1. Introduction

The Hungarian method for solving the assignment problem was developed and published in 1955 [1]. It was named the Hungarian method because two theorems by two Hungarian mathematicians [2, 3] were used. In 1957 [4], it was noticed that this algorithm is strongly polynomial and has a complexity of order $O(n^3)$. This is the reason why the Hungarian method is also known as the Kuhn-Munkres algorithm. Later on, in 1971 the complexity of the method was improved to order $O(n^4)$. A smallest uncovered element is selected to create a single zero at every iteration. This is a weakness and is alleviated by selecting more than one smallest uncovered element thus creating more than one zero at every iteration to come up with what we now call the Accelerating Hungarian (AH) method.

From the numerical illustration of the Hungarian method given in this paper, we require 6 iterations to reach optimality. It can also be shown that selecting a single smallest uncovered element (e) makes the Hungarian method inefficient when creating zeros. From the same numerical illustration of the proposed algorithm (AH) also given in this paper, it can be noted that only one iteration is required to reach optimality and that a total of six zeros are created in one iteration.

Assignment model and the Hungarian method have application in addressing the Weapon Target Assignment (WTA) problem. This is the problem of assigning weapons to targets while considering the maximum probability of kill. The assignment problem is also used in the scheduling problem of physicians and medical staff in the outpatient department of large hospitals with multi-branches. The mathematical modelling of this assignment problem results in complex problems. A hybrid meta-heuristic algorithm SCA-VNS combining a Sine Cosine Algorithm (SCA) and Variable Neighbourhood Search (VNS) based on the Iterated Hungarian algorithm is normally used.

Keywords: Hungarian method, assignment problem, König theorem, Egerváry Theorem, Kuhn-Munkres algorithm

2. Literature review and problem statement

A lot of work has been done on the Hungarian algorithm but up to now this algorithm still has an obvious weakness. A single smallest uncovered element is selected to create a single zero at every iteration. In this paper, this weakness is alleviated by selecting more than one smallest uncovered element thus creating more than one zero at every iteration to come up with what we now call the Accelerating Hungarian (AH) method.

The Accelerating Hungarian method is not a new idea. This idea has been applied to the Hungarian method before in [7]. In that paper, parallel versions of two different variants which are the classical and alternating tree of the Hungarian algorithm for solving the Linear Assignment Problem (LAP) were used. The main contribution of that paper noted was the efficient parallelization of the augmenting path search phase of the Hungarian algorithm. Computational experiments on problems with up to 25 million variables reveal that the GPU-accelerated versions are extremely efficient in solving large assignment problems, as compared to their CPU counterparts. Tremendous parallel speedups were achieved for problems with up to 400 million variables.
The Hungarian method has application in addressing the Weapon Target Assignment (WTA) problem. This is the problem of assigning weapons to targets while considering the maximum probability of kill. In [8], various model formulations, exact algorithms, and heuristic algorithms for the static and dynamic WTA were considered. The formulations used in that paper were placed into a comparable form so as to provide an insight into the developments of the defence-related WTA problems. The available solution methods were compared.

The existing Hungarian method has the weakness of assigning some jobs to dummy or pseudo machines in those cases whereby the number of jobs is more than the machines. In real life, this does not make sense since those jobs assigned to dummy machines are not done. In real life, there is a need to execute all the jobs. This purpose can be served by assigning multiple jobs to a single machine. The paper [9] proposes a Modified Hungarian Method for solving unbalanced assignment problems which gives the optimal policy of assignment of several jobs to a single machine.

The paper [10] investigated the scheduling problem of physicians and medical staff in the outpatient department of large hospitals with multi-branch. The mathematical modelling of this assignment problem resulted in a complex problem. A hybrid meta-heuristic algorithm SCA–VNS combining a Sine Cosine Algorithm (SCA) and variable neighbourhood search (VNS) based on the iterated Hungarian algorithm was used. The performance of the scheduling model obtained is better than in other compared algorithms.

The paper [11] focuses on an extension of the assignment problem with additional conflict (pair) constraints in conjunction with the assignment constraints and binary restrictions. Unlike the well-known assignment problem, this problem is NP-hard. A branch-and-bound algorithm and a Russian Doll Search algorithm using the assignment problem relaxations for lower bound computations were used. From the computational experiments done, the proposed algorithm is efficient.

