

A Graph-Based Neural Approach to Linear Sum Assignment Problems

Carlo Aironi ^{*}, Samuele Cornell [†] and Stefano Squartini [‡]

*Department of Information Engineering
Università Politecnica delle Marche, Italy*

^{*}*c.aironi@pm.univpm.it*

[†]*s.cornell@pm.univpm.it*

[‡]*s.squartini@univpm.it*

Received 12 May 2023

Accepted 1 December 2023

Published Online 17 January 2024

Linear assignment problems are well-known combinatorial optimization problems involving domains such as logistics, robotics and telecommunications. In general, obtaining an optimal solution to such problems is computationally infeasible even in small settings, so heuristic algorithms are often used to find near-optimal solutions. In order to attain the right assignment permutation, this study investigates a general-purpose learning strategy that uses a bipartite graph to describe the problem structure and a message passing Graph Neural Network (GNN) model to learn the correct mapping. Comparing the proposed structure with two existing DNN solutions, simulation results show that the proposed approach significantly improves classification accuracy, proving to be very efficient in terms of processing time and memory requirements, due to its inherent parameter sharing capability. Among the many practical uses that require solving allocation problems in everyday scenarios, we decided to apply the proposed approach to address the scheduling of electric smart meters access within an electricity distribution smart grid infrastructure, since near-real-time energy monitoring is a key element of the green transition that has become increasingly important in recent times. The results obtained show that the proposed graph-based solver, although sub-optimal, exhibits the highest scalability, compared with other state-of-the-art heuristic approaches. To foster the reproducibility of the results, we made the code available at https://github.com/aircarlo/GNN_LSAP.

Keywords: Linear sum assignment; graph neural networks; deep neural networks; smart meters scheduling; smart grid optimization.

1. Introduction

Linear assignment¹ is a fundamental problem in operations research; it aims at assigning the elements of one finite set to the elements of another set. This is done under the condition of one-to-one correspondence, so that the resulting assignment satisfies some optimality criterion, such as minimum cost or, in a dual form, maximum profit. When the sum

of the costs is the objective to be minimized, the problem is called a Linear Sum Assignment Problem (LSAP).

This type of problem is found in many image processing applications such as point matching,² handwritten character and mathematical expression recognition,³ Multiple Object Tracking (MOT),⁴ and object segmentation.⁵ In wireless communication

[‡]Corresponding author.

This is an Open Access article published by World Scientific Publishing Company. It is distributed under the terms of the Creative Commons Attribution 4.0 (CC BY) License which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

systems, it plays an important role in tasks such as mode selection for device-to-device communications,⁶ resource allocation in MIMO systems⁷ and unlicensed channel management for LTE systems.⁸ On audio processing field, the permutation ambiguity problem of multiple source separation⁹ is closely related to LSAP, as well as to end-to-end neural diarization¹⁰ and, in general, for metrics computation, such as diarization error rate¹¹ or permutation-invariant word error rate¹² for automatic speech recognition.

1.1. Related works

A well-established method for linear assignments is the Hungarian algorithm,¹³ developed by Kuhn in 1955 and revised by Munkres in 1957,¹⁴ which succeeds in obtaining the optimal solution without a greedy search. However, it does not scale well with the size of the problem N , since its computational complexity is $\mathcal{O}(N^3)$.

Bertsekas *et al.* proposed the auction algorithm¹⁵ to solve LSAP problems, so called because it mimics the actual auction process. This method comprises two steps: A bidding phase and an assignment phase. Later, an extension based on a scaling strategy further enhanced the performance of the auction algorithm. While being close to the ideal solution, it still has a high computational complexity, which affects its outcomes for linear assignment issues.

A variant of the Hungarian algorithm that uses the shortest alternating paths to supplement primal solutions has been proposed by Jonker and Volgenant.¹⁶ It begins with an initialization phase based on a naive auction algorithm.

Several algorithms, including the interior point method^{17,18} and dual forest method,¹⁹ solve linear assignment problems using general linear programming techniques. Ramakrishnan *et al.*¹⁸ modified the Karmarkar interior-point method²⁰ and created an approximate dual projective algorithm.

Additionally, a lot of heuristic methods based on greedy tactics try to find quick approximate solutions. For the generalized assignment problems, Trick *et al.* suggest a greedy heuristic technique²¹ and demonstrate that adding some randomization to the greedy approach can help find better solutions. Another similar approach is the Greedy Randomized

Adaptive Search Procedure (GRASP).²² Naiem *et al.* develop the Deep Greedy Switching (DGS) algorithm,^{23,24} which starts with a random initial guess and attempts to discover a better solution by searching inside a well-defined neighborhood. Although these methods are much faster than the previous heuristics having polynomial complexity, they often get stuck when the value of the objective function reaches a local optimum. Moreover, because the gradients of these heuristic solutions are somewhat difficult to describe, they cannot be directly included in learning frameworks.

Recently, deep neural networks (DNNs) have achieved promising results on mathematical optimization problems, with practical applications such as wireless resource management allocation,²⁵ link scheduling optimization²⁶ or interference management.²⁷ Several data-driven algorithms have also been proposed for linear assignment problems. In Ref. 28, the assignment task was converted into an equivalent continuous linear programming problem solved by a recurrent neural network. Recent works address smaller sub-tasks by breaking the $N \times N$ assignment problem into N multi classification tasks, using stacked perceptrons layers,²⁹ bidirectional long short-term memory neural network (BDLSTM)³⁰ and bidirectional recurrent neural networks (BiRNN).³¹

This paper is an extension of the work presented in Ref. 32, where we propose a graph-based LSAP description and a DNN technique built on graph learning is used to address the assignment challenge. This extended version of the paper includes a case study of a real application on smart meter scheduling and an in-depth analysis of GNN network performance as the number of hidden layers and batch-size used in training vary. As a comparison, we build upon the framework proposed by Lee *et al.*,²⁹ where LSAPs of different dimensionality N are decomposed into N independent sub-assignment problems, and two types of DNNs, a feed-forward MultiLayer Perceptron (MLP) and a Convolutional Neural Network (CNN) are applied to address the sub-assignments as independent classification tasks.

In this work, we show that using an MLP is the worst approach because it forces the problem to be treated as independent assignments. A better modeling strategy is arguably to process the entire cost matrix as input. However, for example, in the

CNN approach, as convolutional kernels cover a narrow receptive field, it is required to stack several layers to cover the entire cost matrix as the size of the problem increases. Moreover, CNNs have finite receptive field, thus this solution cannot scale to cover arbitrarily large cost matrices. In contrast, the proposed graph representation of the cost matrix together with graph-based DNN techniques allows for an efficient information spread, exploiting relations between all agent-job pairs, even for networks with limited depth, with obvious advantages in terms of scalability.

The paper is organized as follows: in Sec. 2 the Assignment problem is formalized and the steps of the Munkres–Hungarian algorithm are summarized; Sec. 3 provides an overview on Graph Neural Networks (GNN) and their application to combinatorial optimization problems; in Sec. 4 the proposed approach, the loss function, the evaluation metric and the neural architectures are presented in detail, and the experimental results are discussed in Sec. 5. Finally, in Sec. 6 we describe the real-world scenario of smart meter reading scheduling, in smart grids, explaining how the proposed solution for solving the allocation problem, represents an interesting trade-off between optimality and efficiency.

2. Problem Formulation

Typically, in the literature, assignment problems are described by resorting to a toy-example in which N jobs must be assigned to as many different workers or agents; this of course can be extended to any other context in which two sets of the same cardinality are involved and all elements must be paired one by one.

Given two finite sets $I = \{1, 2, \dots, N\}$ and $J = \{1, 2, \dots, N\}$, let us assume that the assignment of element $i \in I$ to element $j \in J$ incurs a cost $c_{i,j}$. The problem has a straightforward integer programming formulation, in which the decision variable $x_{i,j}$ is equal to 1 or 0 to indicate a feasible or infeasible assignment, respectively; therefore, it can be expressed as the minimization of the following objective function:

$$\sum_{i=1}^N \sum_{j=1}^N c_{i,j} x_{i,j}, \quad (1)$$

subject to the constraints:

$$\sum_{i=1}^N x_{i,j} = 1 \quad j = 1, \dots, N, \quad (2)$$

$$\sum_{j=1}^N x_{i,j} = 1 \quad i = 1, \dots, N, \quad (3)$$

$$x_{i,j} \in \{0, 1\} \quad i, j = 1, \dots, N. \quad (4)$$

Another common way of modeling assignment problems is by means of graph theory, through a complete bipartite graph, a structure where the set of vertices can be divided into two disjoint sets or classes, and the only edges connect vertices from one class to those of the other class. Let N be the problem dimensionality, the associated graph has $2N$ nodes, N of which represent agents while the rest represent jobs. The assignment costs, properly arranged in a $N \times N$ adjacency matrix, C , thus represent the weights of connections between nodes.

The resulting assignments can be also modeled as a permutation ϕ of the elements inside set I or set J , or with a permutation matrix X_ϕ , whose elements $x_{i,j} = 1$ if $j = \phi(i)$, and $x_{i,j} = 0$ if $j \neq \phi(i)$, the latter being the *Adjacency matrix* of a bipartite assignment graph (Fig. 1).

2.1. Hungarian algorithm

Formally, the Hungarian algorithm involves manipulating the weights of the bipartite graph in order to find a stable, minimum-weight perfect matching (see Ref. 33). However, most implementations use dynamic programming techniques by acting on cost matrix values, given the key observation that if a number is added to or subtracted from all of the entries of any one row or column of the cost matrix,

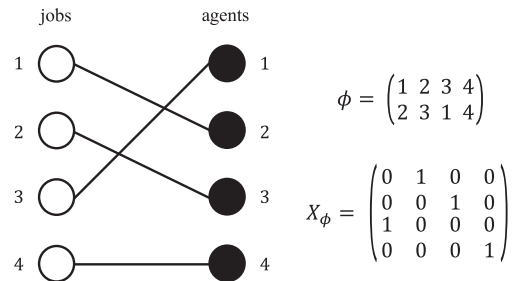


Fig. 1. Different representations for the same assignment permutation.

