsymbol{\pi}_k(\mathbf{X}_k)$ at time step k, the GLMB prediction and update densities $\boldsymbol{\pi}_{k+1|k}(\mathbf{X}_{k+1}|Z_{1:k})$ and $\boldsymbol{\pi}_{k+1}(\mathbf{X}_{k+1}|Z_{1:k+1})$ can be calculated using the Chapman-Kolmogorov equation and Bayes rule. In the update step, each hypothesis creates a new set of hypotheses and the number of hypotheses grows intractably high. Thus, for a real-time capable implementation, it is necessary to truncate the number of resulting hypotheses at each time step. Using the ranked assignment problem, the weights corresponding to the hypotheses can be exploited to calculate the highest weighted hypotheses only, without the need to compute all possible new ones.

B. Ranked Assignment Problem

Every possible association of a set of measurements $Z = \{z_1, \dots, z_{|Z|}\}$ with a set of tracks $I = \{l_1, \dots, l_{|I|}\}$ has an individual cost value c_{ij} that depends on the multi-target model, e.g., Gaussian Mixture or sequential Monte Carlo approximation [4]. Furthermore, it is possible that a track l_i is misdetected with a corresponding cost value of c_i. All the cost values can be organized in a cost matrix C_Z with row i corresponding to track $i \in \{1, \dots, |I|\}$ and column j to either a measurement for $1 \leq j \leq |Z|$ or a misdetection for $|Z| + 1 \leq j \leq |Z| + |I|$:

$$C_Z = \begin{bmatrix} \overbrace{\begin{matrix} c_{11} & \dots & c_{1|Z|} \\ & \ddots & \\ \vdots & c_{ij} & \vdots \\ & & \ddots \\ c_{|I|1} & \dots & c_{|I||Z|} \end{matrix}}^{\text{detected}} & \overbrace{\begin{matrix} c_1 & \infty & \dots & & \infty \\ \infty & c_2 & \ddots & & \\ \vdots & \ddots & \ddots & & \vdots \\ & & & c_{|I|-1} & \infty \\ \infty & \dots & \infty & \infty & c_{|I|} \end{matrix}}^{\text{misdetected}} \end{bmatrix} \tag{2}$$

The entry c_{ij} in the *detected* part of (2) is the cost of associating track i with measurement j and the entry c_i in the *misdetected* part the cost value for misdetecting track i. The ∞-values of the *misdetected* part of the matrix imply invalid associations, i.e., only one cost value indicates the misdetection of track i. Similarly, the *detected* part of C_Z can also contain ∞-values that are created through a gating step to erase unlikely associations.

An assignment matrix S of the same shape $|I| \times (|Z| + |I|)$, with entries s_{ij} of either 0 or 1, models the association of tracks with measurements subject to

$$\textstyle\sum_{i=1}^{|I|} s_{ij} \leq 1, \qquad (j = 1, ..., |Z| + |I|) \tag{3a}$$

$$\textstyle\sum_{j=1}^{|Z|+|I|} s_{ij} = 1, \qquad (i = 1, ..., |I|) \tag{3b}$$

The conditions (3) define characteristics for S where each row has the sum 1 and each column the sum 0 or 1, ensuring that exactly one measurement or misdetection is associated with one track [4]. Given a cost matrix C_Z and an assignment matrix S, the overall cost of an assignment is given by the Frobenius inner product

$$Tr(S^T C_Z) = \sum_{i=1}^{|I|} \sum_{j=1}^{|Z|+|I|} c_{ij} s_{ij}. \tag{4}$$

Note that since $j \in \{1, \dots, |Z| + |I|\}$, the cost value c_{ij} in (4) can also correspond to an entry c_i in (2). The optimal assignment is the one that minimizes (4), which can be calculated, e.g., with the Hungarian method [13] or the Jonker-Volgenant algorithm [14]. For MOT, it is beneficial to calculate more than just the best assignment, which results in the ranked assignment problem to find a set of k minimal cost assignments with increasing order [5].

Equivalent to representing the costs in a cost matrix, the association of measurements with tracks can be modeled by a bipartite graph

$$\mathcal{G} = \{\mathcal{V}_s, \mathcal{V}_t, \mathcal{E}\}, \tag{5}$$

where the tracks are mapped to the source nodes $v_s \in \mathcal{V}_s$ and the measurements and misdetections to the target nodes $v_t \in \mathcal{V}_t$. The cost values c_{ij} and c_i for association of tracks with measurements and misdetections, respectively, are modeled by the edge attributes a_{ij} of the edges $e_{ij} \in \mathcal{E}$ that connect source nodes $v_s^{(i)}$ with target nodes $v_t^{(j)}$. The costs c_i are matched with edges $e_{i,i+|Z|}$. The ∞-values are modeled by excluding edges for the corresponding pairs of v_s and v_t. Similar to the assignment matrix S, an association map can be represented by classifying the edges as either 0 or 1 with the following constraints: (i) From the set of edges originating from a source node, exactly one edge is classified as 1 and the rest as 0. (ii) From the set of edges connected to a target node, at most one edge is classified as 1. When represented as a bipartite graph, the ranked assignment problem is a weighted bipartite matching problem [12].

