International Journal of Applied Science; Vol. 8, No. 4; 2025
ISSN 2576-7240 E-ISSN 2576-7259
https://doi.org/10.30560/ijas.v8n4p16

An Enhanced Genetic Algorithm Framework for Efficiently Solving Balanced and Unbalanced Assignment Problems

S. B. R. D. Dhananjalee[1] & E. M. U. S. B. Ekanayake[1]

[1] Department of Physical Science, Faculty of Applied Sciences, Rajarata University of Sri Lanka, Sri Lanka

Correspondence: S.B. R. D. Dhananjalee, Department of Physical Science, Faculty of Applied Sciences, Rajarata University of Sri Lanka, Mihintale, Sri Lanka. E-mail: dinushidhananjalee@gmail.com

Received: April 12, 2025; Accepted: October 24, 2025; Published: October 28, 2025

Abstract

This study proposes a genetic algorithm (GA)-based approach for solving assignment problems, aiming to minimize costs or maximize profits by determining optimal resource allocations for tasks in both balanced and unbalanced cases. Genetic algorithms, inspired by the concept of natural selection, are widely recognized for their flexibility in handling complex optimization problems. In this research, tournament selection is used to identify the best candidates based on a fitness function that minimizes total cost. A specialized crossover strategy focuses on selecting the lowest-cost assignments with the highest penalty costs. Additionally, swap mutation is applied to prevent redundant assignments. Various numerical examples are provided to illustrate the effectiveness of the proposed method in practical situations. Over time, several techniques like the Maximum Difference Cost Method, New Revised Zero's to One's Method, Bottleneck Cost Method, Modified Ant Colony Optimization Algorithm, and Matrix One's Assignment Method have been developed to handle assignment problems. Among these, the Hungarian Method, introduced in 1955, is well known for providing accurate solutions to small and medium-sized problems. However, this study introduces a novel and generalized approach using a genetic algorithm that can efficiently solve assignment problems involving any number of jobs and machines. The effectiveness of the proposed method has been validated through experiments on benchmark problems and randomly generated datasets for both balanced and unbalanced scenarios. The numerical results confirm the practical applicability of the proposed method and its potential as a powerful tool for solving real-world assignment problems.

Keywords: assignment problem, genetic algorithm, hungarian method, penalty cost, tournament selection

1. Introduction

The assignment problem is a fundamental optimization challenge where a set of tasks must be assigned to a set of resources in a way that minimizes the overall cost or maximizes efficiency. This problem arises in various real-world scenarios such as logistics, workforce allocation, and production scheduling where the goal is to optimally match tasks and resources while adhering to given constraints. Efficiently solving the assignment problem is crucial for improving performance and reducing operational costs in these applications (Rai & Khan, 2019).

The most widely recognized method for solving the assignment problem is the Hungarian method, developed by Harold Kuhn in 1955(E. M. U. S. B., 2021). This algorithm provides an exact solution by minimizing total assignment costs and is renowned for its efficiency in handling small to medium-sized problems.

The Bottleneck Cost Method (Wulan et al., 2019), New Revised Zero's to One's Method (Lakshmi, 2020), Matrix One's Assignment Method, and Maximum Difference Cost Method are recent approaches to the assignment problem, each focusing on optimizing task assignments through strategies like minimizing the largest cost, transforming the cost matrix, ensuring one-to-one assignments, and maximizing cost differences. Techniques such as the Row Minima Method, Column Minima Method, Vogel's Approximation Method, and North-West Corner Method, commonly used for solving transportation problems, can also be applied to both balanced and unbalanced assignment problems (Abdur Rashid, 2017). This is because the assignment problem is a specialized category of the transportation problem. Other existing methods include branch-and-bound techniques, heuristic methods such as the Ant Colony Optimization Algorithm (Deng et al., 2019), Particle Swarm Optimization Algorithm, and Genetic Algorithm, as well as linear programming approaches. Each of these methods is suited to specific contexts depending on the size and complexity of the problem. While these methods reliably solve many instances of the assignment problem, they often face challenges when addressing larger and more dynamic scenarios.