Algorithm 1. Hungarian Algorithm

Input: Cost matrix $C \in \mathbb{R}^{N \times N}$
Output: Assignment matrix $A \in \{0; 1\}^{N \times N}$

Step 1. Subtract the smallest element in each row from all elements in that row.
for $i \leftarrow 1$ to N **do**:
 $m_i = \min(C_{i,:})$
for $j \leftarrow 1$ to N **do**:
 $C_{i,j} = C_{i,j} - m_i$

Step 2. Subtract the smallest element in each column from all elements in that column.
for $j \leftarrow 1$ to N **do**:
 $m_j = \min(C_{:,j})$
for $i \leftarrow 1$ to N **do**:
 $C_{i,j} = C_{i,j} - m_j$

Step 3. Draw the minimum number of lines to cover all zeros in C .
 $l \leftarrow \text{minimum_lines_to_cover_zeros}(C)$
if $l == N$ **then**:
if $C_{i,j} == 0$ **then**:
 $A_{i,j} \leftarrow 1$
else:
 $A_{i,j} \leftarrow 0$
return A
else:
Step 4(a). Extract the submatrix C^* by selecting the columns and rows not yet covered.
 $C^* = C - \{\text{covered_rows_and_columns}\}$
Step 4(b). Find the smallest element in C^*
 $n \leftarrow \min(C^*)$
Step 4(c). Subtract n from each row of C^* ,
for $i \leftarrow 1$ to $|C^*|$ **do**:
 $C_{i,:} = C_{i,:} - n$
Step 4(d). Add n to each column of C^* ,
for $j \leftarrow 1$ to $|C^*|$ **do**:
 $C_{:,j} = C_{:,j} + n$
Go back to **Step 3**

then an optimal assignment for the resulting matrix is also an optimal assignment for the original cost matrix. The block above shows the pseudo-code formulation of the iterative algorithm on which most software libraries are based. When the algorithm ends, the entries of the returned matrix A containing ones indicate the optimal assignments, which ensure that all constraints are satisfied.

3. Graph Neural Networks

GNNs have a broad range of uses in various domains due to the prevalence of graph-structured data, where the lack of an Euclidean structure makes it challenging to use DNNs in a conventional way. In computer vision, GNNs find applications such as generating scene graphs,³⁴ classifying point clouds^{35,36}, and recognizing actions,³⁷ as they help understand the semantic relationships between objects in visual scenes. In natural language processing,^{38,39} GNNs leverage the connections between documents or words to predict document labels for text classification.

They are also useful for creating accurate models of smart transportation systems,^{40,41} where forecasting traffic speed, volume, and road density is crucial. Recommender systems benefit from graph-based frameworks⁴² that treat items and users as nodes. In chemistry, researchers employ GNNs to analyze the graph structures of molecules and compounds.^{43,44} In healthcare, GNNs can build knowledge graphs linking subjects with diseases or symptoms to assist physicians in decision-making.^{45–49} Lastly, in neuroscience, GNNs are valuable for mapping brain connectivity by identifying Regions of Interest (ROIs) from EEG signals^{50–57} and visualizing connections between key nodes, addressing a critical challenge in neuroscientific interpretation.

A graph is represented as $G = (V, E)$, where V is the set of *vertices* or *nodes*, and E is the set of *edges*; let $v_i \in V$ denote a node, and $e_{ij} = (v_i, v_j) \in E$ denote a directed edge from node i to node j . The overall topology is provided by the *Adjacency matrix* $A \in \{0, 1\}^{|V| \times |V|}$, where $A_{ij} = 1$ if $e_{ij} \in E$ and $A_{ij} = 0$ otherwise. A node may have numerical attributes, $\mathbf{x} \in \mathbb{R}^D$, as can edges, $\mathbf{e}_{ij} \in \mathbb{R}^F$.

The notion of graph neural networks was initially outlined in Gori *et al.*⁵⁸ and further elaborated in Scarselli *et al.*⁵⁹ These early models implicitly define the *Spatial Convolution* operator, which was later formalized with the concept of *Message Passing* (MP) mechanism; it tries to capture information by the graph manifold, edges and node feature vectors, by aggregating informative “messages” from a neighborhood of nodes. ConvGNNs generalize the operation of convolution, which is a widely popular concept on image processing field, from grid data to non-Euclidean graph data. A general framework is described by Eq. (5) and depicted in Fig. 2, it

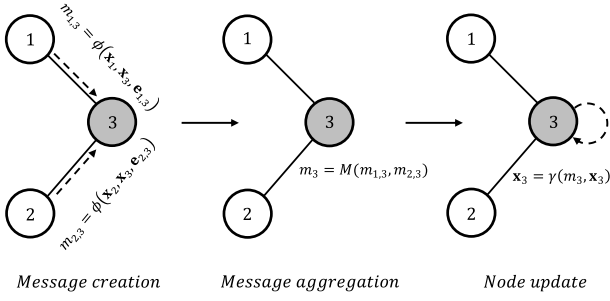


Fig. 2. The three main steps of the Message Passing paradigm in GNNs.

expresses the rule to update the attribute of node v_i , at network layer k :

$$\mathbf{x}_i^{(k)} = \gamma^{(k)}(\mathbf{x}_i^{(k-1)}, m_i^{(k)}), \quad (5)$$

where $m_i^{(k)}$ denotes the *message*, obtained by aggregating node v_i 's previous features $\mathbf{x}_i^{(k-1)}$, neighbors features $\mathbf{x}_j^{(k-1)}$ and their edge features $\mathbf{e}_{j,i}$:

$$m_i^{(k)} = M_{j \in \mathcal{N}(i)}(\phi^{(k)}(\mathbf{x}_i^{(k-1)}, \mathbf{x}_j^{(k-1)}, \mathbf{e}_{j,i})). \quad (6)$$

In the previous equation, M represents a differentiable, permutation invariant function (typically, *sum*, *mean* or *max*), while γ and ϕ denote differentiable parametric functions such as Multi Layer Perceptrons (MLPs).

An interesting strength point of ConvGNNs, compared with other architectures such as dense networks or CNNs, is that they better scale with the graph size, due to the localized action of the MP mechanism, which allows efficient parameters sharing and, consequently, memory savings.

3.1. GNN for combinatorial optimization problems

In the field of Combinatorial Optimization (CO), GNNs have already demonstrated their practical value. They have been used in various contexts, either to produce a solution directly or as an integrated component of an existing solver. Most of the previous works on such topic focused towards finding feasible, optimal or near-optimal solutions, while a smaller number tried to quantify the optimality of the proposed solution, or prove its infeasibility. Below, we briefly review the main works involving GNNs for basic CO problems. For a more extensive review, see Ref. 60.

Prates *et al.*⁶¹ trained a GNN in a supervised manner to solve small-scale instances (up to 105 cities)

of the Traveling Salesman Problem (TSP). A similar structure was further extended by Lemos *et al.*⁶² for the Graph Coloring Problem. To solve the TSP, Joshi *et al.*⁶³ suggested using Residual Graph Convolutional Neural Networks⁶⁴ in a supervised manner. Instead of producing a valid TSP tour, the model provides the probability that each edge belongs to the tour. Li *et al.*⁶⁵ used Graph Convolutional Networks³⁸ on combinatorial problems easily reducible to Maximum Independent Set (MIS) problems. Li *et al.*⁶⁶ studied the use of GNN for Graph Matching, that is, to search for an alignment between two graphs or sub-graphs, such that a cost function is minimized. Fey *et al.*⁶⁷ proposed an extended architecture for the same matching problem, in the first stage of which a GNN learns a node embedding to compute a similarity score between nodes based on local neighborhoods. A GNN-based architecture known as GraphSIM is presented by Bai *et al.*⁶⁸ to address the challenges of graph edit distance and maximum common sub-graph problems, in an end-to-end pipeline. A similar challenge to the one addressed in this paper is presented by Nowak *et al.*⁶⁹ who trained a GNN in a supervised manner to predict solutions to the Quadratic Assignment Problem (QAP). They modeled the QAP instances as two adjacency matrices and used the two corresponding graphs as input to the GNN.

4. Proposed method

In the following, given the assignment problem, the cost overview has been modeled with a fully connected bipartite graph, a structure where the set of vertices can be divided into two disjoint sets or classes, and the only edges connect vertices from one class to those of the other class. Let N be the problem dimensionality, the associated graph has $2N$ nodes; N of which represent agents while the rest represent jobs.

The input raw feature vectors of the nodes, $[\mathbf{x}_1 \dots \mathbf{x}_{2N}]$, are initialized with the cost values between source agents and receiving jobs, according to the cost matrix $C \in \mathbb{R}^{N \times N}$:

$$\mathbf{x}_i = [C_{i,1} \dots C_{i,N}] \quad i = 1, \dots, N \quad \text{agents}, \quad (7)$$

$$\mathbf{x}_{N+j} = [C_{1,j} \dots C_{N,j}] \quad j = 1, \dots, N \quad \text{jobs}. \quad (8)$$

Conversely, all the raw attributes of the edges \mathbf{e}_{ij} in the constructed graph are initialized with

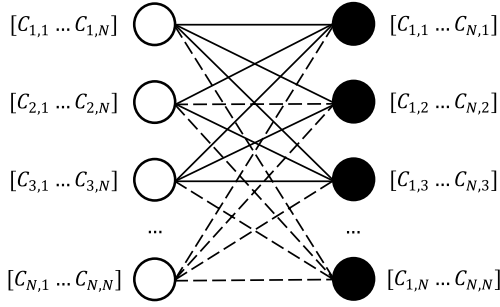


Fig. 3. Illustration of the nodes and edges definition, from the cost matrix to the corresponding bipartite graph.

zero-valued vectors, hence they are not taken into account by the convolution operator.