In [12], The Reviewer Assignment Problem (RAP) is presented. This is one of the most important and apparent processes that involves the process of assigning a reviewer to a paper. Of the factors studied, it was observed that ‘unanimity’ is the most appropriate measure for the overall review process. The use of performance measures such as matching score, authority, coverage, and diversity is the most suitable for measuring the performance of phase one – matching paper and reviewer. It was also noticed that load balance, fairness and accuracy are suitable for measuring the performance of phase two that assigns a reviewer to a paper by satisfying constraints. The study concluded that a hybrid approach with the proper combination of these measures is the best for measuring the performance of the process of assigning a reviewer to a paper.

The paper [13] introduced the concept of the demand correlation pattern (DCP) to describe the correlation among items, based on which a new model is constructed to address the storage location assignment problem (SLAP). To solve the model, a heuristic and a simulated annealing method were developed. The proposed methods were examined and compared using both real data collected from an online retailer and numerical instances that are randomly generated.

In [14], a new graph-theoretic proof of the tropical Jacobi identity is presented. This identity was applied to optimal assignments with supervisions. That is, optimally assigning multiple tasks to one team, or daily tasks to multiple teams, where each team has a supervisor task or a supervised task.

In general, the assignment problem is formulated as given in Section 4. The variants of the assignment problems presented in sources [7–14] are obtained by modifying the constraints and objective function of Table 1.

3. The aim and objectives of the study

The aim of the study is to improve the Hungarian method for linear assignment problems so that it becomes more efficient.

To achieve the aim, the following objectives have been set:

- to increase the number of the smallest uncovered elements;
- to create more than one zero elements at each iteration;
- to illustrate the proposed algorithm by an example.

4. Assignment problem

Let any assignment problem be given by Table 1.

| c_{ij} | c_{i2} | c_{i3} | ... | c_{im} | Demand |
|--------|--------|--------|------|--------|--------|
| c_{11} | c_{21} | c_{31} | ... | c_{m1} | 1      |
| c_{12} | c_{22} | c_{32} | ... | c_{m2} | 1      |
| ...    | ...    | ...    | ... | ...    | ...    |
| c_{1n} | c_{2n} | c_{3n} | ... | c_{mn} | 1      |

Where \( c_{ij} \) is the cost of the assignment. This is a balanced problem, i.e. the number of rows (n) is equal to the number of columns (m), i.e. \( D=n=m \). The objective is to minimize the total assignment cost.

4.1. König’s theorem

In a bipartite graph \( G=(S, T; E) \), the minimum number of elements of \( S \) exposed by matching is equal to the maximum of the deficit \( h(X) \) over the subsets of \( S \) where \( h(X) = |X| - \|f(X)\| \) and \( f(X) \) denotes the set of elements of \( T \) having a neighbour in \( X \). In particular, there is a matching covering \( S \) if and only if \( \|f(X)\| \leq |X| \) holds for every subset \( X \subseteq S \).

Proof.

There are two main ways of proving this theorem. These are the constructive proof and using the linear programming primal-dual relationship. In this paper, we use the constructive proof. For any matching \( M \), we orient the edges in \( M \) from \( T \) to \( S \) and all other edges from \( S \) to \( T \). Let \( R_1 \) and \( R_2 \) denote the set of nodes of \( S \) and of \( T \), respectively that are exposed by \( M \). Let \( Z \) denote the set of nodes of the resulting directed graph which can be reached from \( R_2 \) by a directed path. If the intersection of \( R_2 \) and \( Z \) is not empty (i.e. \( R_2 \cap Z \neq \emptyset \)) then there is a path from \( R_2 \) to \( R_1 \) that alternates in \( M \) and then the symmetric difference of \( M \) and \( P \) is a matching \( M' \) with \( |M'| = |M| + 1 \). In addition, if \( R_2 \cap Z = \emptyset \) then \( L=(T\cap Z)\cup(S-Z) \) is a set of nodes covering all edges and \( |M'| = |L| \). This theorem was published in 1931 [1].
4.2. Egerváry’s theorem

In the same year of 1931, König’s theorem was extended to Egerváry’s theorem [2]. Let \( G = (S, T; E) \) be a complete bipartite graph with \( |S| = |T| \) and let \( c : E \to Z_+ \) be a nonnegative integer-valued weight function. The maximum weight of a perfect matching of \( G \) is equal to the minimum weight of a nonnegative, integer-valued, weighted-covering of \( c \) where a weighted-covering is a function \( \pi : S \cup T \to Z_+ \) for which \( \pi(u) + \pi(v) \geq c(uv) \) for every \( uv \in E \) and the weight of \( \pi \) is defined to be \( \sum \pi(v) : v \in SUT \).

