

Sparse Clustered Neural Networks for the Assignment Problem

Saïd Medjkouh, Bowen Xue, Ghouthi Boukli Hacene

IMT Atlantique, Brest, France

{said.medjkouh,bowen.xue,ghouthi.bouklihacene}@telecom-bretagne.eu

Abstract—The linear assignment problem is a fundamental combinatorial problem and a classical linear programming problem. It consists of assigning agents to tasks on a one-to-one basis, while minimizing the total assignment cost. The assignment problem appears recurrently in major applications involving optimal decision-making. However, the use of classical solving methods for large size problems is increasingly prohibitive, as it requires high computation and processing cost. In this paper, a biologically inspired algorithm using an artificial neural network (ANN) is proposed. The artificial neural network model involved in this contribution is a sparse clustered neural network (SCN), which is a generalization of the Palm-Wilshaw neural network. The presented algorithm provides a lower complexity compared to the classically used Hungarian algorithm and allows parallel computation at the cost of a fair approximation of the optimal assignment. Illustrative applications through practical examples are given for analysis and evaluation purpose.

Keywords—assignment problem, artificial neural network, hungarian algorithm.

I. INTRODUCTION

Assignment problems are essential among problems involving linear optimization as they are needed in various fields and applications that involve assigning machines to tasks, students to groups, jobs to workers, and so on. The aim is to find the optimum assignment that minimizes the total cost or maximizes the global benefit. Moreover, many seemingly different linear optimization problems can be solved as assignment problems by an accurate transformation [1]. In addition, the linear assignment problem occurs usually as a subproblem of more complex problems, such as the traveling salesman problem [2].

In addition to its theoretical importance, the assignment problem is applied in many areas ranging from military applications, such as the well-known weapon to target assignment (WTA) aiming to maximize the total expected damage done to the opponent, to economic-industry applications such as finding the optimal shipping schedule minimizing the shipment cost. Over the last few years, the linear assignment problem has received a particular attention in robotics and control theory due to applications involving task or target allocation [3].

Many algorithms have been proposed to solve this problem, most of them are based on an iterative improvement of a given global cost function [4] while the so-called Kunh Hungarian algorithm [5] was the first algorithm especially designed to solve the assignment problem given a polynomial complexity.

However, high-performance processing is needed for many of its applications, in addition to an increasing need of parallel computing. Recently, contributions have been made to accelerate the Hungarian algorithm with an efficient parallelization using the GPUs [6]. Thus, it is worth to explore more efficient methods, targeting a reduced complexity and distributed computation.

In this paper, a biologically inspired algorithm using an artificial neural network is proposed. The artificial neural network model involved in this contribution is adapted from the SCN designed by Gripon and Berrou in [7], which is a generalization of the Palm-Wilshaw neural network [8]. This algorithm has been explored for the similar Feature Correspondence problem in [9], which can be viewed, just as the assignment problem, as a graph matching problem.

The proposed algorithm provides a lower computation complexity compared to the classical Hungarian algorithm and allows parallel computation at the cost of a fair approximation of the optimal assignment. However, we do not pretend providing a complete solving algorithm. We hope that our approach will be a step forward for further research seeking biologically inspired algorithms.

This paper is organized as follows. Section II gives a brief description of the assignment problem and its combinatorial and mathematical formulation. Then, the Hungarian algorithm is described. Section III presents and analyses the proposed SCN algorithm. Section IV provides three examples of application of the assignment problem using both algorithms. Section V concludes the paper.

II. ASSIGNMENT PROBLEMS AND THE HUNGARIAN ALGORITHM

A. Assignment problems

Assignment problems describe situations where we have to find an optimal way to assign n agents to n tasks. They consist of two components: the assignment as an underlying combinatorial structure and an objective function modeling the “optimal way”.

Assume for example that we have n tasks ($j = 1, 2, \dots, n$) that need to be executed by n machines ($i = 1, 2, \dots, n$). The i th machine has a different performance regarding the j th task, for instance, the time required to perform a task depends on the machine which is assigned to it. Thus, a rating (or cost c_{ij}) is given to each machine-task assignment.

An optimal assignment is one which makes the sum of the costs (e.g., execution time) a minimum. There are $n!$ possible assignments. This corresponds to a permutation ϕ of a set $N = \{1, 2, \dots, n\}$. So, a straightforward method to solve the assignment problem is to consider all the permutations with their corresponding cost. But as the computation complexity increases terribly with n , it is not worth to consider it as a solving method.

There are different ways to model assignment problems depending on the targeted application. In the following, we give a mathematical model which represents the assignment problem by a matrix that we will call *the cost matrix*.