In [4], the ranked assignment problem for the truncation of δ-GLMB hypotheses is solved using Murty's algorithm [5]. A more efficient algorithm is proposed in [15], where the optimal solution of the ranked assignment problem is approximated using a method based on Gibbs sampling. The Gibbs sampling approach, however, has the drawback that it is less accurate. The architecture proposed in this paper also approximates the optimal solution of the ranked assignment

[1] Since the following considerations are similar for both GLMB and δ-GLMB RFSs, the detailed replacements and resulting equations for the latter are excluded.

problem by using the bipartite graph formulation of the problem in order to use GNNs to predict the assignments.

C. Graph Neural Networks

GNNs are a powerful tool to solve tasks on data that is represented with graphs [16], e.g., graph or node classification and link prediction. Using the notion of message passing, GNN layers aggregate information about the neighborhood of nodes using an activation function that combines the node feature information with the edge features [17]. The Graph Attention Network (GAT), e.g., updates each node feature similar to the attention mechanism [18], where an individual importance score is calculated for each neighboring node.

D. Related Work

In literature, there exist some approaches for solving assignment problems with deep learning methods. In [19], a method is proposed where the assignment problem is decomposed into sub-assignment problems that are independently solved with deep neural networks. In [20], a GNN architecture is developed for solving the assignment problem. However, for the GNN framework, quadratic cost matrices are required, which limits the flexibility of the matrices that can be used. In [21], Liu *et al.* introduce GLAN, which is another approach based on GNNs that is able to handle graphs of different sizes. The GLAN network is also evaluated in an MOT scenario, where the network combined with the Faster R-CNN object detector achieves high MOTA and MOTP scores. Nonetheless, all of the mentioned systems are only trained to obtain the best assignment, not to solve the ranked assignment problem. Solving the ranked assignment problem with DL methods has not yet been researched.

III. Ranked Assignment Prediction

The proposed framework to predict the ranked assignments consists of a graph creation module, the Ranked Assignment Prediction Graph Neural Network (RAPNet) and a post-processing stage to extract assignments from the RAPNet output. The three modules are explained in the following.

A. Graph Creation

Like the known methods, our approach uses the cost matrix representation as input data. The first step to use a GNN for the prediction is to transform the cost matrices to bipartite graphs. The transformation includes source and target node creation with corresponding node features x_s and x_t, respectively, and the creation of edge indices e_{ij} that indicate the connection of source nodes i with target nodes j with edge attributes a_{ij}. The edge attributes can directly be taken from the values of the cost matrix. However, because of the dynamic number of tracks and measurements, the cost values of the rows and columns can not directly be taken for the node features, since the dimensions of the features need to be consistent. Thus, for each row and column, the following five features were identified to provide a good representation of the input data: (i) The ratio of non-∞ values to the length of the row or column, respectively, and the values (ii) *min*, (iii) *max*, (iv) *mean* and (v) *l2-norm* are aggregated without consideration of the ∞ entries. The number of source nodes ν_s and target nodes ν_t correlate with the number of rows and columns of the input matrices, respectively. In the considered use case, the input matrices are of the form of C_Z from Eq. (2). From this, $\nu_s = |I|$ and $\nu_t = |I| + |Z|$ follow.

B. Ranked Assignment Prediction Graph Neural Network

The RAPNet takes the created bipartite graphs as inputs. The task of predicting assignments is modeled as a binary classification task on the graph edges. The most effective network architecture, as determined through comprehensive research, is depicted in Fig. 1, following an encoder-decoder structure. The encoder processes both the node and edge features through GNN and Linear layers, respectively. The encoded features are then combined through element-wise multiplication and split into multiple assignment predictions in the decoder. For a better training, the inputs, i.e., the node and edge features, are normalized to the range $[0, 1]$.

Encoder: As shown in Fig. 1, the node and edge features of the input graphs are first updated using Graph Convolutional Network (GCN) layers [22] and Fully Connected (Linear) layers, respectively, with leaky ReLU activation functions (σ_{lr}) in between. After the GCN layers, the updated features are combined and fed through Graph Attention Network (GAT) layers [18], again using leaky ReLU activations. The last GAT layers output encoded features x_s and x_t. Through indexing with the corresponding edge indices, x_s and x_t are multiplied element-wise to combine the features to x_{mul}. Both GCN and GAT layers only update the target node features of bipartite graphs. For updating the source nodes, the source and target nodes are mirrored for each layer. Thus, each of the shown GCN and GAT layer blocks consist of two layers to update both x_s and x_t, respectively.