The assignment problem can be classified into several distinct categories, each addressing specific requirements and constraints. The Balanced Assignment Problem involves an equal number of tasks and resources, facilitating straightforward allocation. Conversely, the Unbalanced Assignment Problem arises when the number of tasks and resources is unequal, often necessitating the inclusion of dummy tasks to ensure feasibility. The Bottleneck Assignment Problem aims to minimize the maximum cost or time required to complete tasks, making it particularly relevant in scenarios with critical time constraints. The Fuzzy Assignment Problem incorporates uncertainty in cost estimation by using fuzzy numbers, making it well-suited for situations where data is imprecise or incomplete. Lastly, the Quadratic Assignment Problem focuses on assigning a set of facilities to specific locations in a manner that minimizes the total cost, which is determined by the pairwise distances and the flow between facilities. These categories highlight the complexity and versatility of the assignment problem, emphasizing the need for tailored approaches to effectively address the unique challenges posed by each type.

Traditional crossover and mutation operators do not account for penalty costs, which can result in less effective solutions. While various crossover techniques have been explored, including 1-point crossover, k-point crossover, shuffle crossover, reduced surrogate crossover, uniform crossover, average crossover (AX), discrete crossover (DC), flat crossover (FC), heuristic/intermediate crossover (HC/IC), statistics-based adaptive nonuniform crossover (SANUX), and binary crossovers, there has been limited research on integrating penalty cost calculations as a crossover operator to enhance the assignment process. To address this gap, this study introduces a novel crossover operator that incorporates penalty costs to improve solution quality. Additionally, a swap mutation strategy is employed to prevent redundant assignments and enhance population diversity.

Therefore, in this research mainly we are focusing on solving both balanced and unbalanced assignment problems by introducing modifications to the Genetic Algorithm (GA), an evolutionary algorithm inspired by the process of natural selection. The key steps of the proposed GA include generating an initial population of potential solutions, selecting the best candidates based on a fitness function that minimizes the total cost, calculating the penalty cost, identifying the lowest cost, and incorporating this information into the crossover and mutation operators to generate improved solutions. This iterative process continues until an optimal or near-optimal solution is achieved. The effectiveness of this approach is tested on both benchmarked and randomly generated assignment problems to demonstrate its efficiency and the proposed approach is particularly advantageous for addressing complex or large-scale assignment problems.

2. Literature Review

The assignment problem has been approached through various optimization techniques over the years, each offering unique strengths suited to different contexts and problem scales. One of the earliest methods is the Linear Programming (LP) Approach, introduced by Dantzig in 1951, which provides a mathematical framework for optimizing a linear objective function while adhering to specific constraints (Idriss Mohamed et al., 2015). This approach forms the basis for many exact solutions in optimization.

In 1955, the Hungarian Method was developed by Kuhn as an efficient algorithm specifically designed to solve balanced assignment problems. It provides an exact solution by minimizing assignment costs and is known for its computational efficiency with small to medium-sized instances. Another classical approach is the Greedy Algorithm, introduced by Prim in 1957, which involves making locally optimal choices at each step to quickly find approximate solutions. Although it does not guarantee global optimality, it is often used for its speed in obtaining feasible solutions.

The Branch and Bound Method, proposed by Land and Doig in 1960, systematically explores all possible solutions by creating a search tree to find the optimal assignment. Although highly effective, it can become computationally intensive for larger problems (Rahman & Islam, 2022). In contrast, Genetic Algorithms (GAs), developed by Holland in 1975, are heuristic search methods inspired by natural selection, offering flexibility and adaptability for large-scale or complex assignment problems (Younas et al., 2018). GAs seeks near-optimal solutions through iterative refinement, making them well-suited for dynamic environments.

In recent years, heuristic methods such as Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) have gained popularity in solving assignment problems (Pierobom et al., 2016). These approaches simulate natural processes to explore the solution space efficiently, enhancing solution quality and computational efficiency in complex problem instances. Additionally, machine learning techniques have been integrated with traditional optimization methods in the early 2020s, enabling models to learn from data and improve solution strategies over time. This integration facilitates adaptive, data-driven optimization for large and evolving problems.

Other noteworthy methods include the Matrix One's Assignment Method, Maximum Difference Cost Method (MDCM), Bottleneck Cost Method (BCM), New Revised Zero's to One's Method (NRZOM), Modified Ant

Colony Optimization Algorithm (ACO), and modified approaches specifically designed for solving unbalanced assignment problems. These methods provide diverse tools to address the varied requirements of assignment problems, from minimizing costs to handling imbalances and incorporating additional constraints.