The following mathematical description is adopted from [10]. Mathematically an assignment is a bijective mapping of

a finite set into itself, i.e., a *permutation*. Assignments can be modeled and visualized in different ways: every permutation ϕ of the set $N \in \{1, \dots, n\}$ corresponds in a unique way to a permutation matrix $X_\phi = (x_{ij})$ with $x_{ij} = 1$ for $j = \phi(i)$ and $x_{ij} = 0$ for $j \neq \phi(i)$. We can view this matrix X_ϕ as adjacency matrix of a bipartite graph $G_\phi = (V, V'; E)$, where the node sets V and V' have n nodes, i.e., $|V| = |V'| = n$, and there is an edge $(i, j) \in E$ if $j = \phi(i)$. Thus, the assignment problem can be formulated as a graph matching problem, as shown in Figure 1, in this case $n = 4$.

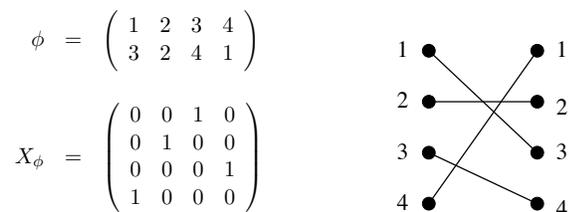


Figure 1. Different representations of assignments.

Thus, we can model the problem as follows: let c_{ij} be the cost (performance rating). A set of elements of a matrix are said to be independent if no two of them lie in the same line (the word “line” applies both to the rows and to the columns of a matrix). The goal is to choose a set of n independent elements of the *cost matrix* $C = (c_{ij})$ so that the sum for these elements is maximum or minimum depending on whether it’s a maximization or minimization problem. We assume, for this, model that the elements of C are integers. The sum of the assigned elements of C in the final assignment is its *cost*. So, the assignment permutation matrix X_ϕ is constituted by 1 in the independent assigned elements (in C) and zeros elsewhere.

The assignment problem can be expressed as the minimization of an objective function $z(\mathbf{X})$:

$$\text{minimize } z(\mathbf{X}) = \sum_{i=1}^N \sum_{j=1}^N c_{ij} x_{ij} \quad (1)$$

This minimization is subject to the *assignment constraints*:

$$\begin{aligned} \sum_{i=1}^n x_{ij} &= 1 \text{ for } j = 1, \dots, n \\ \sum_{j=1}^n x_{ij} &= 1 \text{ for } i = 1, \dots, n \\ x_{ij} &\in \{0, 1\} \text{ for all } i, j = 1, \dots, n \end{aligned} \quad (2)$$

If the cost matrix is not a square matrix, which means that the two graphs to match don’t have the same number of nodes, we can simply wrap the matrix with the needed number rows or columns of its maximum value (minimization problem) or minimum value (maximization problem). This is equivalent to assigning a worker to a fictive job for which we suppose a performance that doesn’t influence the global cost. Typical problems of this type are treated in Section IV.

B. The Hungarian algorithm

The Hungarian algorithm was developed and published in 1955 by Harold Kuhn [11], who gave the name “Hungarian method” because the algorithm was largely based on the earlier works of two Hungarian mathematicians: Dnes Knig and Jen Egervry. James Munkres reviewed the algorithm in 1957 [12] and observed that it is (strongly) polynomial [3].

The Hungarian method is an algorithm which finds an optimal assignment for a given cost matrix C . In our case, we consider the Hungarian algorithm for a minimization problem where the goal is to find the assignment which minimizes the cost.

It is worth to mention some of the mathematical foundations on which the Hungarian algorithm is based.

Theorem 1 [13]: *If a number is added to or subtracted from all of the elements of any row or column of a cost matrix C , then an optimal assignment for the resulting cost matrix is also an optimal assignment for the original cost matrix.*

Theorem 2 [14]: *If m is the maximum number of independent zero elements merits of a matrix C , then there are m lines (each line covering a column or a row) which contain all the zero elements of C .*

The algorithm consists of the six steps below. The first two steps are executed once at the beginning, while Steps 3, 4 and 5 and are repeated until an optimal assignment is found. Then, step 6 is executed to find this assignment. The input of the algorithm is an n by n cost matrix C with only positive elements.

Step 1: Subtract row minimum.

Subtract the lowest element in each row from all the elements of its row.

Step 2: Subtract column minimum.

Similarly, for each column, subtract the lowest element of each column from all the elements in that column.

Step 3: Cover all zeros with a minimum number of lines.

Cover all zeros in the resulting matrix using a minimum number of lines, each line covers one row or one column.

Step 4: Check the number of lines.

If n lines are required, an optimal assignment exist among the zeros. Go to step 6. If less than n lines are required, return to Step 3.

Step 5: Create additional zeros.

Find the smallest element that is not covered by a line in Step 3. Subtract it from all uncovered elements, and add it to all elements that are covered twice and return to step 3.