Dependent on the MOT application, a different number of assignments can potentially be required for each cost matrix and consequently for each graph. To be able to use batched graphs and also to better train the network, a fixed number of output assignments $k_{\text{max}} = 10$ was chosen. This has the disadvantage that it is not possible to optimize the network for tasks with $k > k_{\text{max}}$, but has the advantages that each of the k_{max} solutions are better optimized for predicting one particular solution and that for $k \leq k_{\text{max}}$, the network predicts more than the necessary assignments which increases the possibility of predicting the optimal ones. Thus, the GAT layers and the last GCN and Linear layer in the encoder have an output dimension that is the product of the number of channels for the encoder layers C_{enc} and k_{max}.

Decoder: The decoder architecture is depicted in the right part of Fig. 1. After reshaping, the encoded node features x_{mul} and edge features e_{ftr} are fed into separate Long Short-Term Memory (LSTM) Recurrent Neural Networks (RNN) to create k_{max} different features. Then, the LSTM outputs x_{lstm} and e_{lstm} are multiplied element-wise. After a leaky ReLU activation, two separate groups of linear layers are applied to each of the k_{max} solutions. In between the two

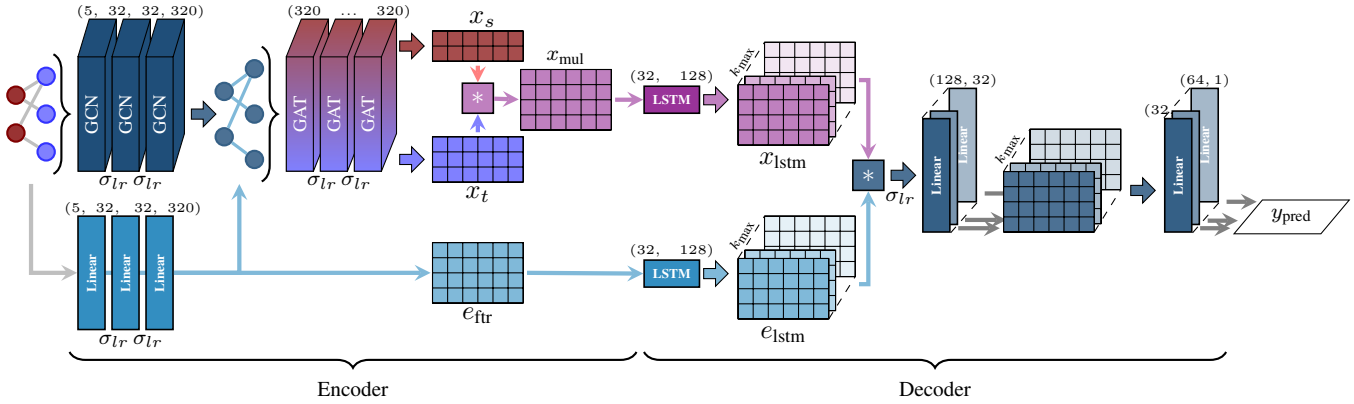


Fig. 1. Architecture of the RAPNet. Left part: Encoding of the node and edge features. Before the multiplication step of x_s and x_t, the feature vectors are indexed with the respective edge index vectors to fit the dimensions. Right part: Decoding using LSTM layers and creation of predicted assignments. The channel sizes $C_{\text{enc}} = 32$, $C_{\text{lstm}} = 128$ and $C_{\text{dec}} = 32$ are shown above each neural network layer. The final output y_{pred} is of shape $|\mathcal{E}| \times k_{\text{max}}$.

groups, the output of the first group for layers $i \in [2, k_{\text{max}}]$ is concatenated with the output of layer $i-1$ before the second group to create an additional dependency between the assignments. Thus, the second up to the last linear layer in the second group receive two times the number of input features compared to the first layer, as indicated in Fig. 1. The resulting output y_{pred} of the RAPNet is of shape $|\mathcal{E}| \times k_{\text{max}}$. Each of the k_{max} columns of y_{pred} represents an individual predicted assignment matrix.

Loss function: Since the problem is a binary classification task, a Sigmoid activation is employed on the output of the neural network y_{pred}. The targets of the neural network y represent the optimal assignments calculated with Murty's algorithm. Then, a (weighted) Binary Cross-Entropy (BCE) function calculates the loss between the network output y_{pred} and the optimal solutions y. The weights for the two classes 0 and 1 are determined based on their ratio to account for class imbalance due to sparsity.