3. Preliminaries

- **Genetic Algorithm**

The genetic algorithm (GA) is an optimization technique inspired by the principles of natural selection and genetics, utilizing a population-based approach to find solutions. It involves iteratively evolving a population of candidate solutions using selection, crossover, and mutation operations to find an optimal or near-optimal solution. The process begins with the choice of encoding type, which is essential for representing the solutions in a format suitable for genetic operations (Haupt, 1995).

- **Encoding Types in Genetic Algorithm**

In genetic algorithms, encoding represents solutions in a format that can be manipulated through genetic operations. The choice of encoding type affects the algorithm's efficiency in exploring the solution space. Binary encoding uses strings of 0 and 1 to represent decision variables, while octagonal encoding uses base-8 digits for a more compact representation. Hexadecimal encoding uses base-16 digits, further compressing the representation. Permutation encoding is used for problems with ordered solutions, such as the traveling salesman or assignment problems, where solutions are specific permutations of integers (Kumar, 2013).

- **Population**

In genetic algorithms, the population is a group of potential solutions, with each solution represented as an individual. This population evolves over generations to find better solutions through genetic operations.

- **String**

String refers to the representation of a potential solution (individual) in the population. It is typically a sequence of characters, numbers, or symbols that encodes the solution to the problem being solved. These strings can vary in format depending on the encoding scheme used, such as binary strings (sequences of 0s and 1s), permutation strings (ordered sets of integers), or other representations like octal or hexadecimal strings (Haupt, 1995).

- **Fitness Function**

The fitness function measures the quality of each individual solution in the population, indicating how close it is to the optimal solution. It helps determine which solutions are selected for reproduction (crossover and mutation) to improve the population over generations.

- **Selection Operator**

Selection is the process of choosing the best-performing individuals (parents) from the current population to create the next generation. This ensures that solutions with higher fitness are more likely to contribute to the offspring. Common selection methods include Tournament Selection, where a random subset of individuals is selected, and the best-performing individual within this group is chosen as a parent, and Roulette Wheel Selection, where individuals are assigned a slice of a "roulette wheel" proportional to their fitness. Individuals with higher fitness have larger slices, giving them a greater chance of selection. This process aims to strike a balance between exploiting high-quality solutions and exploring the solution space.

- **Crossover**

Crossover is a genetic operation that combines two parent solutions to create new offspring by mixing their characteristics. The idea is to exchange portions of the parent solutions' genetic material (encoding) to produce offspring that inherit traits from both parents. This mimics biological reproduction and is intended to combine the strengths of the parents, potentially creating better solutions. Common crossover techniques include single-point crossover, where a crossover point is selected and the genetic material is swapped between parents at that point, and two-point crossover, where two points are chosen, and the segments between those points are exchanged. Crossover is crucial for exploring new areas of the solution space by recombining existing good solutions in novel ways (Stoykova & Spasov, 2019).

- **Mutation**

Mutation is a genetic operation that introduces random changes to an individual’s encoding to preserve diversity within the population. By making small, random changes to an individual solution, mutation introduces new genetic material, helping to explore unexplored areas of the solution space. This process ensures that the algorithm does not become stuck in local optima and encourages innovation. Mutation can be applied to any part of an individual’s genetic code, such as flipping a bit in binary encoding, changing a value in a permutation, or modifying

a gene randomly. The mutation rate is usually kept low to balance exploration and exploitation in the algorithm (Stoykova & Spasov, 2019).

- **Offspring**

Offspring refers to the new solutions (individuals) created from the genetic operations, primarily through crossover and sometimes mutation, applied to parent solutions. After the selection process, the best-performing individuals (parents) are chosen to produce offspring by combining their genetic material. The offspring inherit characteristics from both parents, potentially with some random alterations (mutation). These new individuals then enter the next generation, where they continue to evolve through further selection, crossover, and mutation.

4. Methods

To start the Methods section for solving the Assignment Problem using Genetic Algorithm (GA), it's important to first provide the mathematical formulation of the Assignment Problem (AP). This lays a foundation for understanding the problem structure and the constraints that the GA aims to satisfy.