Step 6: Find the independent zeros.

Find the n independent zeros in the resulting matrix. The positions of these independent zeros correspond to the optimal assignment in the cost matrix.

III. SPARSE CLUSTERED NEURAL NETWORK ALGORITHM

In this section, we describe our ANN model and specify the details of the assignment problem-solving algorithm it implements. We also study its complexity and accuracy compared to the classical Hungarian algorithm.

A. Algorithm description

The neural network we propose for solving assignment problems is constructed, initially, on the cost matrix C . The architecture of this network is adapted from the SCN in [7], which was proposed as a generalization of Palm-Wilshaw networks [8].

The network grid structure corresponds to the 2D configuration of the cost matrix C , this means that each neuron is initialized to its corresponding value in the cost matrix C . Neuron values are represented in a matrix Y , as shown in Figure 2. As in SCNs, we impose a grouping configuration on the network neurons in the form of clusters; neurons of the same row are grouped into one cluster, and the same holds for neurons of the same column. Thus, each neuron belongs to two clusters as shown in Figure 2. Within each cluster, a winner-takes-all (WTA) activation constraint is imposed; only one neuron per cluster can be active at the end of the network activity with a binary activation level (0 or 1), which corresponds to the assignment matrix X_ϕ . However, during the network activity, and before the network matrix Y reaches its final state, this constraint is relaxed into a relaxed-winners-take-all (rWTA) constraint, this constraint is relaxed into a soft winner-takes-all where the highest value is maintained and the others are decreased. Thus, we allow the network to temporarily contain real values. Each neuron is connected to all the other neurons despite those in the same cluster, no connections exist between neurons of the same cluster as in SCNs.

The WTA and rWTA constraints we impose within clusters are meant to encourage the bijective matching constraint between the two graphs V and W .

It is worth to mention that the below description of the algorithm is for a maximization assignment problem. We can use the exact same method by, for example, multiplying the starting cost matrix by -1.

The network activity starts by assigning to each neuron its unary affinity value ($y_{ij} \leftarrow c_{ij}$). Then, within each row cluster, every neuron receives the max-pooled propagated activity of all other neurons to which it connects as shown in Figure 2.

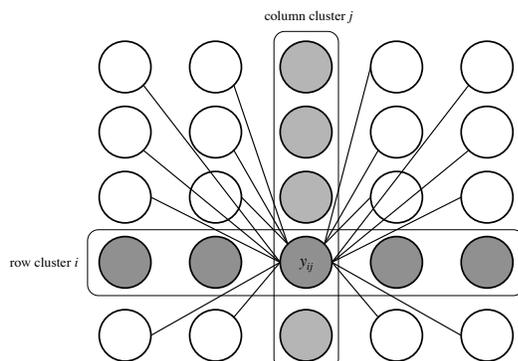


Figure 2. The architecture of the proposed neural network.

This means that each neuron receives the sum of the maximum values of each row (assuming we are processing the row clusters) of the sub-matrix constituted by all the elements of the network matrix despite those containing the

processed value. It is worth noting that the max-pooled activity is received in parallel for all neurons, as shown in (3).

$$y_{ij}^{t+1} \leftarrow \sum_{k \in A} \max_{m \in B} (y_{km}^t) \quad (3)$$

$$\text{with } A = \{1, \dots, n\} \setminus i \text{ and } B = \{1, \dots, n\} \setminus j$$

Where the superscript t denotes the current iteration. The activity values within this cluster are then normalized to their maximum. Then, an rWTA operation is applied. This means that the neurons of the same cluster, despite the maximum value of the cluster (of index (i_{Cmax}, j_{Cmax})), are penalized using a certain activation function that we note $h(\cdot)$. Thus this penalization function doesn't apply to the normalized maximum, which means that $h(1) = 1$. The aim of this penalization is to converge to a certain assignment.

$$y_{ij}^{t+1} \leftarrow h(y_{ij}^t) \quad (4)$$

$$\text{for } (i, j) \in \{1, \dots, n\}^2 \setminus (i_{Cmax}, j_{Cmax})$$

An iteration is finished when both row clusters and column clusters are alternatively processed once as up-described. Notice that for row clusters, max-pooling and rWTA are applied row-wise, while they are applied column-wise for column clusters.

Figure 3. Proposed Sparse Neural Network Algorithm.