Proof.

Let \( \pi \) be a nonnegative integer-valued weighted-covering of \( c \) with minimum weight. If there is a perfect matching \( M \) in the subgraph \( G_\pi \) of tight edges, where an edge \( uv \) is called tight if \( \pi(u) + \pi(v) = c(uv) \), then \( M \) is a maximum weight perfect matching of \( G \) whose weight is equal to the weight of \( \pi \). If there is no perfect matching in \( G_\pi \), then König’s theorem implies that there is a deficient set \( X \subseteq S \) in \( G_\pi \). Increase the \( \pi \)-value of each node in \( \Gamma G_\pi (X) \) by 1 and decrease the \( \pi \)-value in \( X \) by 1. Using this we obtain another weighted covering \( \pi' \) of whose weight is smaller than that of \( \pi \). In that case where \( \pi' \) is negative, increase \( \pi' \)-values on the elements of \( S \) by 1 and decrease the \( \pi' \)-values on the elements of \( T \) by 1. Since \( G \) is complete bipartite and \( c \geq 0 \), the resulting \( \pi' \) is a nonnegative weighted-cover of \( c \) whose weight is smaller than that of \( \pi \) in a contradiction with the minimum choice of \( \pi \).

4.3. Hungarian method

Hungarian algorithm was developed and published in 1953 [1]. It was named the Hungarian algorithm because it was based on the works of two Hungarian mathematicians. In 1957, it was shown that the Hungarian algorithm is strictly polynomial.

Step 1. Subtract the smallest entry in each row from all the other entries in the row. This will make the smallest entry in the row now equal to 0.

Step 2. Subtract the smallest entry in each column from all the other entries in the column. This will make the smallest entry in the column now equal to 0.

Step 3. Draw lines through the row and columns that have 0 entries such that the fewest lines possible are drawn.

Step 4. If there are \( \pi \) lines drawn, an optimal assignment of zeros is possible and the algorithm is finished. If the number of lines is less than \( \pi \), then the optimal number of zeroes is not yet reached. Go to the next Step 5.

Step 5. Find the smallest entry (\( c_{ij} \)) not covered by any line. Subtract this entry from each row that isn’t crossed out, and then add it to each column that is covered by two lines. Then, go back to Step 3.

4.3.1. Numerical illustration of the Hungarian method

Solve the following assignment problem using the Hungarian method (Table 2).

| Given assignment problem | Worker |
|--------------------------|--------|
| 10 8 3 9 24 13 1 |
| 14 24 2 32 18 12 2 |
| 44 16 19 22 15 19 3 |
| 2 2 3 1 1 1 4 |
| 31 32 4 43 28 41 5 |
| 25 62 2 29 46 22 6 |

| Job | 1 | 2 | 3 | 4 | 5 | 6 |

The Hungarian algorithm steps are presented from Table 3 to Table 17.

Table 3

| Row minima and column minima |
|------------------------------|
| 10 8 3 9 24 13 |
| 14 24 2 32 18 12 |
| 44 16 19 22 15 19 |
| 2 2 3 1 1 1 |
| 31 32 4 43 28 41 |
| 25 62 2 29 46 22 |

Table 4

| Subtracting the row minima |
|-----------------------------|
| 7 5 0 6 21 10 |
| 12 7 0 30 16 10 |
| 42 14 0 20 17 17 |
| 1 1 2 0 0 0 |
| 27 28 0 39 24 37 |
| 23 60 0 27 44 20 |

Table 5

| Subtracting the column minima |
|------------------------------|
| 6 4 0 6 21 10 |
| 11 21 0 30 16 10 |
| 41 13 0 20 17 17 |
| 0 0 2 0 0 0 |
| 26 27 0 39 24 37 |
| 22 59 0 27 44 20 |