4.1 Mathematical Formulation of the Assignment Problem

The Assignment Problem is a combinatorial optimization problem where a set of n tasks needs to be assigned to n resources in a way that minimizes the total cost or maximizes the total efficiency. The problem is typically represented as a cost matrix C = $[C_{ij}]$, where C_{ij} denotes the cost of assigning task i to resource j. The objective is to find an assignment that minimizes the total cost, subject to the constraints that each task is assigned to exactly one resource, and each resource is assigned to exactly one task.

Let X_{ij} be a binary decision variable such that:

$$X_{ij} = \begin{cases} 1, & \text{if task i is assigned to resource j} \\ 0, & \text{otherwise} \end{cases}$$

Minimize Z = $\sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij} X_{ij}$

subject to the constraints;

$$\sum_{i=1}^{n} X_{ij} = 1; \text{ j} = 1, 2, \ldots, \text{n}$$

$$\sum_{j=1}^{n} X_{ij} = 1; \text{ i} = 1, 2, \ldots, \text{n}$$

4.2 Methodology for Solving Balanced Assignment Problem

Step 1: Binary encoding was selected as the representation method for solutions in the optimization problem. We use '0' could represent the absence of assignment, indicating that a particular resource is not assigned to a task and '1' could represent the presence of assignment, indicating that a particular resource is assigned to a task.

Step 2: Generate n random initial solutions for an n×n assignment matrix. Two of these solutions should include allocations along the two diagonals of the matrix. The remaining n – 2 solutions should include different allocations of the first row. When constructing these n – 2 solutions, if an element from the first row is selected, the corresponding column is prohibited for further selection. For the second row, the lowest available element is chosen, and this process continues row by row, always selecting the lowest value from the remaining elements.

Step 3: Using tournament selection for select the best out of these to become a parent. The individuals with the lowest total cost, representing higher fitness values, are chosen as parents. This process is iterated to select additional parents for generating offspring in subsequent generations.

The fitness function for the Assignment Problem can be mathematically expressed as:

$$\text{F(x)} = \sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij} X_{ij}$$

where:

n = Number of tasks (or resources).

C_{ij} = The cost of assigning task i to resource j.

$$X_{ij} = \begin{cases} 1, & \text{if task i is assigned to resource j} \\ 0, & \text{otherwise} \end{cases}$$

Step 4: For each row i, the row-wise arithmetic average (M_i) is calculated by summing all the cost values in that row and dividing by the number of resources. This is expressed mathematically as:

$$M_i = \frac{\sum_{j=1}^{n} C_{ij}}{n} \quad (1)$$

Step 5: For each row j, the column-wise arithmetic average (Mj) is calculated by summing all the cost values in that column and dividing by the number of tasks. This is expressed mathematically as:

$$M_j = \frac{\sum_{i=1}^{n} C_{ij}}{n} \quad (2)$$

Step 6: The penalty cost for each row and column is determined by calculating the absolute difference between the maximum and minimum cost values relative to the arithmetic average for that row or column. For a row i, the penalty cost is given by:

$$D_i = |\alpha_i - \beta_i| \quad (3)$$

where α_i is the minimum value among the cost values C_{ij} that are greater than the arithmetic average of the ith row, and β_i is the maximum value among the cost values C_{ij} that are less than the arithmetic average of the ith row. Similarly, for a column j, the penalty cost is calculated as:

$$D_j = |\alpha_j - \beta_j| \quad (4)$$

Similarly, where α_j is the minimum value among the cost values C_{ij} that are greater than the arithmetic average of the jth column, and β_j is the maximum value among the cost values C_{ij} that are less than the arithmetic average of the jth column. The Excel functions are;

=MIN(IF(A1:P12>=AVERAGE(A1:P12), A1:P12)) & =MAX(IF(A1:P12<=AVERAGE(A1:P12), A1:P12)) can be used to calculate the values of α_i and β_i , respectively. Similarly, we can find α_j and β_j too.

Step 7: Apply crossover techniques to generate new solutions by selecting the lowest cost with the highest penalty cost (If there is a tie in penalty costs, break the tie by selecting the cell with the lowest cost, and if ties still remain, choose the cell with the largest corresponding arithmetic average, and if all options are identical, select the cell at random).

Step 8: To prevent identical allocations in offspring, which is not allowed in an assignment problem, swap mutation is used. We identify which resource is not allocated for a particular task, then select two positions on the chromosome at random and interchange their values to allocate that resource to the task.