Input : Cost matrix C
Output: Assigned matrix Y

```

1  $Y \leftarrow C$ 
2 repeat
3   foreach  $i \in \{1, \dots, N_{rows}\}$  do
4     foreach  $j \in \{1, \dots, N_{col}\}$  do
5        $y_{ij}^{t+1} \leftarrow \sum_{k \in A} \max_{m \in B} (y_{km}^t)$ 
6     end
7   end
8   foreach  $i \in \{1, \dots, N_{rows}\}$  do
9     foreach  $j \in \{1, \dots, N_{col}\}$  do
10       $y_{ij}^{t+1} \leftarrow y_{ij}^{t+1} / \max_{k \in \{1, \dots, N_{col}\}} (y_{ik}^{t+1})$ 
11       $y_{ij}^{t+1} \leftarrow h(y_{ij}^{t+1})$ 
12    end
13  end
14  foreach  $j \in \{1, \dots, N_{col}\}$  do
15    foreach  $i \in \{1, \dots, N_{rows}\}$  do
16       $y_{ij}^{t+1} \leftarrow \sum_{m \in B} \max_{k \in A} (y_{km}^t)$ 
17    end
18  end
19  foreach  $j \in \{1, \dots, N_{col}\}$  do
20    foreach  $i \in \{1, \dots, N_{rows}\}$  do
21       $y_{ij}^{t+1} \leftarrow y_{ij}^{t+1} / \max_{k \in \{1, \dots, N_{rows}\}} (y_{kj}^{t+1})$ 
22       $y_{ij}^{t+1} \leftarrow h(y_{ij}^{t+1})$ 
23    end
24  end
25 until  $Y$  converges OR last iteration attained;
```

Finally, we obtain a processed matrix with final values of y_{ij} . An activation threshold is applied, where only neurons with a maximal activation value are kept active ($y_{ij} \leftarrow 1$)

while others are deactivated ($y_{ij} \leftarrow 0$). A WTA operation is then applied within every row and column cluster; if more than one neuron is active in a given cluster, we choose randomly one neuron to activate while we deactivate the others in order to give (at least) a partial assignment, meaning that the assignment is not complete and some tasks may have not been assigned to some machines. The obtained partial-assignment is part of a fair approximation assignment.

B. Algorithm analysis

The main advantage of the proposed method is its lower complexity compared to the classical Hungarian approach. Another advantage is that our approach implements a cooperative algorithm, meaning at each neuron needs only to know about the activity of few neighboring neurons, which allows the algorithm to run in a parallel fashion.

As Munkres has shown in [12], the Hungarian algorithm solves the assignment problem in $\mathcal{O}(n^3)$ time and demonstrated that the final maximum on the number of operations needed is $(11n^3 + 12n^2 + 31n)/6$.

The proposed artificial neural network algorithm attains its final solution either by converging or after N_{iter} iterations in $\mathcal{O}(n^2)$ time. This can be done by implementing it in a slightly different way from the proposed version. In fact, in order to avoid the three nested loops when carrying the max-pooling step, we can first calculate the two maximums of each row (resp. column) keeping their index, and then process the max-pooled propagated activity. Considering an operation as one iteration in the loop, we need n^2 operations to calculate the maximums with their respective index. Then, n^2 operations to process the max-pooled propagated activity. Finally, n^2 operations to carry out the penalization. Thus, processing the rows costs $3n^2$ and $3n^2$ for the columns. All this is executed N_{iter} times. The overall number of operations is

Figure 4 shows the worst-case complexity curves of each method referenced to the total number of operations.

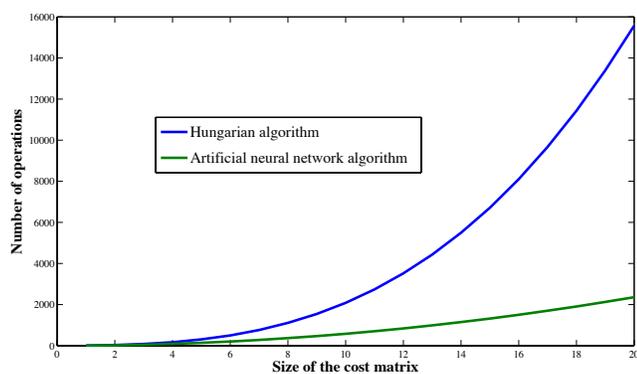


Figure 4. Worst-case complexity of the Hungarian and the proposed algorithms.

However, even though the proposed algorithm enjoys a $\mathcal{O}(n^2)$ complexity and a high parallelism level, it gives only a partial assignment of the overall approximate assignment.

In some cases, the artificial neural network algorithm gives an exact solution. For example, if the final optimal assignment corresponds to the maximum value of each row

(or column), respecting the independence constraint, we can easily notice that the solution is attained by the neural networks algorithm in one iteration. This because the activity of the most active neuron increases with the max-pool propagation while decreasing the activity of the other neurons by penalizing them. As in this case, every most active neuron is in an independent cluster, the result is directly obtained.