The GNN structure is composed of K layers, through which the graph preserves its bipartite layout, while node feature vectors are updated according to the MP operator expressed, for the generic node v and all its neighbors w , by the following equation:

$$\mathbf{x}_v^{(k)} = \frac{1}{|\mathcal{N}(v)|} \sum_{w \in \mathcal{N}(v)} \text{MLP}(\mathbf{x}_v^{(k-1)} | \mathbf{x}_w^{(k-1)}). \quad (9)$$

Comparing Eq. (9) with the general structure described in Sec. 3, it can be seen that messages are generated by a learnable function (MLP), a dense network whose input vector is obtained by concatenating the attribute vector of node v with that of each of its neighbors. Then, the average acts as the aggregation function of messages from neighboring nodes, collected in a set of K hops.

Finally, the feature map at last layer, $X^{(K)} = [\mathbf{x}_1^{(K)} \dots \mathbf{x}_{2N}^{(K)}]^\top$ is transformed to the actual output $Y \in \mathbb{R}^{N \times N}$ through a linear projection with learnable parameters $\Theta \in \mathbb{R}^{N \times 2N}$, to obtain the estimated assignment matrix:

$$Y = \Theta X^{(K)}. \quad (10)$$

4.1. Loss function and evaluation metric

The proposed model solves the assignment task as $2N$ separate classifiers to jointly comply both constraints of the assignment problem formulation (Eqs. (2) and (3)); at train stage, the predicted scores are obtained from the output logits Y by applying softmax operations, in both row-wise and column-wise directions:

$$\mathbf{r}_i = \text{softmax}(\mathbf{y}_{i,1} \dots \mathbf{y}_{i,N}), \quad (11)$$

$$\mathbf{c}_j = \text{softmax}(\mathbf{y}_{1,j} \dots \mathbf{y}_{N,j}), \quad (12)$$

then, given the ground truth binary assignment matrix $\hat{Y} \in \mathbb{R}^{N \times N}$, the cross-entropy loss is computed for each of the $2N$ separate classifiers, in the form of *negative-log likelihood*:

$$\mathcal{L}_r = -\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^N \mathbf{r}_{ij} \cdot \log(\hat{\mathbf{y}}_{i,j}), \quad (13)$$

$$\mathcal{L}_c = -\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^N \mathbf{c}_{ij} \cdot \log(\hat{\mathbf{y}}_{i,j}), \quad (14)$$

finally, the total loss $\mathcal{L} = \mathcal{L}_r + \mathcal{L}_c$ is backpropagated to update the network weights.

At inference stage, the output prediction matrix Y passes through a threshold criterion, to obtain a binary assignment map, whose rows and columns are one-hot encoded vectors. The criterion allows for “collision” avoidance (e.g. multiple jobs assigned to the same agent or multiple agents tasked with the same job); it consists of an iterative procedure, in which the assignment with the highest output value (thus the highest probability of being a correct match) is selected from the prediction matrix. Then the corresponding row and column are deleted, and the process is repeated until a single entry remains.

To benchmark our proposed approach we use accuracy as defined in Ref. 29, that is, the amount of jobs correctly matching their optimal agents, divided by N . This also lets us to compare directly with methods proposed in Ref. 29.

4.2. Network architecture

The Graph Neural Network consisted of $K = 2$ layers, which was experimentally proven to be the best performing depth for each of the problem dimensionality (N) evaluated. A higher number of layers showed the same level of accuracy, with the drawback of parameters, memory and time consumption increase, as evidenced by the ablation study of Sec. 5.2.

The MLP network used for message propagation has one input layer of size $2N$, an hidden layer with 128 neurons, ReLU activation and an output layer of size N . This configuration was applied identically for all investigated values: $N = \{2, 4, 8, 12, 16\}$. The choice of a single hidden layer for the MLP follows the network design made by the authors of Ref. 35

and Ref. 70, where the MP operator (9) was originally introduced.

4.3. Dataset

The policy we adopted to generate data samples follows the one implemented in the reference paper: we generated 100.000 synthetic cost matrices, drawing samples from a continuous uniform distribution: $c_{i,j} \sim U[0,1)$, then, 80% of such matrices were used for training, while the remaining 20% were reserved for the validation process.

When running experiments, several times with identical settings, we ensured that the chosen amount of data is sufficiently large to avoid the model to overfit. With the same criterion we generated 20.000 samples which are used for testing. In order to further reduce the dependence of the results with respect to a specific data distribution, we investigated a minor variation on train phase by generating different samples at every epoch, but did not obtain a noticeable improvement.

The ground truth decision matrix \hat{Y} is obtained at runtime for each sample, using the Hungarian algorithm; specifically we used the *munkres*⁷¹ Python package which implements the original algorithm.

4.4. Compared learning approaches

The MLP and CNN approaches we took as comparison²⁹ address LSAP by first decomposing it into N separate sub-assignment problems on how to assign one of N jobs to agent j . Only constraints of Eqs. (2) and (4) are strictly guaranteed, while the constraint of Eq. (3) is not taken into consideration at train time, hence there may exist some collisions such that one job may be assigned to different agents simultaneously. To prevent this issue, a greedy collision-avoidance rule is applied to finally state the actual assignment.

The MLP and CNN architectures developed in Ref. 29 consisted of N models in parallel; the former has four layers with 32, 64, 256, and N hidden neurons, and ReLU nonlinearity, while the latter (CNN) includes five convolutional layers, each containing 32, 32, 32, 32 and N kernels of size 1×1 , and an output projection map. In both cases, cross-entropy is taken as objective criterion, while Adam⁷² is used as optimization algorithm.

Table 1. Accuracy performance comparison for the proposed GNN architecture and reference learning approaches, for different size N .

Size N	2	4	8	12	16
MLP ²⁹	0.9849	0.9763	0.7019	0.5918	0.5614
CNN ²⁹	0.9974	0.9829	0.8295	0.6605	0.6274
GNN	0.9997	0.9660	0.8477	0.7856	0.7361

5. Experiments and Results

Models were trained up to 50 epochs; the learning rate was initially set to the value of $6 \cdot 10^{-3}$, and then halved if validation loss is not improved within a patience interval of five epochs. Empirically, it has been found that the learning rate is halved only once within the entire training stage; an extension of the training interval does not lead to further improvements. Stochastic Gradient Descent (SGD) was used as optimization algorithm, with L_2 weight decay of $5 \cdot 10^{-4}$. Once the training is over, the model checkpoint with best validation accuracy is selected and evaluated on the test set.

5.1. Results

We report in Table 1 the results obtained in terms of accuracy, in conjunction with the bar plot of Fig. 4, where we compared the proposed graph approach with the other solutions described on Sec. 4.4: MLP and CNN.

The proposed GNN model exhibited relative performance improvements of 0, 2% ($N = 2$), 2, 2%

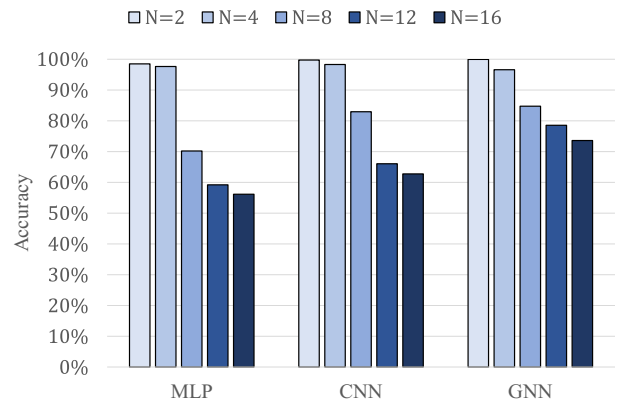


Fig. 4. Accuracy comparison bar chart between the proposed GNN architecture and reference learning approaches.

Table 2. Resource requirements and MAC operations for different models.

Size N	Learnable parameters					MACs				
	2	4	8	12	16	2	4	8	12	16
MLP ²⁹	38.8k	81.3k	183.1k	317.7k	497.4k	38.1k	79.8k	180.2k	313.3k	491.5k
CNN ²⁹	9.7k	13.7k	32.1k	64.4k	125.9k	26.4k	215.5k	1.79M	6.29M	15.46M
GNN	2.5k	4.9k	9.6k	14.4k	19.2k	18.4k	147.5k	1.18M	3.98M	9.44M

($N = 8$), 18, 9% ($N = 12$) and 17, 3% ($N = 16$) if compared with the conventional CNN. Improvements rise to 1, 5% ($N = 2$), 20, 8% ($N = 8$), 32,7% ($N = 12$) and 31, 1% ($N = 16$) if compared with the conventional MLP. Conversely, a slight worsening of accuracy has been observed for $N = 4$, however, the significant improvement achieved as N increases suggests that large LSAPs may benefit more from the GNN approach than the CNN or MLP ones. An interesting aspect which characterizes the GNN framework is the limited amount of memory required to store network parameters.

Since the MLP and the CNN approached the LSAP problem as N different classification sub-problems, the

parameters are shared to a limited extent within each of these classifiers; moreover, each of them needs the full cost matrix as input, which causes a noticeable overhead. On the other hand, the message passing operator (Eq. (9)) allows for efficient reuse of the internal MLP, since it operates at node-level. Table 2 reports the exact count of learnable parameters and MAC, Multiply and Accumulate operations involved for each of the considered architectures for different graph sizes; the values have been estimated by performing inference with a single-batch input sample. Figure 5 helps to better visualize the trends of required parameters (Fig. 5, top) and MAC (Fig. 5, bottom) as N increases.