Table 6

| Covering zeros with minimum number of lines (iteration 1) |
|--------------------------------------------------------|
| 6 4 6 21 10 |
| 11 21 30 16 10 |
| 41 13 20 17 17 |
| 26 27 39 24 37 |
| 22 59 27 44 20 |

Table 7

| Adding 4 to the elements covered by two lines and subtract 4 from the uncovered elements |
|---------------------------------------------------------------------------------------|
| 2 0 0 17 3 17 6 |
| 7 17 0 26 12 6 |
| 37 9 0 16 9 13 |
| 0 0 6 0 0 0 |
| 22 23 0 35 20 33 |
| 28 55 0 23 40 16 |

Covering zeros with minimum number of lines (iteration 2)
The number of lines \( h_3 \)=3. This implies that the solution is not optimal. The smallest uncovered element is \( e_s \)=6. We then add 6 to the elements covered by two lines and subtract 6 from all elements that are uncovered as given in Table 9.

### Table 9

| Adding 6 to the elements covered by two lines and subtracting 6 from the uncovered elements |
|---------------------------------|-----------------|-----------------|-----------------|
| 2     | 0               | 6               | 3               | 17              | 6               |
| 1     | 11              | 0               | 20              | 6               | 0               |
| 31    | 3               | 0               | 10              | 3               | 7               |
| 0     | 0               | 12              | 0               | 0               | 0               |
| 16    | 17              | 0               | 29              | 14              | 27              |
| 22    | 49              | 0               | 17              | 34              | 10              |

The number of lines \( h_3 \)=4. This implies that the solution is not optimal. The smallest uncovered element is \( e_s \)=1. We add 1 to all elements covered by two lines and subtract 1 from all elements that are uncovered as given in Table 10.

### Table 10

| Adding 1 to the elements covered by two lines and subtracting 1 from the uncovered elements |
|---------------------------------|-----------------|-----------------|-----------------|
| 1     | 0               | 6               | 3               | 17              | 6               |
| 0     | 11              | 0               | 19              | 5               | 0               |
| 30    | 3               | 0               | 9               | 2               | 7               |
| 0     | 1               | 13              | 0               | 0               | 1               |
| 15    | 17              | 0               | 28              | 13              | 27              |
| 21    | 49              | 0               | 16              | 33              | 10              |

The number of lines \( h_3 \)=4. This implies that the solution is not optimal. The smallest uncovered element is \( e_s \)=2. We add 2 to all elements covered by two lines and subtract 2 from all elements that are uncovered as given in Table 11.

### Table 11

| Adding 2 to the elements covered by two lines and subtracting 2 from the uncovered elements |
|---------------------------------|-----------------|-----------------|-----------------|
| 1     | 0               | 8               | 3               | 17              | 6               |
| 0     | 11              | 2               | 19              | 5               | 0               |
| 28    | 1               | 0               | 7               | 0               | 5               |
| 0     | 1               | 15              | 0               | 0               | 1               |
| 13    | 15              | 0               | 26              | 11              | 25              |
| 19    | 47              | 0               | 14              | 34              | 8               |

The number of lines \( h_4 \)=5. This implies that the solution is not optimal. The smallest uncovered element is \( e_s \)=8. We then add 8 to the elements covered by two lines and subtract 8 from all elements that are uncovered as given in Table 15.

### Table 15

| Adding 8 to the elements covered by two lines and subtracting 8 from the uncovered elements |
|---------------------------------|-----------------|-----------------|-----------------|
| 1     | 0               | 16              | 3               | 17              | 6               |
| 0     | 11              | 0               | 10              | 19              | 5               |
| 28    | 1               | 8               | 7               | 0               | 5               |
| 0     | 1               | 23              | 0               | 0               | 1               |
| 5     | 7               | 0               | 8               | 3               | 17              |
| 11    | 39              | 0               | 6               | 23              | 0               |

The number of lines \( h_4 \)=6. This implies that an optimal solution is available as given in Table 17.