In a more general case, an interesting way to use the proposed algorithm is to implement it as a preprocessing before the Hungarian algorithm. This combined “ANN-Hungarian” approach, is worth to be considered. In fact, the preprocessing with the neural network algorithm of a given cost matrix C_n outputs a partially assigned matrix. This means that we can remove the assigned rows and columns and process the resulting partial sub-matrix C_p of size $p < n$ to find the rest of assignments. The overall resulting assignment X_n is a fair approximation of the optimal assignment. This approach is illustrated in Figure 5.

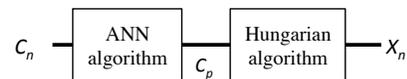


Figure 5. The proposed Artificial Neural Network used as a preprocessing for the Hungarian algorithm.

An experimental study of the accuracy of this combined method is shown in Figure 6. It has been made on a normally distributed matrix of size $n = 4$ for increasing standard deviation values, in the context of a maximization assignment problem. The accuracy is calculated as the ratio between the optimal cost and the approximate cost given by this combined approach. The used penalization function $h()$ for the ANN algorithm is a simple multiplication by a penalization factor $h(x) = 0.5x$. In this experiment, we used only one iteration.

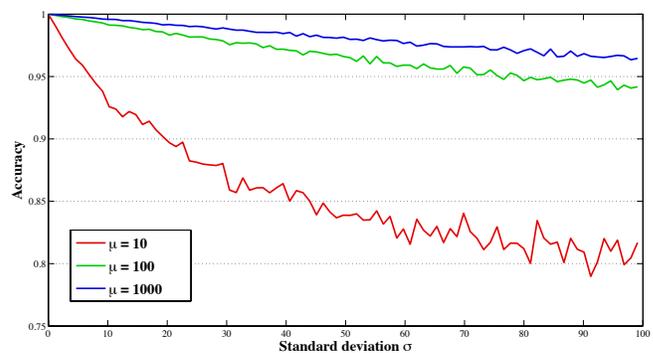


Figure 6. Experimental study of the proposed model's accuracy for randomly generated cost matrix of size $n = 4$.

From these experimental results, we can confirm that the approximation of the combined method is worth to be considered. Especially, for high values matrix because the gap between values due to the penalization function is more significant for high values. Thus, the approximation is more fairly made and the approximate assignment provides a cost close to the optimal one.

IV. APPLICATIONS

In this section, three major applications of the assignment problem are illustrated through practical examples. A comparison is made in each case between the Hungarian algorithm’s optimal cost and the result obtained by the proposed neural network approach combined to the Hungarian algorithm.

A. Military application

In military operations, problems in planning and scheduling often require feasible and close to optimal solutions with limited computing resources and within very short time periods [15]. Especially the weapons developed in contemporary technology give very less chance to defend friendly assets as enemy forces execute complex saturated attacks. Therefore, quick and efficient reactions to subsidize these attacks become very vital to survive in the combat arena. Thus, assigning the limited resources (own weapons) to the hostile targets to achieve certain tactical goals [15] becomes an important issue called as Weapon Target Assignment (WTA) problem.

Nowadays, there are more and more algorithms being applied in military affairs to design those advanced equipments. The Hungarian algorithm is also attractive enough to solve the problems of how to assign tasks to one’s own limited number of weapons and obtain a maximum gain. For example, the Hungarian algorithm has already been used for distributing missions to an array of radars. Each radar in one’s own array can emit interference signal to interfere each radar in enemy’s array, but the coefficients of these effects of interference for different enemy radars are different. So, that it needs to find the best combination to finish this interfere mission to get a maximum effect of interference. Thus, this radar assignment problem is modeled by a graph matching problem and solved with the Hungarian algorithm. In this case, as quick decisions have to be made, it can be interesting to consider using the neural networks proposed algorithm as a more time-efficient alternative as it allows a lower time complexity and a better level of parallelism.

$$\begin{pmatrix} 0.0384 & 0.3818 & 0.0165 & 0.1033 & 0.2596 & 0.2281 & 0.0229 & 0.0227 \\ 0.0181 & 0.2689 & 0.0117 & 0.0497 & 0.2818 & 0.0179 & 0.1084 & 0.1450 \\ 0.0692 & 0.2526 & 0.1153 & 0.2433 & 0.1697 & 0.1013 & 0.0705 & 0.0967 \\ 0.0261 & 0.4215 & 0.1693 & 0.0953 & 0.0316 & 0.0244 & 0.2169 & 0.0917 \\ 0.0146 & 0.1314 & 0.2029 & 0.1619 & 0.0863 & 0.2202 & 0.1170 & 0.0242 \\ 0.0187 & 0.3480 & 0.0282 & 0.0507 & 0.3440 & 0.0011 & 0.2469 & 0.1423 \\ 0.0459 & 0.3464 & 0.1236 & 0.1844 & 0.0574 & 0.1774 & 0.0493 & 0.1037 \\ 0.0352 & 0.1748 & 0.1020 & 0.0806 & 0.3111 & 0.1871 & 0.0715 & 0.0585 \\ 0.0262 & 0.2309 & 0.0026 & 0.2004 & 0.2028 & 0.1989 & 0.0394 & 0.0856 \end{pmatrix} \quad (5)$$