C. Post-processing Module

The output y_{pred} of the RAPNet can directly be converted to k_{max} predicted assignment matrices $A^{(i)}_{\text{pred}}$, $i \in [1, k_{\text{max}}]$. From each assignment matrix $A^{(i)}_{\text{pred}}$, the predicted assignment $a^{(i)}_{\text{pred}}$ is extracted by taking the maximum value of each row. As a consequence of inaccurate predictions, it is possible that invalid assignments are created, i.e., that a column is assigned twice. Thus, a greedy strategy expands the number of chosen assignments. The greedy strategy exploits the imperfect output $A^{(i)}_{\text{pred}}$ by taking more than only the maximum value of each row. It additionally uses other entries where the predicted values are greater than a threshold θ_{sig} to increase the likelihood of getting valid assignments. Since the sigmoid activation function creates values between 0 and 1, we set $\theta_{\text{sig}} = 0.5$.

Algorithm 1 summarizes the post-processing procedure. The maximum value of each row is taken and set to a_{found} (code line 3). Then, the number of entries that are greater than θ_{sig} is determined for each row and stored in a_{sum} (line 4). As long as there are rows with multiple assignments, which is equivalent to a_{sum} containing entries ≥ 2, the one with the most entries $\geq \theta_{\text{sig}}$ is taken and the highest entry is set to 0. Then, a new possible assignment a_{add} is taken from the maximum entries of the modified predicted assignment matrix and added to $a^{(i)}_{\text{found}}$. This is repeated until all redundant predictions in the rows of $A^{(i)}_{\text{pred}}$ are added to a_{found} (lines 5-9). Finally, the invalid solutions in a_{found} are removed (line 10) and all valid solutions from each $A^{(i)}_{\text{pred}}$ are concatenated and returned (lines 11-13). The steps in the for-loop can be done independently for all $A^{(i)}_{\text{pred}}$. Thus, the for-loop is parallelized to reduce the computational overhead.

Algorithm 1 Greedy Post Processing

Input: y_{pred}, k_{max}, **Output:** Assignments a_{valid}

1: Get $A^{(i)}_{\text{pred}}$, $i \in [1, k_{\text{max}}]$ from y_{pred}
2: **for** Each $A^{(i)}_{\text{pred}}$ **do**
3: $\quad a^{(i)}_{\text{pred}} = \text{argmax}_{\text{rows}} A^{(i)}_{\text{pred}}$, $a_{\text{found}} \leftarrow \{a^{(i)}_{\text{pred}}\}$
4: $\quad A_{\text{round}} \leftarrow \text{Round } A^{(i)}_{\text{pred}}$, $a_{\text{sum}} \leftarrow \sum_{\text{rows}} A_{\text{round}}$
5: $\quad$ **while** Entries ≥ 2 in a_{sum} **do**
6: $\quad\quad$ Row $j \leftarrow \text{argmax}(\sum_{\text{rows}} A_{\text{round}})$
7: $\quad\quad$ Set max value of row j in $A^{(i)}_{\text{pred}}$ to 0
8: $\quad\quad a_{\text{add}} = \text{argmax}_{\text{rows}} A^{(i)}_{\text{pred}}$, $a_{\text{found}} = a_{\text{found}} \cup \{a_{\text{add}}\}$
9: $\quad\quad$ Calculate new $A_{\text{round}}, a_{\text{sum}}$ from updated $A^{(i)}_{\text{pred}}$
10: $\quad a^{(i)}_{\text{valid}} \leftarrow$ Valid solutions of a_{found}
11: $a_{\text{valid}} \leftarrow$ Concatenate all $a^{(i)}_{\text{valid}}$
12: Keep at most k_{max} solutions with lowest cost
13: **Return** a_{valid}

IV. Experiments

This section summarizes the parameters of the RAPNet and compares its performance to the optimal solution of Murty's algorithm and the solutions calculated with the Gibbs sampling method.

A. Training Setup

The RAPNet is initialized with 32 encoder channels, 128 LSTM channels and 32 decoder channels (see also Fig. 1). The training involved 20 epochs with an AdamW optimizer

[23] with weight decay $\lambda = 0.001$ and the cosine annealing learning rate scheduler [24] with initial learning rate of $\gamma = 0.001$ and minimum value of $\gamma_{\text{min}} = 0.0001$. The data used for the training included both synthetic graphs as well as graphs that were extracted from a simulated MOT scenario that was introduced in [25] for the evaluation of the PM-δ-GLMB filter. Since the primary focus is solving the ranked assignment problem for the δ-GLMB update step, the synthetically created graphs are designed to be similar to the ones resulting from the cost matrices of the MOT simulation. Thus, the matrices and equivalently the graphs are also designed to have a detected and misdetected part. The entries were created using a Gaussian Mixture model with two components with mean $[-2.5, 0.5]$, covariance $[0.5, 1.5]$, and equal weights, respectively. These values were chosen to be alike to the values of the simulation data. Lastly, possible ∞-values of the detected part were also considered by randomly adding ∞-values with a probability threshold $\vartheta \in [0, 1]$. A threshold of 0 means that none of the values are ∞, a threshold of 1 means that all values are set to ∞.