### Table 17

| Optimal allocation |
|-------------------|-----------------|-----------------|-----------------|
| 1     | 0               | 16              | 3               | 17              | 6               |
| 0     | 11              | 10              | 19              | 5               |
| 28    | 1               | 8               | 7               | 0               |
| 0     | 1               | 23              | 0               | 0               |
| 5     | 7               | 0               | 8               | 3               |
| 11    | 39              | 0               | 6               |

The Hungarian method has many versions and the original algorithm had asymptotic complexity \( O(n^4) \) as given in [4]. Then later on it was shown to have a complexity of order \( O(n^3) \) as given in [5, 6].

### Strengths of the Hungarian method.

This method is easier to apply than the transportation simplex method. It is made up of only addition and subtraction operations and application of the optimality test. It can handle degenerate transportation problems better than the transportation simplex method.

### Weakness of the Hungarian method.

The Hungarian method has the obvious disadvantage of selecting the smallest uncovered element which may happen to be only one number. In this case, it implies one zero is created. Creating a single zero at every iteration is a weakness when the problem is large.

### 4. Accelerating Hungarian method

A way can be used to alleviate the weakness of creating a single zero at every iteration. This is done by creating more than one smallest uncovered element at every iteration. Creating more than one zero in an iteration is the same as the
accelerating Hungarian method. After covering all the zeros with the minimum number of lines it can be noticed that some rows and columns can be changed without necessarily changing the minimum number of lines. In this paper, we call such rows and columns flexible rows or columns. This important fact can be used to increase the number of the smallest uncovered elements to more than one. In order to do this, there is a need to identify the flexible rows and columns immediately after covering all the zeros. Sometimes a set of columns or rows can be flexible but cannot be changed at the same time. Such a set of rows or columns is called a flexible dependent set.

4. 4. 1. Numerical illustration 2 – flexible dependent set

Table 18 is a table of reduced cost entries with zeros covered by a minimum number of lines.

| Zeros covered with a minimum number of lines |
|-----------------|-----------------|-----------------|
| 12              | 35              | 27              |
| 31              | 28              | 23              |
| 22              | 49              | 20              |
| \(c_1\)         | \(c_2\)         | \(c_4\)         |

From the columns \(c_1\) to \(c_4\), it can be noted that three of these columns are flexible and dependent. The columns are \(c_1\), \(c_2\), \(c_4\) and these cannot all be changed at the same time. Only two columns can be changed at a time without changing the minimum number of lines covering the zeros.

4. 4. 2. Changing the flexible row or columns

Once the flexible rows and columns have been identified, then these can be changed to increase the number of the smallest uncovered elements so as to make the Hungarian method efficient. As an illustration, we use Table 19 to Table 23.

| Identifying the smallest uncovered element |
|-----------------|-----------------|-----------------|
| 12              | 35              | 27              |
| 31              | 28              | 23              |
| 22              | 49              | 20              |
| \(c_1\)         | \(c_2\)         | \(c_4\)         |

In Table 19, the only smallest uncovered element is 12. There is a need to change the flexible columns so as to increase the number of the smallest uncovered elements. To change the flexible columns, we select the smallest uncovered elements in the other two columns as given in Table 20. These numbers are 28 and 20.

| Smallest uncovered elements in the other 2 columns |
|-----------------|-----------------|-----------------|
| 12              | 35              | 27              |
| 31              | 28              | 23              |
| 22              | 49              | 20              |
| \(c_1\)         | \(c_2\)         | \(c_4\)         |

In this case, these two numbers cannot all be changed at the same time. We have to fix the column of 28 and change the columns of 12 and 20. The smallest uncovered element is 12 and columns \(c_1\) and \(c_4\) are adjusted as \((c_1+16)\) and \((c_4+8)\), respectively. The resulting cost matrix is presented in Table 21. The number of the smallest uncovered elements has increased from 1 to 3.

| Adjusted cost matrix |
|----------------------|----------------------|
| 28                   | 35                   |
| 47                   | 28                   |
| 38                   | 49                   |

This process of increasing the number of the smallest uncovered elements makes the proposed approach (AH) more powerful than the original Hungarian version. Subtracting 28 from uncovered elements and adding 28 to all elements covered by two lines we have Table 22.

| Resulting cost matrix |
|-----------------------|-----------------------|
| 0                     | 7                     |
| 19                    | 0                     |
| 13                    | 6                     |
| 21                    | 0                     |
| 0                     | 31                    |
| 10                    | 21                    |
| 0                     | 0                     |