In one case, there are 9 radars in our own side when there are 8 in enemy’s side to interfere. According to the properties of each radar, a jamming effective matrix can be built as the matrix (5). The Hungarian algorithm aims to obtain the minimum result of the best combination. In our case, we want to obtain from this jamming effective matrix is the maximum effect of interference. Thus, the matrix can not be treated by the Hungarian algorithm directly because the matrix is not a square matrix and is not adapted for minimization version of the Hungarian algorithm previously described.

To build the final matrix, the first step is to use the maximum value 0.4215 and subtract it from all the elements of the matrix in order to build a new matrix whose size is 9×8 . The matrix needed for a minimization problem is built, with respect to Theorem 1 in Section II. The second step is to add a new column whose values are all 0.4215 so that the

in-existent radar won’t change the final result in order to have a 9×9 square matrix. The final built cost matrix is:

$$\begin{pmatrix} 0.3831 & 0.0397 & 0.4050 & 0.3182 & 0.1619 & 0.1934 & 0.3986 & 0.3988 & 0.4215 \\ 0.4034 & 0.1526 & 0.4098 & 0.3718 & 0.1397 & 0.4036 & 0.3131 & 0.2765 & 0.4215 \\ 0.3523 & 0.1689 & 0.3062 & 0.1782 & 0.2518 & 0.3202 & 0.3510 & 0.3248 & 0.4215 \\ 0.3954 & 0.0000 & 0.2522 & 0.3262 & 0.3899 & 0.3971 & 0.2046 & 0.3298 & 0.4215 \\ 0.4069 & 0.2901 & 0.2186 & 0.2596 & 0.3352 & 0.2013 & 0.3045 & 0.3973 & 0.4215 \\ 0.4028 & 0.0735 & 0.3933 & 0.3708 & 0.0775 & 0.4204 & 0.1746 & 0.2792 & 0.4215 \\ 0.3756 & 0.0751 & 0.2979 & 0.2371 & 0.3641 & 0.2441 & 0.3722 & 0.3178 & 0.4215 \\ 0.3863 & 0.2467 & 0.3195 & 0.3409 & 0.1104 & 0.2344 & 0.3500 & 0.3630 & 0.4215 \\ 0.3953 & 0.1606 & 0.4189 & 0.2211 & 0.2187 & 0.2226 & 0.3821 & 0.3359 & 0.4215 \end{pmatrix} \quad (6)$$

Finally, all of the essential requirements have been met and the assignment matrix (7) is obtained using the Hungarian algorithm. From this matrix, it can be found that the 9th radar in our side is assigned to interfere the additional in-existent radar. So, that the 9th radar will not participate in this mission.

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad (7)$$

In this example, the optimal cost obtained by the Hungarian algorithm is 1.8446 and the one obtained by the ANN-Hungarian combined approach is 1.087, which corresponds to an accuracy of 59%. This result can be explained by Figure 6. In this case, the mean of the matrix (6) is too low so that the combined method gives an average accuracy.

B. Express company application

Assignment problems have various economic-industry applications such as finding the optimal shipping schedule minimizing the shipment cost.

The assignment problem is a particular case of the well known transportation problem which deals with problems involving several product sources and several destinations of products. The assignment problem that we consider in this case is a balanced transportation problem where all supplies and demands are equal to 1. The cost considered between sources and destinations is for example the shipment time and the problem is to assign each source to one destination.

In the following example, we consider an express company in France which aims to send fast mail to capitals of other European countries every day through shipment points in different cities in France. The problem is to assign each shipment point to a destination in Europe. So, the cost of the assignment is the time of time of travel between each 2 cities. The objective is to find the assignment that minimizes this global cost. A matrix of the transport time between different cities is built as the cost matrix C as shown in the matrix (8). Where c_{ij} = time from i to j . These values have been estimated using Google map for illustrative purpose.