Table 3 and Fig. 6 show the peak RAM memory use, for each of the considered methods, when solving an instance of the assignment problem. The results reveal that the CNN method has significantly more memory consumption than the MLP-based one. This is largely due to the fact that the allocation of the 2D convolutional kernels is quite costly. It is important to note, as previously mentioned, that these two methods actually use N models in parallel on sub-instances of the entire problem. The GNN network is positioned between MLP and CNN in terms of RAM usage, mainly due to the locality of the graph-convolutional operator and the fact it does not require N models in parallel. As we can see it is always less demanding with

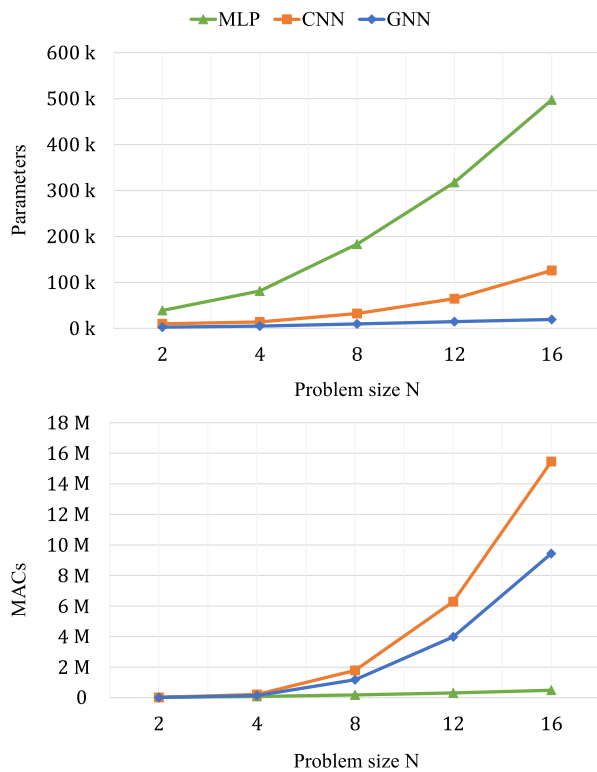


Fig. 5. Trend in demand for memory parameters requirements (top) and MAC operations (bottom).

Table 3. Measurements of peak RAM memory usage for the execution of a single LSAP instance. Values are in kB.

Size N	2	4	8	12	16
MLP ²⁹	6.1	12.2	24.6	37.3	50.2
CNN ²⁹	8.7	67.4	548.3	1900.4	4629.1
GNN	17.4	72.6	319.5	786.1	1480.0
Hungarian ¹⁴	56.0	176.0	608.0	1296.0	2240.0

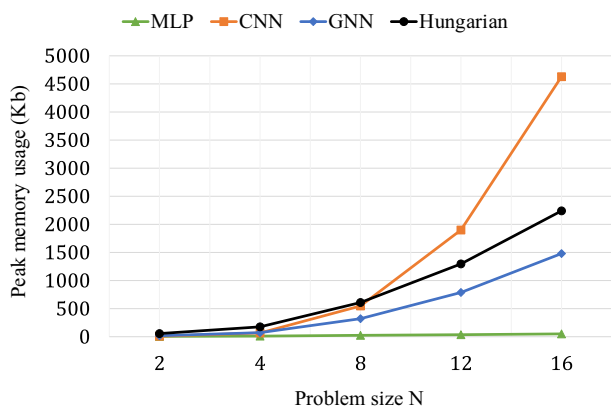


Fig. 6. Peak RAM memory usage for the different algorithms in exam.

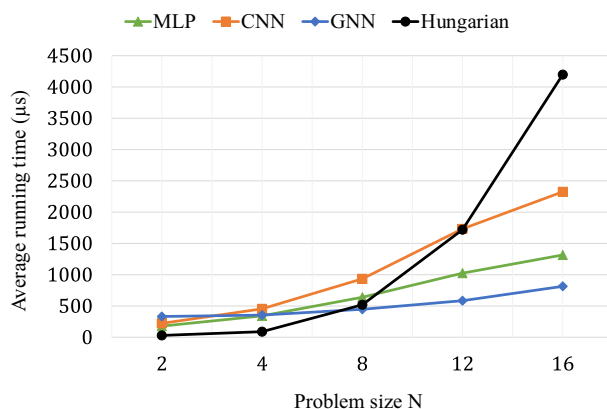


Fig. 7. CPU average running times for the different algorithms in exam.

respect to the Hungarian algorithm, for all addressed case studies (different N values).

The peak RAM occupation allows us to estimate the maximum memory footprint of a specific solver, and hence the hardware minimum requirements; in order to get a better insight into resource consumption (it is almost always possible to trade-off memory occupation for execution time) we perform also, in parallel, a time-profile analysis, reporting on Table 4 and Fig. 7 an estimation of mean execution time, obtained over 10.000 runs, between the proposed approach, the comparison ones, and the Hungarian algorithm implemented in plain Python language.⁷¹ We can note a cubic trend in time complexity, which makes the Hungarian algorithm impracticable to apply on resource-constrained devices, even for small dimensions.

Looking at Fig. 5 (bottom) we observe that the MAC operations required by the MLP are significantly lower than those required by the GNN, however, Fig. 7 seems to indicate an opposite trend. As before, this behavior is because execution times do not depend exclusively on the number of raw operations, but are also very sensitive to other

time-greedy operations such as read/write memory accesses. Indeed, we expect that the GNN, making heavy use of parameter sharing, can be significantly fast even if it experiences a high load in terms of MACs.

Figure 8 further emphasizes the trade-off between effectiveness and efficiency introduced by the graph-based approach, compared with the Hungarian algorithm.

Figure 9 shows the comparison between the accuracy obtained in terms of correct assignments and the cost accuracy. The latter quantifies the deviation of the obtained assignment cost from the optimal one. As already mentioned, the GNN is a sub-optimal solver, however, an interesting aspect emerges from this graph: although the number of correct assignments produced by the GNN is significantly lower than the optimal solution, the error in

Table 4. Measurements of CPU average load time when processing a single LSAP instance. Times are in μs .

Size N	2	4	8	12	16
MLP ²⁹	177.2	340.8	639.2	1023.6	1314.4
CNN ²⁹	221.5	454.0	934.0	1731.0	2324.4
GNN	331.1	353.8	446.7	583.5	813.3
Hung. ¹⁴	28.7	88.2	518.6	1719.2	4197.5

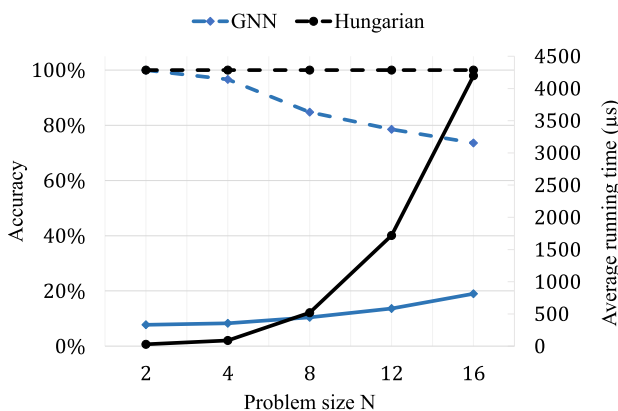


Fig. 8. Inference time (solid lines) versus accuracy (dashed lines).

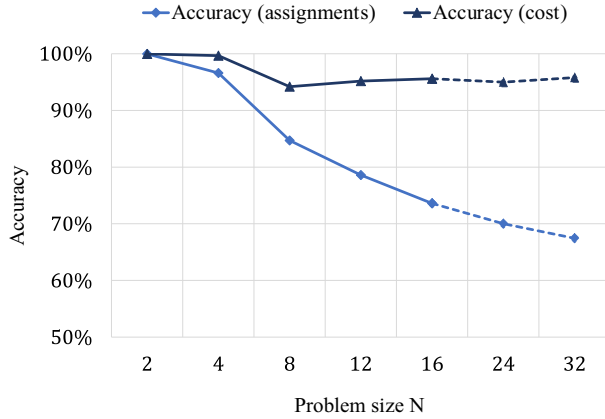


Fig. 9. Trends in accuracy of exact assignments and deviation from the minimum attainable cost.

terms of total cost is quite small. This indicates that the GNN is very efficient in determining those assignments that have a major impact on minimizing total cost, while errors are concentrated on the remaining assignments which have little impact on total cost. To further confirm this behavior, we performed additional experiments with $N = 24$ and $N = 32$, although previous analyses stop at $N = 16$ to comply with the configuration used in the comparison methods, MLP and CNN.

All experiments were conducted on an Ubuntu 16.04 machine, with six Intel(R) Core(TM) i7-6850K CPUs @ 3.60 GHz and 32GB RAM; network models were developed in Python language with PyTorch and PyTorch geometric⁷³ framework libraries.

5.2. Ablation studies

For ablation studies on the proposed model, we conducted experiments with different parameters, including batch size (BS) and number of GNN hidden layers (K), to explore how these factors may affect the overall accuracy.

Figure 10 demonstrates that a high batch value decelerates the convergence in terms of total required epochs. On the other hand, contrary to what was reported in Ref. 29, a low batch size did not lead to unstable convergence behavior. In fact we were able to achieve the best results in the least number of epochs, using the batch value of 1.

Figure 11 shows the results in terms of accuracy and inference time, achieved for several values of GNN hidden layers: $K = \{2, 3, 4, 5\}$.