Several parameters of the synthetic graphs can be modified, i.e., the threshold ϑ, the value k, and the row number ν_s. The column number ν_t of the detected part of the cost matrix is chosen to be in the range $[\min(1, \nu_s - 1), \nu_s + 4]$. For the training dataset, the value of k is sampled from a Poisson distribution with mean 4. Note that the mean value of k of the simulated data is 3.5692. The threshold ϑ is increased from 0.1 to 0.9 in steps of 0.1 and ν_s from 1 to 15.

B. Evaluation Setup

The first part of the evaluation compares the performance of the RAPNet alone (RAPNet-a) and the RAPNet with the post-processing stage (RAPNet-PP) to the Gibbs sampler and Murty's algorithm using two different parameter sweeps on synthetically created data with changing parameters ν_s and the maximum number of k. Second, the performance on a validation set including only the simulated data is compared. Lastly, the needed computational times of the three methods to create assignments are compared.

Three different scores are used for the comparison. First, the accuracy of each single assignment w.r.t. the optimal assignments is calculated by comparing each of the k optimal assignments to the predicted ones. Second, the overall costs of the chosen assignments are used. If less than k assignments were found, the missing assignments are penalized with $c_{\text{max}} + 0.1$, with c_{max} being the cost of the worst assignment. The first assignments are particularly important for the MOT application because they contain the most likely associations. Thus, a new score called *weighted position* (*wp*) is proposed, which evaluates the position of the predicted assignments compared to the optimal assignments:

$$wp = \sum_{i=1}^{k} w_i \kappa_i \tag{6}$$

In (6), κ_i rates the position of the predicted assignment $a_{\text{pred}}^{(i)}$ to the optimal one $a^{(i)}$ with

$$\kappa_i = \begin{cases} 3, \text{ if } a_{\text{pred}}^{(i)} = a^{(i)} \\ 2, \text{ if } a_{\text{pred}}^{(i)} \in \left\{a^{(i-\rho)}, \dots, a^{(i-1)}, a^{(i+1)}, \dots, a^{(i+\rho)}\right\} \\ 1, \text{ if } a_{\text{pred}}^{(i)} \in \left\{a^{(j)} \setminus \{a^{(i-\rho)}, \dots, a^{(i+\rho)}\}, j \in [1, k]\right\} \\ 0, \text{ otherwise,} \end{cases}$$

where ρ is adjustable and in our case set to 2. The position is weighted with the weight value

$$w_i = \frac{2 \cdot (k + 1 - i)}{k \cdot (k + 1)} \tag{7}$$

to emphasize the first assignments. The score is able to better validate whether generally good assignments were found compared to the accuracy, even if not all optimal ones are generated. The highest value for wp is 3 that results from the assignments of Murty's algorithm.

C. Performance Evaluation

The plots of the described sweeps are shown in Fig. 2.

Note that the accuracy score is only computable for assignment i if $k \geq i$. Also, the accuracy plots of Murty's algorithm and the RAPNet-a are excluded for readability reasons.

The plots on the left show the influence of the maximum number of k, i.e., k_{max}. On the left side, the accuracy plot demonstrates that the accuracies of the first 4 assignments are practically independent of k_{max}. This applies to both the Gibbs sampler and the RAPNet-PP, with the RAPNet-PP being generally more accurate than the Gibbs sampler except for $k = 1$, where the Gibbs sampler always finds the optimal solution that is used as the initial assignment. Concerning the *wp* scores on the left, both the RAPNet-PP and the RAPNet-a outperform the Gibbs sampler. The decreasing *wp* score for higher values of k_{max} for both the RAPNet and the Gibbs sampler can be explained by the fact that the weight value is smaller for the first assignments with a higher k_{max}, but the accuracy of both methods decreases for assignments of higher k. The average costs of the k assignments can be seen in the plot at the bottom left. Both the *wp* score and the cost plot show that the developed post-processing algorithm is able to improve the predictions of the RAPNet-a. The costs are often negative, which follows from the derived Gaussian Mixture model. For lower k_{max}, the two RAPNet versions perform better than the Gibbs sampler. The Gibbs sampler surpasses the RAPNet-PP for $k_{\text{max}} > 11$ and the RAPNet-a for $k_{\text{max}} > 5$. One reason why the RAPNet is worse for higher values of k_{max} is obviously the design choice of using a fixed number of assignments to be predicted. Thus, the cost of the RAPNet-a has to increase for $k > k_{\text{max}}$, while the RAPNet-PP potentially creates more than just $k = 10$ solutions. In combination with the other plots, it can be concluded that the Gibbs sampler finds more solutions for higher k which, however, are usually not the optimal solutions.