In general, for any assignment cost matrix of the form given in Table 23, with columns \(c_1\), \(c_2\), \(c_3\), ..., \(c_n\), integers \(\lambda_1, \lambda_2, \lambda_3, ..., \lambda_n \geq 0\) can be used to change the columns to \((c_1+\lambda_1)\), \((c_2+\lambda_2)\), \((c_3+\lambda_3)\), ..., \((c_n+\lambda_n)\), such that the elements, \((c_{12}+\lambda_2)\), \((c_{23}+\lambda_3)\), ..., \((c_{3n}+\lambda_n)\) as given in Table 23 and Table 24.

| Assignment cost matrix in general |
|----------------------------------|------------------|
| \(c_{11}\)                       | \(c_{12}\)       |
| \(c_{21}\)                       | \(c_{22}\)       |
| ...                              | ...              |
| \(c_{n1}\)                       | \(c_{n2}\)       |
| \(c_{13}\)                       | \(c_{13}\)       |
| ...                              | ...              |
| \(c_{1n}\)                       | \(c_{1n}\)       |
| \(c_{23}\)                       | \(c_{23}\)       |
| ...                              | ...              |
| \(c_{nn}\)                       | \(c_{nn}\)       |

| Cost matrix after adding integers to columns |
|----------------------------------------------|------------------|
| \((c_{11}+\lambda_1)\)                     | \((c_{12}+\lambda_2)\) |
| \((c_{21}+\lambda_1)\)                     | \((c_{22}+\lambda_2)\) |
| \((c_{31}+\lambda_1)\)                     | \((c_{32}+\lambda_2)\) |
| \((c_{13}+\lambda_3)\)                     | \((c_{13}+\lambda_3)\) |
| \((c_{23}+\lambda_3)\)                     | \((c_{23}+\lambda_3)\) |
| \((c_{33}+\lambda_3)\)                     | \((c_{33}+\lambda_3)\) |
| \((c_{1n}+\lambda_n)\)                     | \((c_{1n}+\lambda_n)\) |
| ...                                           | ...              |
| \((c_{2n}+\lambda_n)\)                     | \((c_{2n}+\lambda_n)\) |
| \((c_{3n}+\lambda_n)\)                     | \((c_{3n}+\lambda_n)\) |

A new cost matrix can be created by adding constants \(\lambda_1, \lambda_2, ..., \lambda_n\) as given in Table 24 and the optimal solution does not change.

Columns of any balanced cost matrix can be changed by adding constants and this does not change the optimal solution.

4. 4. 3. Creating a column or row of only zeros

The costs in any column or any row of any balanced assignment problem can be made the same by adding constants. Suppose we select column 1 and making all entries in this column the same we have Table 25.
This important property can be exploited to solve the assignment problem. In solving assignment problems, zeros are created in rows and columns such that the total cost of the allocated cells is zero. Such a solution is optimal since all costs are nonnegative and there is no way the total cost can be less than zero.

### 4. 5. Accelerating Hungarian Method

Assuming a balanced assignment problem, the proposed Accelerating Hungarian (AH) method is summarized as follows:

1. **Step 1.** Create all zero row or column. Ensure that all other rows and columns have at least a zero.

2. **Step 2.** Cover all zeros with a minimum number of lines. If the number of lines is equal to \( n \) then an optimal solution is available. Else go to Step 3.

3. **Step 3.** Select the smallest uncovered element (\( e_{ls} \)). Identify flexible rows and columns and their minima.

4. **Step 4.** From the flexible rows, select the largest smallest minimum (\( e_{ls} \)). Adjust all other flexible rows and columns so that their smallest elements also become \( e_{ls} \).

5. **Step 5.** Subtract \( e_{ls} \) from all uncovered adjusted elements. Add \( e_{ls} \) to all elements covered by two lines and return to Step 2.

### 4.5.1. Illustration 3 – proposed Accelerating Hungarian method

Apply the AH method to the illustration given in Table 27. **Step 1.**

#### Table 27

| Creating the first column of numbers |
|-------------------------------------|
|                                     |
| 44 42 37 43 58 47                   |
| 44 54 32 62 48 42                   |
| 44 16 2 22 15 19                    |
| 44 44 45 43 43 43                   |
| 44 45 17 56 41 54                   |
| 44 81 21 48 65 41                   |

The first column of same numbers is created by (7).