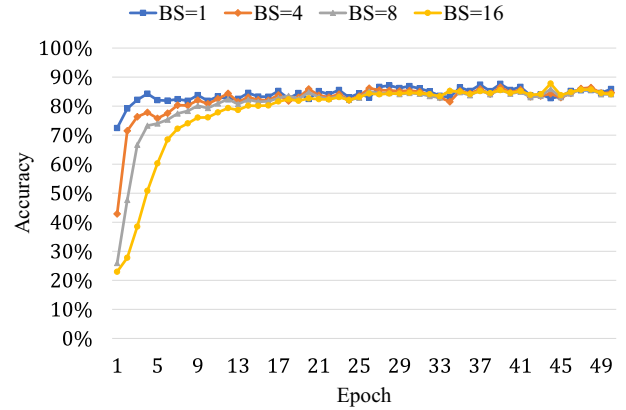


Fig. 10. Validation accuracy at different training epochs, for different batch size values.

The lack of improvement in accuracy, as depth increases, is due to a phenomenon, known in the GNN literature as *over-smoothing*,^{74–76} according to which node features tend to converge to the same vector and become nearly indistinguishable as the result of applying multiple graph convolutional layers.

6. Scheduling of Smart Meter Data Access in Electricity Distribution Grids

Nowadays, smart meters play a crucial role in the operation of modern power distribution networks. Although their functionality is mainly targeted for billing purposes, they have the potential to enable smart grid capabilities, such as forecasting of household consumption, photo-voltaic load matching and

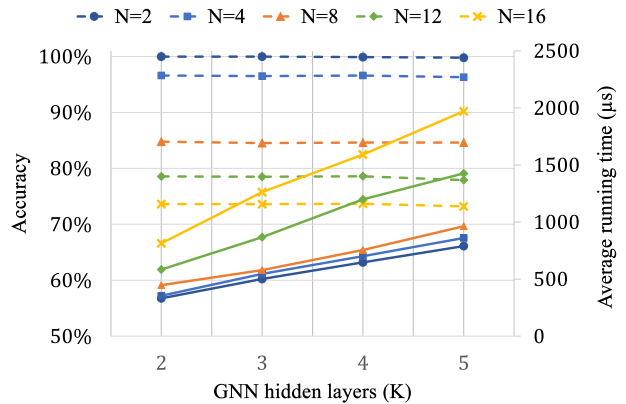


Fig. 11. Comparison plot, showing the relationship between GNN network depth (K), accuracy level (dashed lines) and average inference time (solid lines).

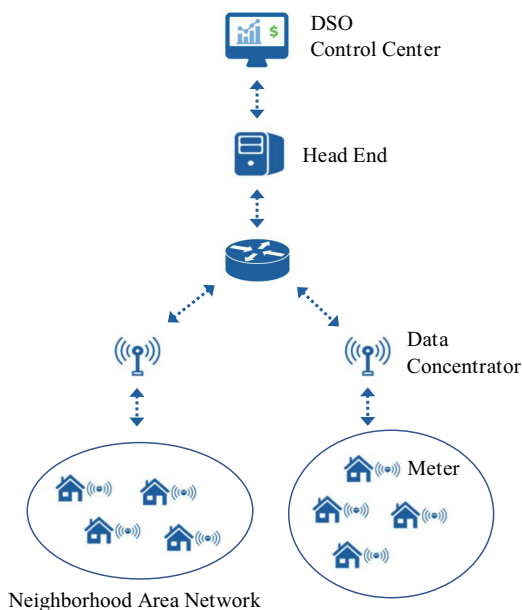


Fig. 12. Typical Advanced Metering Infrastructure (AMI).

detection of outages and flickers in Low Voltage (LV) grids.⁷⁷

Measurements of active power, current or voltage data are made at specific time intervals and rely on bidirectional communication to transmit information to a control center, utility or retail company.

These agents are coordinated through a centralized head-end system.⁷⁸ Due to bandwidth restrictions in communicating, the head-end accesses multiple meters sequentially, through data concentrators. The data are then transmitted in clusters, from the concentrators to the head-end system, but due to resource-constrained communication networks, it is challenging to obtain the relevant measurements in near real time.

Figure 12 depicts a high-level architecture of an Advanced Meter Infrastructure (AMI), consisting of Smart Meters (SMs), Data Concentrators (DC), Head-End station (HE) and Distribution System Operator (DSO) control center. The low bandwidth AMI communication networks between DC and smart meters result in high latency and infrequent updates, which strongly affects data quality in real-time data-demanding applications.

Classically, the information validity is quantified by the so-called *age*,⁷⁹ which is the time from measurement until the data are being utilized. A deeper

link from the information *age* to the signal dynamics leads to another common metric, called mismatch probability⁸⁰ (mmPr), which in previous works⁸¹ has shown to be a useful parameter for describing information quality.

Since the timings at which SMs are accessed directly affects the age of the information, altering the scheduling order can determine the quality of the data information.

Several works addressed the scheduling problem under the assumption of such metrics and concepts. In addition to Refs. 82 and 83, which face the problem without a specific policy but with brute-force analysis, Refs. 84–86 use joint routing and TDM-based scheduling in wireless mesh networks. These works optimized the scheduling algorithm by taking into account communication network constraints incurred due to wireless interference. Recently, Farooq *et al.*⁸⁷ experimented with the use of the Hungarian algorithm to find the best sequence for accessing smart meter data in case of low-performance networks.

6.1. System description

A time diagram showing how smart meters are accessed during an example reading cycle, is shown in Fig. 13. The DC employs a reactive approach, which means that it submits a request for access to meter data, and the meter replies with the needed response. This information is forwarded by DC to

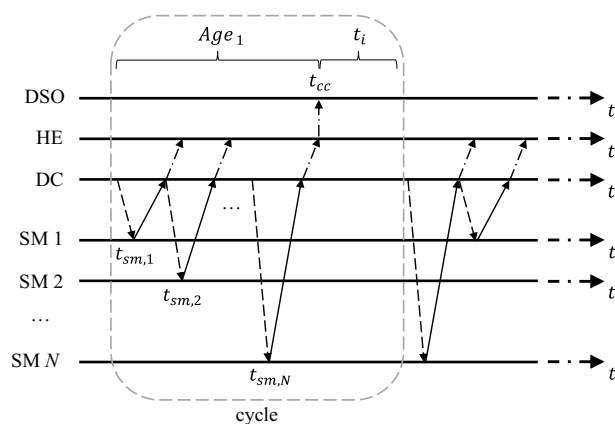


Fig. 13. Example of SM data access time diagram. Dashed arrows represent data requests from DC, while solid arrows represent responses from SM. Dash-dot arrows mark data feedback from DC to DSO through HE.

the Head-End (HE), which accumulates it and only at the end of the cycle forwards it to the controller or any other grid monitoring software. Every access to meters is associated by a specific access delay, which takes into account both communication stack delays and any potential cache or other systemic optimization measures that may have been used.

All of the smart meters' data are gathered within one collection cycle. When the previous data are transmitted to the controller, the subsequent cycle starts. After all the readings are taken, there is an idle time t_i that can be used for features like updating the firmware or sending alerts.

Due to their position in the timetable, each meter experiences different ageing. For instance, on the highlighted cycle of Fig. 13, meter N (smN) is accessed shortly before data transmission to the DSO system, since it is placed in the last position of the read queue; as a result, smN will have the shortest access delay and in a similar vein, $sm1$ will experience the longest access delay.

6.2. Information quality metrics

In this work, we considered the following performance metrics in relation to the access strategies:

- Information Age: the period of time between when data are acquired by the smart meter and when they are finally available to the DSO (for simplicity, the feedback time from HE to DSO is neglected). For example, the age of the n th meter is expressed by

$$\text{Age}_n = t_{cc} - t_{sm,n}, \quad (15)$$

where t_{cc} and $t_{sm,n}$ are taken from the same read cycle (see Fig. 13).

- Mismatch probability (mmPr): the probability that any of the N values of the information elements that are used at the requester does not match the current true value at the remote location, within a sensitivity threshold ε :

$$\text{mmPr}(n) = P\{I_n(t_{cc}) - I_n(t_{cc} - \text{Age}_n) > \varepsilon\}, \quad (16)$$

where $I_n(\cdot)$ indicates the information message, i.e. power, current or voltage readings, of n th smart meter.

6.3. Dataset

To perform the simulations we used the free-access dataset published by the Indian Council on Energy, Environment and Water, as part of the study “*What Smart Meters Can Tell Us, Insights on Electricity Supply and Use in Mathura and Bareilly Households*”.⁸⁸

The data were collected using smart meters installed in nearly 100 urban households in Mathura and Bareilly districts of Uttar Pradesh, India. These smart meters recorded electricity consumption patterns and power supply information at three-minute intervals, from May 2019 to October 2021. The data also provided information on the situation of power supply (including hours timestamp, voltage, current withdrawn and other related variables). Figure 14 shows an example of the data series contained in the dataset; the consumption profile recorded from two households, over a 24 h range.

6.4. GNN assignment solver

To assess the impact of the access scheduling, we performed an analysis based on the system described above. We considered coverage areas with 4, 8, 16 and 32 smart meters and we made the following assumptions about the system dynamics:

- Meters are queried at fixed intervals of 15 min. This is currently the most frequent time resolution of meters readings, as stated in Refs. 82 and 87. In addition, this assumption allows for a high level of

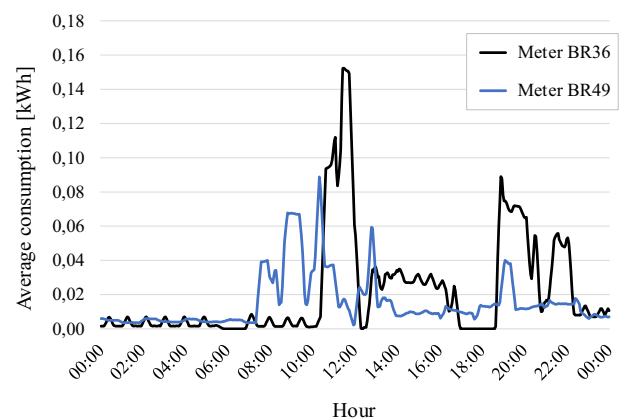


Fig. 14. Consumption profile of two different households over 24 h.

abstraction that does not take into account the type of connection between each meter and the control center, whether it is through a slow or unreliable line, thus with high latency times, or through a reliable, wide-band network. We assume that within the established time interval, data communication has occurred correctly.