All three of the plots of the size sweep of Fig. 2 on the right show a performance decrease of both the RAPNet and

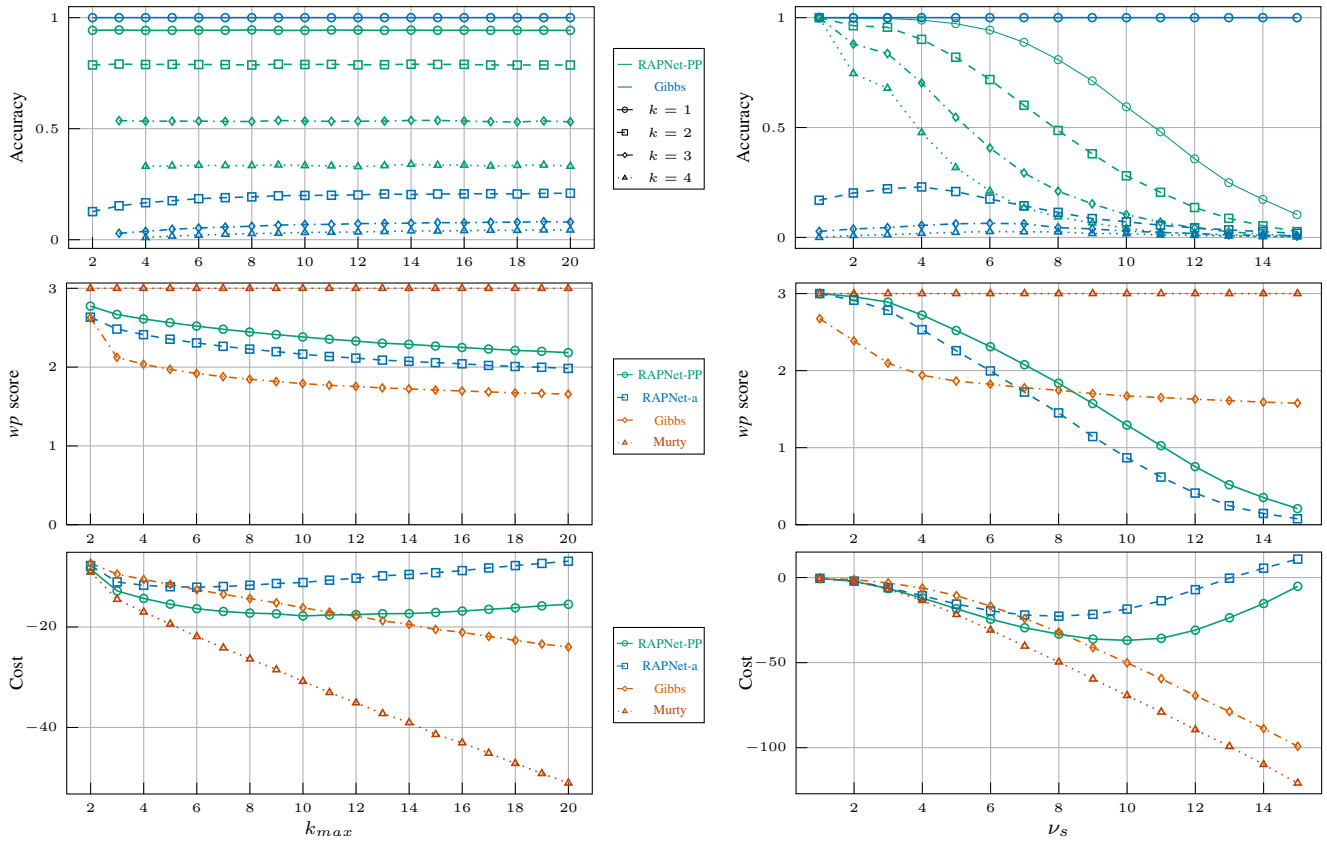


Fig. 2. Plots of the two sweeps of the parameters $k_{\max}$ and ν_s evaluated with the scores accuracy, *wp* and cost. For readability reasons in the accuracy plots, only the accuracies of the first 4 solutions for the RAPNet with the post-processing and the Gibbs sampler are shown. The legends on the right are also valid for the left plots. In the legends, RAPNet-PP and RAPNet-a indicate the versions with and without post-processing, respectively.

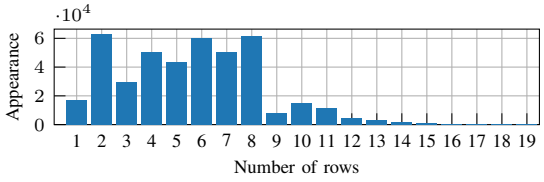


Fig. 3. Distribution of matrix sizes in the simulation data.

the Gibbs sampler w.r.t. bigger matrices. Compared with the Gibbs sampler, both RAPNet versions have better *wp* scores and lower cost values for $\nu_s < 9$ or $\nu_s < 8$, respectively. The Gibbs sampler outperforms the RAPNet versions for bigger matrices. This can be explained with the fast decline of the accuracy curves of the RAPNet seen in the accuracy plot on the right, while the Gibbs sampler especially profits from using the optimal solution for $k = 1$. However, for the considered MOT scenario, matrices with more than 8 rows are very rare. This can clearly be seen in Fig. 3, where the matrix appearance frequency in relation to the number of rows of the simulation dataset is plotted.