\((r_1 + 34), (r_2 + 30), (r_3 + 0)\).

\((r_4 + 42), (r_5 + 11), (r_6 + 19)\). (7)

Creating the first column of zeros we have Table 28.

#### Table 28

| Creating the first column of zeros |
|-----------------------------------|
|                                    |
| 0 42 37 43 58 47                  |
| 0 54 32 62 48 42                  |
| 0 16 2 22 15 19                   |
| 0 44 45 43 43 43                   |
| 0 45 17 56 41 54                   |
| 0 81 21 48 65 41                   |

The first column of zeros is created by \((c_1 - 44)\). Ensuring that we have a zero in every column, we have Table 29.
Table 29
Ensuring at least a zero in all other columns

|   | 26 | 35 | 21 | 43 | 28 |
|---|----|----|----|----|----|
| 0 | 0  | 0  | 0  | 0  | 0  |
| 0 | 28 | 43 | 21 | 28 | 24 |
| 0 | 29 | 15 | 34 | 26 | 35 |
| 0 | 65 | 19 | 26 | 50 | 22 |

The other zeros in the other columns are created by (10).

Step 2.
Covering zeros with a minimum number of lines we have Table 30.

Table 30
Covering zeros with a minimum number of lines

|   | 26 | 35 | 21 | 43 | 28 |
|---|----|----|----|----|----|
| 0 | 0  | 0  | 0  | 0  | 0  |
| 0 | 28 | 43 | 21 | 28 | 24 |
| 0 | 29 | 15 | 34 | 26 | 35 |
| 0 | 65 | 19 | 26 | 50 | 22 |

The number of lines is $h_1=2$ and is less than 6 which implies an optimal solution is not available.

Step 3.
Columns $c_2$ to $c_6$ are flexible and dependent. The smallest uncovered element $e_{15}$ and the largest smallest minimum ($e_{19}$). The smallest uncovered number in the flexible columns is given in bold in Table 31.

Table 31
Smallest uncovered element and flexible columns

|   | 26 | 35 | 21 | 43 | 28 |
|---|----|----|----|----|----|
| 0 | 0  | 0  | 0  | 0  | 0  |
| 0 | 28 | 43 | 21 | 28 | 24 |
| 0 | 29 | 15 | 34 | 26 | 35 |
| 0 | 65 | 19 | 26 | 50 | 22 |

Step 4
The flexible columns are adjusted as $(c_2+0)$, $(c_3+11)$, $(c_5+0)$ and $(c_6+4)$. The adjusted cost matrix is given in Table 32.

Table 32
Adjusted cost matrix

|   | 26 | 46 | 26 | 43 | 32 |
|---|----|----|----|----|----|
| 0 | 0  | 0  | 20 | 0  | 17 |
| 0 | 12 | 15 | 19 | 7  | 1  |
| 26 | 0  | 11 | 5  | 0  | 4  |
| 0  | 2  | 28 | 0  | 2  | 2  |
| 0  | 3  | 0  | 13 | 0  | 13 |
| 0  | 39 | 4  | 5  | 24 | 0  |

Step 5
Subtract ($e_{19}=-26$) from all uncovered adjusted elements. Add 26 to all elements covered by two lines as given in Table 33.

Table 33
Subtracting 26 from uncovered elements and adding 26 to $c_{31}$

|   | 26 | 46 | 26 | 43 | 32 |
|---|----|----|----|----|----|
| 0 | 0  | 0  | 20 | 0  | 17 |
| 0 | 12 | 15 | 19 | 7  | 1  |
| 26 | 0  | 11 | 5  | 0  | 4  |
| 0  | 2  | 28 | 0  | 2  | 2  |
| 0  | 3  | 0  | 13 | 0  | 13 |
| 0  | 39 | 4  | 5  | 24 | 0  |

The minimum number of lines $h_2=6$, implying an optimal solution is available which is given in yellow.

6. Discussion of experimental results

From the numerical illustration of the known version of the Hungarian method given in Section 4.3.1, a single uncovered element or cell is selected to create a single new zero element. As shown in this numerical illustration, 6 iterations are required to create new zeros so as to reach optimality.