- All meters are polled every cycle. This is a simplification since in real-world scenarios some SMs may be queried at a lower rate and thus excluded from one or more cycles. However, we bypass this option, which can be the subject of a future parametric study.
- Based on simulation studies, and relying on previous works^{82,87} we set the tolerance threshold for the mmPr parameter (ϵ) at 10% of the mean power value.

The first step in the analysis is to determine the mmPr profiles for each of the meters, simulating every possible position in the read queue. As mentioned earlier, we considered four different scenarios for the total number of SMs inside the domain of a data concentrator: 4, 8, 16 and 32. We estimated the mmPr values over a 24 h time frame, as we believe that the nonstationarity of the consumption data requires recalculation of the reading order after this time interval, to maintain the optimality of the scheduling criterion.

Figure 15 shows the trend of calculated mmPr values during a 24 h time interval, for 8 smart meters from the dataset. Both the observation day and the

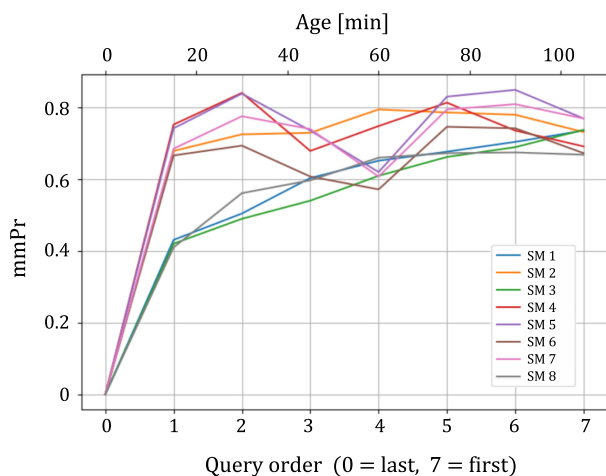


Fig. 15. Calculated mmPr profiles, in relation to information age and position of SM in the reading queue.

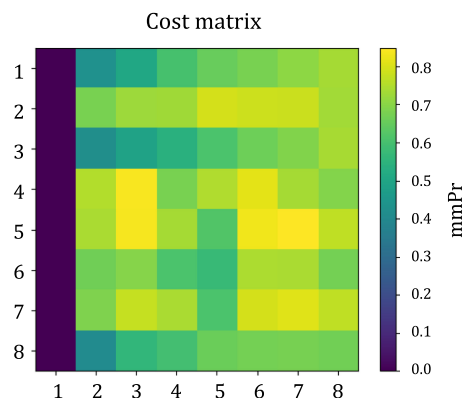


Fig. 16. Example of the 8×8 cost matrix, related to the mmPr curves of Fig. 15.

smart meters were chosen with a random criterion, only for illustrative purposes.

Once the mmPr values for a single day were obtained, the problem of determining the reading order was easily modeled as a Linear Sum Assignment Problem (LSAP) because a one-to-one assignment must be determined between the meters and the positions in the read queue in order to identify the lowest “cost” that is the overall mmPr of a schedule.

The cost matrix was then defined by vectorizing the mmPr curves and overlaying them in an $N \times N$ grid, as shown on Fig. 16. The range of cost values lies in $[0, 1]$, so we could use the same training strategy as defined in Sec. 4.

6.5. Results and discussion

In the following, two methods were applied to solve the assignment problem, the Hungarian algorithm, as defined in Ref. 14, and the GNN data-driven assignment method, which is sub-optimal but has lower complexity. The results are presented in the following tables.

Moreover, the fastest way to deal with the assignment problem is to disregard cost optimization and make random assignments. This obviously eliminates the computation time but drastically lowers the level of accuracy, which, for a problem of size N , can be calculated on a statistical basis as $1/N!$ being $N!$ the possible number of valid assignments.

Table 5 shows the total cost obtained from each of the two allocation methods, i.e. the minimum attainable mean mmPr. This value cannot be zero, since even with a few meters within each read cycle,

Table 5. Assignment cost, e.g. mmPr values obtained by the Hungarian algorithm (Min), the worst possible solution (Max) and the GNN method.

Problem size	Min	Max	GNN	Err %
$N = 4$	0.4134	0.5309	0.4153	1.61%
$N = 8$	0.5032	0.6559	0.5071	2.55%
$N = 16$	0.5661	0.7195	0.5699	2.48%
$N = 32$	0.6318	0.8057	0.6353	2.01%

mismatch of information is virtually inevitable. The minimum value of mmPr, given in the second column, also corresponds to that obtained by the Hungarian algorithm, since it is optimal. In contrast, the GNN approach fails to always determine the correct assignment, leading to sub-optimal scheduling. The column labeled with “Max” reports the upper bound of mmPr values, achieved with the worst scheduling. The rightmost column of Table 5 reports the percentage deviation between the GNN prediction and the minimum value, within the spanning range.

Table 6 shows the accuracy in terms of matching assignments, between the optimal solution and the GNN solver.

Finally, Table 7 reports the time results in the execution of the two algorithms, which is the main motivation in favor of the GNN approach.

As pointed out earlier, the size of the assignment problem greatly affects the execution time of the Hungarian algorithm, although it manages to reduce its complexity to $\mathcal{O}(N^3)$, which is much lower than the brute-force solution. Table 7 shows that GNN produces faster assignments, by about $1.1\times$, for $N = 8$, $5.1\times$ for $N = 16$ and $28.2\times$ for $N = 32$. The trend suggests that the advantage would grow for larger N . This is significant in real-world scenarios, as some smart grids may hold up to a hundred meters within the same subsystem, as stated in Ref. 82.

Table 6. Accuracy levels obtained by the Hungarian algorithm and the GNN method.

Problem size	Hungarian	GNN
$N = 4$	100%	85.0%
$N = 8$	100%	75.0%
$N = 16$	100%	66.5%
$N = 32$	100%	65.0%

Table 7. Mean execution time for a single-batch LSAP instance. Values are in μs .

Problem size	Hungarian	GNN
$N = 4$	88.2	353.8
$N = 8$	518.6	446.7
$N = 16$	4197.5	813.3
$N = 32$	34064.8	1209.0

7. Conclusions

In this work we proposed a novel learning framework by adapting a data-driven approach of the linear sum assignment problem, and then compared the performances with two existing DNN-based strategies.

We demonstrated experimentally that the proposed approach has competitive performance, compared to previous MLP and CNN based approaches, regarding small assignment problems. For larger problems it is able to outperform significantly these approaches, while requiring significantly less computational resources.

Subsequently, this study introduced the typical components of an AMI, Advanced Meter Infrastructure of a smart grid, explaining how a proper scheduling of the smart meters query order can minimize information obsolescence during the data collection process. Smart meter power readings from different grid spans are employed, as experimental settings. Although simulation results show lower accuracy of the GNN approach compared to the optimal solution, there is a notable gain in computational demand and execution time, the more so as the number of counters in the neighborhood area network is larger.


Furthermore, we observed that the incorrect assignments of the GNN network actually minimally affect the error in terms of total cost. This indicates that although the network has been trained to minimize the assignments on a binary cross-entropy basis, it also succeeds to exploit the cost values on its decision mechanism, and tends to produce assignments which while not optimal (lower accuracy) have total assignment cost close to the optimal solution.


Future work will attempt to explore new application areas for the assignment problem, such as integrating the proposed GNN solver into a


speaker-independent multi-talker speech separation model^{89–91} in order to increase performance or speed up training time. In such systems the network produces multiple outputs in an unpredictable order, which must be assigned to the corresponding target signals to perform properly supervised training. Assignment is made based on a metric that quantifies the similarity between each target signal and the possible candidate output.

Another potential application we might consider is the use of the GNN LSAP solver for Multi-Object Tracking and Segmentation (MOTS). This involves not only detecting and segmenting objects in a sequence of video frames, but also assigning consistent IDs to each visible instance of the same object. Typically, this is formulated as consecutive assignment problems involving a few dozen objects and must be carried out under tight constraints, to be in synch with a real-time video stream.⁹² In such context, the use of the proposed sub-optimal GNN solver could bring considerable advantage.