Thus, a comparison of the framework's performance to the Gibbs sampler on a validation set that only includes simulation data is presented in Table I. While the Gibbs sampler generates the optimal solution for the accuracy of the initial assignment, both RAPNet versions perform better in all other scores. The post-processing improves the already good performance of the RAPNet itself. With Murty's algorithm, the optimal accuracy scores, *wp* score and cost value are 1.0, 3.0 and -5.92, respectively.

TABLE I

COMPARISON OF THE RAPNET VERSIONS TO THE GIBBS SAMPLER ON SIMULATION DATA.

Framework	Accuracies $i = 1$	$i = 2$	$i = 3$	$i = 4$	*wp*	Cost
RAPNet-a	0.99	0.91	0.73	0.54	2.74	5.68
RAPNet-PP	0.99	**0.95**	**0.82**	**0.66**	**2.80**	**3.23**
Gibbs	**1**	0.18	0.06	0.04	2.11	14.10

D. Computational Complexity

Lastly, the needed computational time of the parts of the framework, the Gibbs sampler and Murty's algorithm w.r.t. the matrix sizes is summarized in Table II. For the RAPNet and the post-processing, the computational time is taken for non-batched and batched data with a batch size of 32, i.e., RAPNet_{32} and PP_{32}. The experiments were done on an AMD Ryzen 9 7950x CPU for Murty's algorithm and the Gibbs sampler and an NVIDIA RTX 4080 GPU for the parts of the RAPNet framework. The comparison shows that, without batching, the RAPNet's needed computational time is generally higher compared to the Gibbs sampler and Murty's algorithm. For the batched version, except for very small matrices with $\nu_s < 3$, the computational time of RAPNet_{32} for each single graph in the batch is similar to that of the Gibbs sampler. The table also shows the increased slowdown of Murty's algorithm for bigger matrices. For

TABLE II

TIME CONSUMPTION OF THE DIFFERENT MODULES, IN MS.

Rows ν_s	Graph Creation	RAP-Net$_{32}$	RAP-Net	PP$_{32}$	PP	Gibbs	Murty
1	0.29	0.12	3.04	0.32	0.53	0.05	0.03
3	0.30	0.15	3.03	0.39	0.59	0.12	0.14
5	0.30	0.19	3.06	0.45	0.64	0.19	0.28
7	0.31	0.25	3.11	0.47	0.68	0.26	0.43
9	0.31	0.32	3.15	0.48	0.70	0.32	0.61
11	0.31	0.39	3.15	0.48	0.71	0.38	0.83
13	0.30	0.48.11	0.48	0.69	0.43	1.06	
15	0.33	0.58	3.22	0.48	0.69	0.50	1.38

the MOT application, the overhead of the graph creation step can be reduced by directly creating graphs instead of cost matrices. For the unbatched version, the post-processing function adds a relatively high overhead that is less for the batched version. We argue that the worse computational overhead for unbatched graphs results from the complexity of the GNN layers. This fact is supported by an investigation into the ratio of the needed time of the RAPNet's encoder to the decoder, which shows that the encoder takes about $4/5$th of the RAPNet's computational time. However, the practically constant time of the RAPNet for unbatched graphs shows an advantage compared to the other methods, where the needed time increases for bigger graphs. It is part of our future work to resolve the current limitations concerning the computational complexity of the framework.

V. CONCLUSION AND FUTURE WORK

This paper presented a novel approach utilizing a Graph Neural Network for solving the ranked assignment problem, with a particular focus on assignment problems within the update step of the δ-GLMB filter. The proposed RAPNet was compared to the Gibbs sampler and Murty's algorithm on both data from a simulated MOT scenario and synthetic data, showing the capabilities of the RAPNet itself and in combination with the post-processing stage, but also some of the current restrictions.

Our work can be the foundation of further research in this field. In our own future work, we want to reduce the computational complexity and integrate our approach into our tracking framework. As motivated in the introduction, we will also extend the framework to handle assignment problems that result from MS-MOT scenarios. For MS-MOT scenarios, it is also very interesting how the computational complexities of the different algorithms scale, especially due to the NP-hard complexity of getting an optimal solution. An investigation into using not only the cost values as inputs to the network, but also explicit track representations, could also be worthwhile. Here, due to its flexibility, our GNN-based approach could make use of the additional features that are not used by the Gibbs sampler.