Step 9: Calculate the total cost associated with the new offspring. If the solution is better than before, these new offspring will be parents in the next generation. These steps are repeated until the optimum value is obtained.

4.3 Methodology for Solving Unbalanced Assignment Problem

Step 1: Binary encoding was selected as the representation method for solutions in the optimization problem. We use '0' could represent the absence of assignment, indicating that a particular resource is not assigned to a task and '1' could represent the presence of assignment, indicating that a particular resource is assigned to a task.

Step 2: To convert the unbalanced assignment problem into a balanced one, we add a dummy row or dummy column to the assignment matrix. This dummy row/column is filled with zero costs, ensuring that the number of tasks matches the number of resources.

Step 3: Generate n random initial solutions for an n×n assignment matrix. Two of these solutions should include allocations along the two diagonals of the matrix. The remaining n – 2 solutions should include different allocations of the first row. When constructing these n – 2 solutions, if an element from the first row is selected, the corresponding column is prohibited for further selection. For the second row, the lowest available element is chosen, and this process continues row by row, always selecting the lowest value from the remaining elements.

Step 4: Using tournament selection for select the best out of these to become a parent. The individuals with the lowest total cost, representing higher fitness values, are chosen as parents. This process is iterated to select additional parents for generating offspring in subsequent generations.

The fitness function for the Assignment Problem can be mathematically expressed as:

$$F(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij} X_{ij}$$

where:

n = Number of tasks (or resources).

C_{ij} = The cost of assigning task i to resource j.

$$X_{ij} = \begin{cases} 1, & \text{if task i is assigned to resource j} \\ 0, & \text{otherwise} \end{cases}$$

Step 5: Calculate the row-wise arithmetic average (Mi) for each actual row in the assignment matrix without including any dummy rows or columns, the formula is:

$$M_i = \frac{\sum_{j=1}^{n-m_2} c_{ij}}{n-m_2} \quad (5)$$

Where m_2 represents dummy columns.

Step 6: Calculate the column-wise arithmetic average (M_j) without considering the dummy row or dummy columns, the formula can be adjusted as:

$$M_j = \frac{\sum_{i=1}^{n-m_1} c_{ij}}{n-m_1} \quad (6)$$

Where m_1 represents dummy rows.

Step 7: To calculate the penalty cost for each row (D_i) without considering the zero-cost elements in a dummy row or column, the formula can be modified as follows:

$$D_i = |\alpha_i - \beta_i| \quad (7)$$

Similarly, for a column j, the penalty cost (D_j) is calculated as:

$$D_j = |\alpha_j - \beta_j| \quad (8)$$

Here, α_i, α_j, β_i and β_j are defined as in Step 06 of the algorithm for the balanced assignment problem, excluding zero-cost elements in the dummy row and dummy column.

Step 8: Crossover techniques are applied iteratively to generate new solutions, prioritizing the lowest-cost solution (ignoring zero elements in the dummy row or column) with the highest penalty cost, until only a dummy element and one valid task-resource pair remain((If there is a tie in penalty costs, break the tie by selecting the cell with the lowest cost, and if ties still remain, choose the cell with the largest corresponding arithmetic average, and if all options are identical, select the cell at random).

Step 9: To prevent identical allocations in offspring, which is not allowed in an assignment problem, swap mutation is used. We identify which resource is not allocated for a particular task, then select two positions on the chromosome at random and interchange their values to allocate that resource to the task.

Step 10: Calculate the total cost associated with the new offspring. These steps are repeated until the optimum value is obtained.

5. Results and Discussion

5.1 Results

Example 1:

To illustrate the application of the proposed method, consider an example where five jobs need to be assigned to five machines. The objective is to minimize the total cost of assignment while ensuring each job is assigned to exactly one machine, and each machine is assigned exactly one job (E. M. U. S. B., 2021). The cost matrix representing the assignment problem is provided below, and the solution process follows the proposed method step by step to achieve the optimal allocation.

Table 1. Initial Balanced Assignment Problem

	P	Q	R	S	T
A	12	8	7	15	4
B	7	9	1	14	10
C	9	6	12	6	7
D	7	6	14	6	10
E	9	6	12	10	6

Step 01: The solution is represented using binary encoding, where 1 indicates a task is assigned to a machine, and 0 indicates it is not.

Step 02: For the 5×5 assignment matrix, five initial population solutions are generated. Two solutions assign tasks along the diagonals, while the other three are constructed by selecting the lowest available costs row by row, ensuring unique allocations.