ORCID

Carlo Aironi  <https://orcid.org/0000-0002-5754-4864>

Samuele Cornell  <https://orcid.org/0000-0002-5358-1844>

Stefano Squartini  <https://orcid.org/0000-0001-9374-0128>

References

1. R. Burkard and E. Dragoti-Cela, *Linear Assignment Problems and Extensions*, 1st edn. (Kluwer Academic Publishers, 1999).
2. W. Lian and L. Zhang, A concave optimization algorithm for matching partially overlapping point sets, *Pattern Recogn.* **103** (2020) 107322.
3. N. S. Hirata and F. D. Julca-Aguilar, Matching based ground-truth annotation for online handwritten mathematical expressions, *Pattern Recogn.* **48**(3) (2015) 837–848.
4. K. Smith, D. Gatica-Perez, J.-M. Odobez and S. Ba, Evaluating multi-object tracking, in *IEEE Computer Society Conf. Computer Vision and Pattern Recognition* (IEEE, 2005), pp. 36–36.
5. R. M. Haralick and L. G. Shapiro, Image segmentation techniques, *Comput. Vision, Graphics Image Process.* **29**(1) (1985) 100–132.
6. G. Yu, L. Xu, D. Feng, R. Yin, G. Y. Li and Y. Jiang, Joint mode selection and resource allocation for device-to-device communications, *IEEE Trans. Commun.* **62**(11) (2014) 3814–3824.
7. X. Lu, Q. Ni, W. Li and H. Zhang, Dynamic user grouping and joint resource allocation with multi-cell cooperation for uplink virtual MIMO systems, *IEEE Trans. Wireless Commun.* **16**(6) (2017) 3854–3869.
8. H. Song, X. Fang and Y. Fang, Unlicensed spectrum fusion and interference coordination for ITE systems, *IEEE Trans. Mobile Comput.* **15**(12) (2016) 3171–3184.
9. D. Yu, M. Kolbæk, Z.-H. Tan and J. Jensen, Permutation invariant training of deep models for speaker-independent multi-talker speech separation, in *IEEE Int. Conf. Acoustics, Speech and Signal Processing* (IEEE, 2017), pp. 241–245.
10. Y. Fujita, N. Kanda, S. Horiguchi, Y. Xue, K. Nagamatsu and S. Watanabe, End-to-end neural speaker diarization with self-attention, in *IEEE Automatic Speech Recognition and Understanding Workshop* (IEEE, 2019), pp. 296–303.
11. D. Raj, L. P. Garcia-Perera, Z. Huang, S. Watanabe, D. Povey, A. Stolcke and S. Khudanpur, Dover-lap: A method for combining overlap-aware diarization outputs, in *IEEE Spoken Language Technology Workshop* (IEEE, 2021), pp. 881–888.
12. S. Watanabe, M. Mandel, J. Barker and E. Vincent, Chime-6 challenge: Tackling multispeaker speech recognition for unsegmented recordings, in *6th Int. Workshop on Speech Processing in Everyday Environments*, Barcelona/Virtual, Spain, 2020, pp. 1–7.
13. H. W. Kuhn, The hungarian method for the assignment problem, *Naval Res. Logistics Quart.* **2**(1–2) (1955) 83–97.
14. J. Munkres, Algorithms for the assignment and transportation problems, *J. Soc. Indus. Appl. Math.* **5**(1) (1957) 32–38.
15. D. P. Bertsekas, The auction algorithm: A distributed relaxation method for the assignment problem, *Ann. Oper. Res.* **14**(1) (1988) 105–123.
16. R. Jonker and T. Volgenant, A shortest augmenting path algorithm for dense and sparse linear assignment problems, in *DGOR/NSOR: Papers of the 16th Annual Meeting of DGOR in Cooperation with NSOR/Vorträge der 16. Jahrestagung der DGOR zusammen mit der NSOR* (Springer, 1988), pp. 622–622.
17. N. Karmarkar and K. Ramakrishnan, Computational results of an interior point algorithm for large scale linear programming, *Math. Program.* **52**(1) (1991) 555–586.
18. K. Ramakrishnan, N. Karmarkar and A. P. Kamath, An approximate dual projective algorithm for solving assignment problems, in *Network Flows and Matching* (American Mathematical Society, 1991), pp. 431–451.
19. M. Akgül and O. Ekin, A dual feasible forest algorithm for the linear assignment problem, *RAIRO-Oper. Res.* **25**(4) (1991) 403–411.

20. N. Karmarkar, A new polynomial-time algorithm for linear programming, in *Proc. 16th Annual ACM Sympo. Theory of Computing* (Society for Industrial and Applied Mathematics, 1984), pp. 302–311.
21. M. A. Trick, A linear relaxation heuristic for the generalized assignment problem, *Naval Res. Logist.* **39**(2) (1992) 137–151.
22. R. A. Murphey, P. M. Pardalos and L. S. Pitsoulis, A greedy randomized adaptive search procedure for the multitarget multisensor tracking problem, in *Network Design: Connectivity and Facilities Location* 40 (American Mathematical Society, 1997), pp. 277–302.
23. A. Naiem, M. El-Beltagy and P. Ab, Deep greedy switching: A fast and simple approach for linear assignment problems, in *7th Int. Conf. Numerical Analysis and Applied Mathematics* (AIP, 2009), p. 9999.
24. A. Naiem and M. El-Beltagy, On the optimality and speed of the deep greedy switching algorithm for linear assignment problems, in *IEEE Int. Symp. Parallel & Distributed Processing, Workshops and Phd Forum* (IEEE, 2013), pp. 1828–1837.
25. H. Sun, X. Chen, Q. Shi, M. Hong, X. Fu and N. D. Sidiropoulos, Learning to optimize: Training deep neural networks for wireless resource management, in *IEEE 18th Int. Workshop on Signal Processing Advances in Wireless Communications* (IEEE, 2017), pp. 1–6.
26. P. de Kerret, D. Gesbert and M. Filippone, Team deep neural networks for interference channels, in *IEEE Int. Conf. Communications Workshops* (IEEE, 2018), pp. 1–6.
27. H. Sun, X. Chen, Q. Shi, M. Hong, X. Fu and N. D. Sidiropoulos, Learning to optimize: Training deep neural networks for interference management, *IEEE Trans. Signal Process.* **66**(20) (2018) 5438–5453.
28. Q. Liu and Y. Zhao, A one-layer projection neural network for linear assignment problem, in *2015 34th Chinese Control Conference* (IEEE, 2015), pp. 3548–3552.
29. M. Lee, Y. Xiong, G. Yu and G. Y. Li, Deep neural networks for linear sum assignment problems, *IEEE Wireless Commun. Lett.* **7**(6) (2018) 962–965.
30. N. Minh-Tuan and Y.-H. Kim, Bidirectional long short-term memory neural networks for linear sum assignment problems, *Appl. Sci.* **9**(17) (2019) 3470.
31. Y. Xu, A. Osep, Y. Ban, R. Horaud, L. Leal-Taixé and X. Alameda-Pineda, How to train your deep multi-object tracker, in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition* (IEEE, 2020), pp. 6787–6796.
32. C. Aironi, S. Cornell and S. Squartini, Tackling the linear sum assignment problem with graph neural networks, in *Int. Conf. Applied Intelligence and Informatics*, Reggio Calabria, Italy, September 1–3, 2022 (Springer, Nature, 2022), pp. 90–101.
33. A. Grinman, *The Hungarian Algorithm for Weighted Bipartite Graphs*, Massachusetts Institute of Technology (2015).
34. J. Johnson, A. Gupta and L. Fei-Fei, Image generation from scene graphs, in *Proc. IEEE Conf. Computer Vision and Pattern Recognition* (IEEE, 2018), pp. 1219–1228.
35. Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein and J. M. Solomon, Dynamic graph CNN for learning on point clouds, *ACM Trans. Graph.* **38** (2019) 12.
36. L. Landrieu and M. Simonovsky, Large-scale point cloud semantic segmentation with superpoint graphs, in *2018 IEEE/CVF Conf. Computer Vision and Pattern Recognition*, Vol. 1 (IEEE, 2017), pp. 4558–4567.
37. S. Yan, Y. Xiong and D. Lin, Spatial temporal graph convolutional networks for skeleton-based action recognition, in *Proc. 32 AAAI Conf. Artificial Intelligence and 30th Innovative Applications of Artificial Intelligence Conference and 8th AAAI Symp. Educational Advances in Artificial Intelligence* (AAAI Press, 2018), pp. 7444–7452.
38. T. N. Kipf and M. Welling, Semi-supervised classification with graph convolutional networks, in *Proc. 5th Int. Conf. Learning Representations* (OpenReview.net, 2017).
39. W. Hamilton, Z. Ying and J. Leskovec, Inductive representation learning on large graphs, *Adv. Neural Inf. Process. Syst.* **30** (2017) 1025–1035.
40. Y. Li, R. Yu, C. Shahabi and Y. Liu, Diffusion convolutional recurrent neural network: Data-driven traffic forecasting, in *Int. Conf. Learning Representations* (OpenReview.net, 2018).
41. B. Yu, H. Yin and Z. Zhu, Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting, in *Int. Joint Conf. Artificial Intelligence Organization* (AAAI Press, 2018), pp. 3634–3640.
42. R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton and J. Leskovec, Graph convolutional neural networks for web-scale recommender systems, in *Proc. 24th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining* (Association for Computing Machinery, New York, 2018), p. 974983.
43. J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals and G. E. Dahl, Neural message passing for quantum chemistry, in *Proc. 34th Int. Conf. Machine Learning*, eds. D. Precup and Y. W. Teh, Proceedings of Machine Learning Research, Vol. 70 (JMLR.org, 2017), pp. 1263–1272.
44. A. Fout, J. Byrd, B. Shariat and A. Ben-Hur, Protein interface prediction using graph convolutional networks, *Adv. Neural Inf. Process. Syst.* **30** (2017) 6533–6542.
45. E. Choi, Z. Xu, Y. Li, M. Dusenberry, G. Flores, E. Xue and A. Dai, Learning the graphical structure