REFERENCES

[1] Y. Bar-Shalom, P. Willett, and X. Tian, *Tracking and Data Fusion: A Handbook of Algorithms*. YBS Publishing, 2011.

[2] R. P. S. Mahler, *Statistical Multisource-Multitarget Information Fusion*. USA: Artech House, Inc., 2007.

[3] B.-T. Vo and B.-N. Vo, "Labeled Random Finite Sets and Multi-Object Conjugate Priors," *IEEE Transactions on Signal Processing*, vol. 61, no. 13, pp. 3460–3475, 2013.

[4] B.-N. Vo, B.-T. Vo, and D. Phung, "Labeled Random Finite Sets and the Bayes Multi-Target Tracking Filter," *IEEE Transactions on Signal Processing*, vol. 62, no. 24, pp. 6554–6567, 2014.

[5] K. G. Murty, "Letter to the Editor - An Algorithm for Ranking all the Assignments in Order of Increasing Cost," *Operations Research*, vol. 16, no. 3, pp. 682–687, 1968.

[6] B.-N. Vo, B.-T. Vo, and M. Beard, "Multi-Sensor Multi-Object Tracking With the Generalized Labeled Multi-Bernoulli Filter," *IEEE Transactions on Signal Processing*, vol. 67, no. 23, pp. 5952–5967, 2019.

[7] M. Buchholz, J. Müller, M. Herrmann, J. Strohbeck, B. Völz, M. Maier, J. Paczia, O. Stein, H. Rehborn, and R.-W. Henn, "Handling occlusions in automated driving using a multiaccess edge computing server-based environment model from infrastructure sensors," *IEEE Intelligent Transportation Systems Magazine*, vol. 14, no. 3, pp. 106–120, 2022.

[8] N. Wojke, A. Bewley, and D. Paulus, "Simple Online and Realtime Tracking with a Deep Association Metric," in *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 3645–3649.

[9] Y. Du, Z. Zhao, Y. Song, Y. Zhao, F. Su, T. Gong, and H. Meng, "StrongSORT: Make DeepSORT great again," *IEEE Transactions on Multimedia*, 2023.

[10] P. Voigtlaender, M. Krause, A. Osep, J. Luiten, B. B. G. Sekar, A. Geiger, and B. Leibe, "MOTS: Multi-Object Tracking and Segmentation," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 7934–7943.

[11] Y.-H. Wang, J.-W. Hsieh, P.-Y. Chen, M.-C. Chang, H.-H. So, and X. Li, "SMILEtrack: SiMIlarity LEarning for Occlusion-Aware Multiple Object Tracking," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 6, pp. 5740–5748, Mar. 2024.

[12] R. Burkard, M. Dell'Amico, and S. Martello, *Assignment Problems*. Society for Industrial and Applied Mathematics, 2012.

[13] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97.

[14] R. Jonker and T. Volgenant, "A shortest augmenting path algorithm for dense and sparse linear assignment problems," *Computing*, vol. 38, pp. 325–340, 1987.

[15] B.-N. Vo, B.-T. Vo, and H. G. Hoang, "An Efficient Implementation of the Generalized Labeled Multi-Bernoulli Filter," *IEEE Transactions on Signal Processing*, vol. 65, no. 8, pp. 1975–1987, 2017.

[16] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The Graph Neural Network Model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[17] Y. Ma and J. Tang, *Deep Learning on Graphs*. Cambridge University Press, 2021.

[18] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=rJXMpikCZ

[19] M. Lee, Y. Xiong, G. Yu, and G. Y. Li, "Deep Neural Networks for Linear Sum Assignment Problems," *IEEE Wireless Communications Letters*, vol. 7, no. 6, pp. 962–965, 2018.

[20] C. Aironi, S. Cornell, and S. Squartini, "Tackling the Linear Sum Assignment Problem with Graph Neural Networks," in *Applied Intelligence and Informatics*. Cham: Springer Nature Switzerland, 2022, pp. 90–101.

[21] H. Liu, T. Wang, C. Lang, S. Feng, Y. Jin, and Y. Li, "GLAN: A Graph-based Linear Assignment Network," *CoRR*, vol. abs/2201.02057, 2022.

[22] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in *International Conference on Learning Representations*, 2017. [Online]. Available: https://openreview.net/forum?id=SJU4ayYgl

[23] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=Bkg6RiCqY7

[24] ——, "SGDR: Stochastic Gradient Descent with Warm Restarts," in *International Conference on Learning Representations*, 2017. [Online]. Available: https://openreview.net/forum?id=Skq89Scxx

[25] M. Herrmann, C. Hermann, and M. Buchholz, "Distributed Implementation of the Centralized Generalized Labeled Multi-Bernoulli Filter," *IEEE Transactions on Signal Processing*, vol. 69, pp. 5159–5174, 2021.