The Accelerating Hungarian method is illustrated in Section 4.5.1. In this illustration, the uncovered elements are slightly modified so that the number of the smallest elements increases to 6. As a result of this, 6 new zeros are created in just one iteration to reach optimality. The Accelerating Hungarian method has the obvious superiority of creating more zeros in one iteration than the conventional Hungarian method.

The limitation of the study is that there are no computational comparisons with other exact methods in terms of processing time to optimality. In this paper, this is considered as an area for further research.

The only disadvantage of the proposed Accelerating Hungarian method is that it requires more time in handling the many uncovered elements. This challenge can be alleviated by adding additional parallel processors for those additional smallest uncovered elements.
7. Conclusions

1. After covering the zeros with the minimum number of lines, some flexible uncovered columns can be identified. These flexible columns can be used to generate more than one smallest uncovered element.

2. More than one smallest uncovered elements results in the creation of more zeros in one iteration. More zeros mean faster Hungarian steps toward the optimal solution.

3. The proposed Accelerating Hungarian method is illustrated in Section 4.5.1. In this numerical illustration, it is shown that iterations are decreased from 6 to only one.

References

1. Kuhn, H. W. (1955). The Hungarian method for the assignment problem. Naval Research Logistics Quarterly, 2 (1-2), 83–97. doi: https://doi.org/10.1002/nav.3800020109

2. König, D. (1931). Graphok és matricák. Matematikai és Fizikai Lapok, 38, 116–119.

3. Egerváry, J. (1931). Matrixok kombinatorius tulajdonságairól. Matematikai és Fizikai Lapok, 38, 16–28.

4. Munkres, J. (1957). Algorithms for the Assignment and Transportation Problems. Journal of the Society for Industrial and Applied Mathematics, 5 (1), 32–38. doi: https://doi.org/10.1137/0105003

5. Edmonds, J., Karp, R. M. (1972). Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems. Journal of the ACM (JACM), 19 (2), 248–264. doi: https://doi.org/10.1145/321694.321699

6. Tomizawa, N. (1971). On some techniques useful for solution of transportation network problems. Networks, 1 (2), 173–194. doi: https://doi.org/10.1002/net.3230010206

7. Date, K., Nagi, R. (2016). GPU-accelerated Hungarian algorithms for the Linear Assignment Problem. Parallel Computing, 57, 52–72. doi: https://doi.org/10.1016/j.parco.2016.05.012

8. Kline, A., Ahner, D., Hill, R. (2019). The Weapon-Target Assignment Problem. Computers & Operations Research, 105, 226–236. doi: https://doi.org/10.1016/j.cor.2018.10.015

9. Rabbani, Q., Khan, A., Qaddoos, A. (2019). Modified Hungarian method for unbalanced assignment problem with multiple jobs. Applied Mathematics and Computation, 361, 493–498. doi: https://doi.org/10.1016/j.amc.2019.05.041

10. Lan, S., Fan, W., Liu, T., Yang, S. (2019). A hybrid SCA–VNS meta-heuristic based on Iterated Hungarian algorithm for physicians and medical staff scheduling problem in outpatient department of large hospitals with multiple branches. Applied Soft Computing, 85, 105813. doi: https://doi.org/10.1016/j.asoc.2019.105813

11. Öncan, T., Şuvak, Z., Akyüz, M. H., Altinel, İ. K. (2019). Assignment problem with conflicts. Computers & Operations Research, 111, 214–229. doi: https://doi.org/10.1016/j.cor.2019.07.011

12. Patil, A. H., Mahalle, P. N. (2020). Trends and Challenges in Measuring Performance of Reviewer Paper Assignment. Procedia Computer Science, 171, 709–718. doi: https://doi.org/10.1016/j.procs.2020.04.077

13. Zhang, R.-Q., Wang, M., Pan, X. (2019). New model of the storage location assignment problem considering demand correlation pattern. Computers & Industrial Engineering, 129, 210–219. doi: https://doi.org/10.1016/j.cie.2019.01.027

14. Niv, A., MacCaig, M., Sergeev, S. (2020). Optimal assignments with supervisions. Linear Algebra and Its Applications, 595, 72–100. doi: https://doi.org/10.1016/j.laa.2020.02.032