$$\begin{pmatrix} 70 & 95 & 105 & 70 & 195 & 110 & 120 & 190 & 95 & 135 & 205 & 130 \\ 90 & 110 & 135 & 100 & 345 & 90 & 115 & 160 & 100 & 245 & 220 & 200 \\ 95 & 200 & 255 & 115 & 360 & 75 & 100 & 160 & 120 & 265 & 330 & 235 \\ 75 & 130 & 220 & 95 & 235 & 130 & 95 & 200 & 125 & 270 & 330 & 265 \\ 90 & 145 & 240 & 105 & 320 & 110 & 75 & 190 & 140 & 285 & 355 & 275 \\ 110 & 135 & 235 & 110 & 300 & 105 & 80 & 180 & 130 & 280 & 325 & 235 \\ 105 & 235 & 135 & 120 & 350 & 85 & 245 & 280 & 235 & 270 & 380 & 275 \\ 215 & 210 & 240 & 95 & 260 & 140 & 105 & 355 & 215 & 280 & 330 & 290 \\ 190 & 175 & 195 & 75 & 155 & 230 & 140 & 315 & 85 & 235 & 295 & 210 \\ 100 & 395 & 375 & 235 & 400 & 330 & 305 & 355 & 340 & 410 & 490 & 380 \\ 90 & 335 & 420 & 255 & 520 & 355 & 350 & 440 & 305 & 455 & 515 & 405 \\ 115 & 120 & 135 & 120 & 305 & 70 & 115 & 150 & 100 & 175 & 215 & 140 \end{pmatrix} \quad (8)$$

The matrix shows how many minutes are needed to get to different destinations from different cities. For each column, the table describe the time for getting to “London”, “Berlin”, “Copenhagen”, “Amsterdam”, “Bern”, “Roma”, “Madrid”, “Athens”, “Prague”, “Stockholm”, “Moscow”, “Warsaw”. And for each rows, the table shows those principal French cities “Paris”, “Lyon”, “Marseille”, “Nantes”, “Bordeaux”, “Toulouse”, “Montpellier”, “Rennes”, “Strasbourg”, “Limoges”, “La Rochelle” and “Nice”. It can be found that when we need to send a mail to Bern, the capital of Switzerland, there is no through-flight so in such a situation, it will be faster to transport by car. The resulting optimal assignment provided by the Hungarian algorithm is shown in the following Table:

TABLE I. Optimal assignment

Paris	to	Stockholm	Lyon	to	Moscow
Marseille	to	Roma	Nantes	to	Prague
Bordeaux	to	Amsterdam	Toulouse	to	Berlin
Rennes	to	Madrid	Montpellier	to	Copenhagen
Strasbourg	to	Bern	La Rochelle	to	London
Limoges	to	Athens	Nice	to	Warsaw

The optimal cost obtained for this assignment is 1422. The cost resulting from the ANN-Hungarian combination is 1582. This corresponds to an accuracy of 90%. In this case, the combined method offers an acceptable result as the mean of the matrix (8), which represents the average time of traveling in minute, is high enough to ensure a fine accuracy.

C. Students to projects assignment

A classical problem in various areas is to assign tasks to agents. We provide the following example using a real case situation to illustrate the solving process.

The students of a university, namely Telecom Bretagne, are asked to make their semester projects preference. Each student chooses 5 projects from a range of 25 and rank them from 1 to 5 depending on his preference. There are 108 students and no more than 5 students are assigned to each project. The goal is to find a combination of students-project that maximizes the global satisfaction. This means that every student may not have its first choice, but globally all the students get a fair project assigned based on their preference. This situation is modeled by table II where **Stu** stands for student and **P** for project. The data used in this section has been provided by Telecom Bretagne project committee.

TABLE II. Choices of students

	P1	P2	P3	P4	P5	P6	...	P19	P20	21	22	...	24	25
Stu 1	4	...	4	3	3	...	5	...	2
Stu 2	4	5	2	3	1
Stu 3	5	4	2	...	1	...	3
...
Stu 108	4	3	1	5	2

This is a typical asymmetric assignment problem, meaning that the graphs to match are not of the same size and the cost matrix not a square matrix. We have 108 students to match with 25 projects. A couple of transformations are needed to model it as an assignment problem.

A matrix of 108 × 25 is built from Table II to show these 108 students different preferences. For the other projects that a student did not choose, the matrix needs to be filled with an integer of a bigger value than 5 in order not to compromise the preferences and to represent this students unwillingness. In our experiment, we choose the integer 100. Then, to embody that each project will be filled with 5 students, the columns of this matrix needs to be extended. We do this by repeating each column 5 times building 125 columns. This means that each project corresponds to 5 students. The size of the matrix is then 108 × 125, but as we need a square cost matrix, another 17 inexistent students are added with all preferences set to 100. Their affectation to a project is considered empty and will not affect the global assignment. Thus this final 125 × 125 cost matrix, as shown in the matrix (9) is processed by the solving algorithms.