- of electronic health records with graph convolutional transformer, *Proc. AAAI Conf. Artif. Intell.* **34** (2020) 606–613.
46. M. Ahmadlou, H. Adeli and A. Adeli, Improved visibility graph fractality with application for the diagnosis of autism spectrum disorder, *Phys. A: Stat. Mech. Appl.* **391** (2012) 4720–4726.
 47. K. Raeisi *et al.*, A class-imbalance aware and explainable spatio-temporal graph attention network for neonatal seizure detection, *Int. J. Neural Syst.* **33**(9) (2023) 2350046.
 48. Y. Zhao, M. Xue, C. Dong, J. He, D. Chu, G. Zhang, F. Xu, X. Ge and Y. Zheng, Automatic seizure identification from eeg signals based on brain connectivity learning, *Int. J. Neural Syst.* **32**(11) (2022) 2250050.
 49. S. Zhang, H. Wang, Z. Zheng, T. Liu, W. Li, Z. Zhang and Y. Sun, Multi-view graph contrastive learning via adaptive channel optimization for depression detection in eeg signals, *Int. J. Neural Syst.* **33**(11) (2023) 2350055.
 50. J. Lian and F. Xu, Epileptic eeg classification via graph transformer network, *Int. J. Neural Syst.* **33**(8) (2023) 2350042.
 51. X. Che, Y. Zheng, X. Chen, S. Song and S. Li, Decoding color visual working memory from EEG signals using graph convolutional neural networks, *Int. J. Neural Syst.* **32**(2) (2022) 2250003.
 52. M. Ahmadlou, H. Adeli and A. Adeli, New diagnostic EEG markers of the alzheimers disease using visibility graph, *J. Neural Trans.* **117** (2010) 1099–1109.
 53. M. Ahmadlou, H. Adeli and A. Adeli, Graph theoretical analysis of organization of functional brain networks in ADHD, *Clinical EEG Neurosci. : Official J. EEG Clin. Neurosci. Soc.* **43** (2012) 5–13.
 54. M. Ahmadlou and H. Adeli, Complexity of weighted graph: A new technique to investigate structural complexity of brain activities with applications to aging and autism, *Neurosci. Lett.* **650** (2017) 103–108.
 55. J. delEtoile and H. Adeli, Graph theory and brain connectivity in alzheimer’s disease, *Neurosci.: A Rev. J. Bringing Neurobiol. Neurol. Psych.* **23** (2017) 616–626.
 56. M. H. Rafiei, L. V. Gauthier, H. Adeli and D. Takabi, Self-supervised learning for electroencephalography, *IEEE Trans. Neural Netw. Learn. Syst.* (2022) 1–15.
 57. N. Feng, F. Hu, H. Wang and B. Zhou, Motor intention decoding from the upper limb by graph convolutional network based on functional connectivity, *Int. J. Neural Syst.* **31**(12) (2021) 2150047.
 58. M. Gori, G. Monfardini and F. Scarselli, A new model for learning in graph domains, in *Int. Joint Conf. Neural Networks*, Vol. 2 (IEEE, 2005), pp. 729–734.
 59. F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner and G. Monfardini, The graph neural network model, *IEEE Trans. Neural Netw.* **20**(1) (2008) 61–80.
 60. Q. Cappart, D. Chételat, E. B. Khalil, A. Lodi, C. Morris and P. Veličković, Combinatorial optimization and reasoning with graph neural networks, in *Proc. 30th Int. Joint Conf. Artificial Intelligence*, ed. Z.-H. Zhou (2021), pp. 4348–4355.
 61. M. Prates, P. H. Avelar, H. Lemos, L. C. Lamb and M. Y. Vardi, Learning to solve np-complete problems: A graph neural network for decision TSP, in *Proc. AAAI Conf. Artificial Intelligence*, Vol. 33(1), (AAAI Press, 2019), pp. 4731–4738.
 62. H. Lemos, M. Prates, P. Avelar and L. Lamb, Graph colouring meets deep learning: Effective graph neural network models for combinatorial problems, in *IEEE 31st Int. Conf. Tools with Artificial Intelligence* (IEEE, 2019), pp. 879–885.
 63. K. J. Chaitanya, T. Laurent and X. Bresson, An efficient graph convolutional network technique for the travelling salesman problem, preprint (2019), arXiv:abs/1906.01227.
 64. X. Bresson and T. Laurent, Residual gated graph convnets, preprint (2017), arXiv:1711.07553.
 65. Z. Li, Q. Chen and V. Koltun, Combinatorial optimization with graph convolutional networks and guided tree search, in *Advances in Neural Information Processing Systems*, eds. S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi and R. Garnett, Vol. 31 (Curran Associates Inc., 2018), p. 13.
 66. Y. Li, C. Gu, T. Dullien, O. Vinyals and P. Kohli, Graph matching networks for learning the similarity of graph structured objects, in *Proc. 36th Int. Conf. Machine Learning*, eds. K. Chaudhuri and R. Salakhutdinov, Proceedings of Machine Learning Research, Vol. 97 (2019), pp. 3835–3845.
 67. M. Fey, J. E. Lenssen, C. Morris, J. Masci and N. M. Kriege, Deep graph matching consensus, in *Int. Conf. Learning Representations* (PMLR, 2020), pp. 1220–1243.
 68. Y. Bai, H. Ding, K. Gu, Y. Sun and W. Wang, Learning-based efficient graph similarity computation via multi-scale convolutional set matching, in *Proc. AAAI Conf. Artificial Intelligence*, Vol. 34 (AAAI Press, 2020), pp. 3219–3226.
 69. A. W. Nowak, S. Villar, A. S. Bandeira and J. Bruna, Revised note on learning algorithms for quadratic assignment with graph neural networks, preprint (2017), arXiv:abs/1706.07450, 1–5.
 70. C. R. Qi, H. Su, K. Mo and L. J. Guibas, Pointnet: Deep learning on point sets for 3d classification and segmentation, preprint (2016), arXiv:1612.00593.
 71. B. M. Clapper, Munkres implementation for python (2022), <https://github.com/bmc/munkres>.
 72. D. Kingma and J. Ba, Adam: A method for stochastic optimization, in *International Conf. Learning Representations* (OpenReview.net, 2014) p. 15.

73. M. Fey and J. E. Lenssen, Fast graph representation learning with pytorch geometric, preprint (2019), arXiv: abs/1903.02428.
74. K. Oono and T. Suzuki, Graph neural networks exponentially lose expressive power for node classification, preprint (2019), arXiv: Learning abs/1905.10947.
75. Q. Li, Z. Han and X.-M. Wu, Deeper insights into graph convolutional networks for semi-supervised learning, in *Proc. 32nd AAAI Conf. Artificial Intelligence and 30th Innovative Applications of Artificial Intelligence Conference and 8th AAAI Symposium on Educational Advances in Artificial Intelligence* (AAAI Press, 2018), pp. 3538–3545.
76. H. NT and T. Maehara, Revisiting graph neural networks: All we have is low-pass filters, preprint (2019), arXiv: abs/1905.09550.
77. B. Völker, A. Reinhardt, A. Faustine and L. Pereira, Watts up at home? Smart meter data analytics from a consumer-centric perspective, *Energies* **14** (2021) 719.
78. F. Lemerrier, G. Habault, G. Z. Papadopoulos, P. Maill, N. Montavont and P. Chatzimisios, Communication architectures and technologies for advanced smart grid services: Communication networks and services, in *Transportation and Power Grid in Smart Cities* (Wiley Telecom, 2018), pp. 217–245.
79. R. D. Yates, The age of information in networks: Moments, distributions, and sampling, *IEEE Trans. Inf. Theory* **66**(9) (2020) 5712–5728.
80. M. Bøgsted, R. L. Olsen and H.-P. Schwefel, Probabilistic models for access strategies to dynamic information elements, *Performance Evaluation* **67**(1) (2010) 43–60.
81. M. S. Kemal, R. L. Olsen and H.-P. Schwefel, Information quality aware data collection for adaptive monitoring of distribution grids, in *The 1st EAI Int. Conf. Smart Grid Assisted Internet of Things* (EAI, 2017), pp. 11–20.
82. M. S. Kemal, R. L. Olsen and H.-P. Schwefel, Optimized scheduling of smart meter data access: A parametric study, in *2018 IEEE Int. Conf. Communications, Control, and Computing Technologies for Smart Grids* (IEEE, 2018), pp. 1–6.
83. A. Shawky, R. Olsen, J. Pedersen and H. Schwefel, Network aware dynamic context subscription management, *Comput. Netw.* **58** (2014) 239–253.
84. K. Jain, J. Padhye, V. N. Padmanabhan and L. Qiu, Impact of interference on multi-hop wireless network performance, in *Proc. 9th Annual Int. Conf. Mobile Computing and Networking* (Springer, 2003), pp. 66–80.
85. R. L. Cruz and A. V. Santhanam, Optimal routing, link scheduling and power control in multi-hop wireless networks, in *IEEE INFOCOM 2003 22nd Annual Joint Conf. IEEE Computer and Communications Societies*, Vol. 1 (IEEE, 2003), pp. 702–711.
86. J. Zhang, H. Wu, Q. Zhang and B. Li, Joint routing and scheduling in multi-radio multi-channel multi-hop wireless networks, in *2nd Int. Conf. Broadband Networks* (IEEE, 2005), pp. 631–640.
87. A. Farooq, K. Shahid and R. L. Olsen, Scheduling of smart meter data access using hungarian algorithm, in *25th Int. Symp. Wireless Personal Multimedia Communications* (IEEE, 2022), pp. 351–356.
88. S. Agrawal, S. Mani, K. Ganesan and A. Jain, *What Smart Meters Can Tell Us: Insights on Electricity Supply and Use in Mathura and Bareilly Households: Report* (Council on Energy, Environment and Water, 2020).
89. T. von Neumann, C. Boeddeker, K. Kinoshita, M. Delcroix and R. Haeb-Umbach, Speeding up permutation invariant training for source separation, in *ITG Conf. Speech Communication* (IEEE, 2021), pp. 99–103.
90. M. Yousefi, S. Khorram and J. Hansen, Probabilistic permutation invariant training for speech separation, in *Proc. Interspeech 2021* (IEEE, 2019), pp. 4604–4608.
91. S. Dovrat, E. Nachmani and L. Wolf, Many-speakers single channel speech separation with optimal permutation training, in *Proc. Interspeech 2021* (IEEE, 2021), pp. 3890–3894.
92. A. Choudhuri, G. Chowdhary and A. G. Schwing, Assignment-space-based multi-object tracking and segmentation, in *2021 IEEE/CVF Int. Conf. Computer Vision* (IEEE, 2021), pp. 13578–13587.