$$\begin{pmatrix} 100 & 100 & 100 & 100 & 100 & \dots & 2 & 2 & 2 & 2 & 2 & 1 & 1 & 1 & 1 & 1 \\ 100 & 100 & 100 & 100 & 100 & \dots & 1 & 1 & 1 & 1 & 1 & 100 & 100 & 100 & 100 & 100 \\ 100 & 100 & 100 & 100 & 100 & \dots & 100 & 100 & 100 & 100 & 100 & 100 & 100 & 100 & 100 & 100 \\ \dots & \dots \\ 100 & 100 & 100 & 100 & 100 & \dots & 100 & 100 & 100 & 100 & 100 & 100 & 100 & 100 & 100 & 100 \end{pmatrix} \quad (9)$$

The real assignment made by the university has a cost of 181, which means that the unsatisfactoriness rate is 181 - 108 = 73. The Hungarian algorithm gives a final optimal assignment with a cost of 146, which means that the unsatisfactoriness rate is 38. The artificial neural network algorithm combined with the Hungarian algorithm gives a fair cost of 151 with an unsatisfactoriness rate of 43. The obtained results correspond to an accuracy of 96.7%. The ANN-Hungarian works better in this case because for row 109 to row 125, choices of students will all be filled with the high value 100 as for row 1 to row 108, only 25 columns represent student’s real choices of each row will not be 100. So, the mean of the matrix (9) is high (≈ 814) while the variance (≈ 75) is enough to ensure a high accuracy. Therefore, ANN-Hungarian method is a good choice to solve this problem.

V. CONCLUSION AND FUTURE WORK

In this paper, we proposed a new approach for treating assignment problems using artificial neural network. We presented the classical Hungarian algorithm to point the need for a reduced complexity and parallel computing. The proposed algorithm satisfies these needs at the cost of a partial assignment solution and a fair approximation of the cost. Even though the solution might not be complete, it is worth to consider this approach either for specific cases where final winners match with the correct assignment or seek to use it as a preprocessing with the Hungarian algorithm. Finally, we presented some examples of application of the assignment problem using both approaches. Further development of our model will include trying different penalization functions such as sigmoid functions. We shall also study the theoretical evolution of the accuracy in order to give a more precise evaluation of the advantage of using this neural network model for solving the assignment problem.

ACKNOWLEDGMENT

This work has been performed in the framework of the research minor program at Telecom Bretagne. In this context, we would like to thank Dr. Vincent Gripon for his support and contributions.

REFERENCES

- [1] C. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, ser. Dover Books on Computer Science Series. Dover Publications, 1998.
- [2] M. Held and R. M. Karp, "The traveling-salesman problem and minimum spanning trees," *Operations Research*, vol. 18, no. 6, 1970, pp. 1138–1162.
- [3] M. M. Zavlanos and G. J. Pappas, "Dynamic assignment in distributed motion planning with local coordination," *IEEE Transactions on Robotics*, vol. 24, no. 1, 2008, pp. 232–242.
- [4] M. Balinski, "Signature methods for the assignment problem," *Operations research*, vol. 33, no. 3, 1985, pp. 527–536.
- [5] M. S. Hung, "Technical Note A Polynomial Simplex Method for the Assignment Problem," *Operations Research*, vol. 31, no. 3, 1983, pp. 595–600.
- [6] K. Date and R. Nagi, "Gpu-accelerated hungarian algorithms for the linear assignment problem," *Parallel Computing*, vol. 57, 9 2016, pp. 52–72.
- [7] V. Gripon and C. Berrou, "Sparse neural networks with large learning diversity," *IEEE transactions on neural networks*, vol. 22, no. 7, 2011, pp. 1087–1096.
- [8] F. Schwenker, F. T. Sommer, and G. Palm, "Iterative retrieval of sparsely coded associative memory patterns," *Neural Networks*, vol. 9, no. 3, 1996, pp. 445–455.
- [9] A. Aboudib, V. Gripon, and G. Coppin, "A Neural Network Model for Solving the Feature Correspondence Problem," in *International Conference on Artificial Neural Networks*. Springer, 2016, pp. 439–446.
- [10] R. E. Burkard and E. Cela, "Linear assignment problems and extensions," in *Handbook of combinatorial optimization*. Springer, 1999, pp. 75–149.
- [11] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, 1955, pp. 83–97.
- [12] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the society for industrial and applied mathematics*, vol. 5, no. 1, 1957, pp. 32–38.
- [13] *Operations Research*, ser. Core business program. McGraw-Hill Education (India) Pvt Limited, 2008. [Online]. Available: <https://books.google.com.hk/books?id=Kc6y14jtkYYC> [accessed: 2017-02-10]
- [14] D. König, "Über graphen und ihre anwendung auf determinantentheorie und mengenlehre," *Mathematische Annalen*, vol. 77, no. 4, 1916, pp. 453–465.
- [15] A. Toet and H. de Waard, *The Weapon-Target Assignment Problem*. TNO Human Factors Research Institute, 1995.