Table 2. Initial Population Selection

String No	Initial Population					Total Cost
1	00001	00010	00100	01000	10000	4+14+12+6+9 = 45
2	10000	01000	00100	00010	00001	12+9+12+6+6 = 45
3	01000	00100	00010	10000	00001	8+1+6+7+6 = 28
4	00100	10000	01000	00010	00001	7+7+6+6+6 = 32
5	00010	00100	01000	10000	00001	15+1+6+7+6 = 35

Step 03: Solutions 3 and 4, having the lowest total costs. It demonstrates the highest fitness values. As a result, these solutions are selected as parents for the next generation through the tournament selection process.

Parent 1 : 01000 00100 00010 10000 00001

Cost : (8) (1) (6) (7) (6)

Parent 2 : 00100 10000 01000 00010 00001

Cost : (7) (7) (6) (6) (6)

In Steps 04, 05, and 06, the row-wise and column-wise arithmetic averages are calculated, and the penalty costs are determined by considering the difference between the nearest maximum and minimum values relative to these averages.

Table 3: Calculation of Penalty Costs

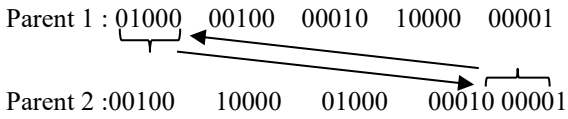
	P	Q	R	S	T	M_1	D_1
A	12	8	7	15	4	9.2	4
B	7	9	1	14	10	8.2	2
C	9	6	12	6	7	8	2
D	7	6	14	6	10	8.6	3
E	9	6	12	10	6	8.6	3
M_1	8.8	7	9.2	10.2	7.4		
D_1	2	2	5	4	3		

The third column contains the highest difference, so the corresponding lowest cost value is initially selected. However, this value has already been assigned in Parent 1. Therefore, the corresponding row and column are excluded, and the arithmetic averages and penalty costs are recalculated to perform the crossover again.

Table 4. Calculation of Penalty Costs

	P	Q	R	S	T	D_1	M_2	D_2
A	12	8	7	15	4	4	9.75	4
B	7	9	1	14	10	2	-	-
C	9	6	12	6	7	2	7	0
D	7	6	14	6	10	3	7.25	3
E	9	6	12	10	6	3	7.75	3
D_1	2	2	5	4	3			
M_2	9.25	6.5	-	9.25	6.75			
D_2	3	2	-	4	1			

The first row and the fourth column contain the maximum differences, and the corresponding lowest cost is 4. Therefore, in step 07 this value is selected to perform the crossover between Parent 1 and Parent 2.



After the crossover process, the resulting offspring (O.S) represent the new solutions with task-resource allocations based on the lowest-cost values and adherence to constraints.

Off Spring 1 : 00001 00100 00010 10000 00001

Off Spring 2 : 00100 10000 01000 00010 01000

In step 08, to avoid identical allocations in the offspring 1, we will use the swap mutation operator.

Off Spring 1 : 00001 00100 00010 10000 00001

After interchanging the binary digits, the newly generated offspring 1 is represented with updated task-resource allocations.

New Off Spring 1 : 00001 00100 00010 10000 01000

Similarly, to avoid identical allocations in the offspring 2, we will use the swap mutation operator again.

Off Spring 2 : 00100 10000 01000 00010 01000

After interchanging the binary digits, the newly generated offspring 2 is represented with updated task-resource allocations.

New Off Spring 2 : 00100 10000 00001 00010 01000

In Step 09, Compute the total cost associated with the newly generated offspring by summing the costs corresponding to the selected task-resource pairs.

Table 5. Total Cost Calculation for New Offspring

String No	New Off Spring					Total Cost
1	00001	00100	00010	10000	01000	4+1+6+7+6 = 24
2	00100	10000	00001	00010	01000	7+7+7+6+6 = 33

The transition from the first generation to the second generation yields a better solution compared to the initial generation. With this improvement, the process concludes, reaching the optimal solution.

Table 6. Optimal Solution Achievement

Assignment	Cost
A to T	4
B to R	1
C to S	6
D to P	7
E to Q	6

Therefore, after the allocation;

Total minimum cost = 4+1+6+7+6 = 24

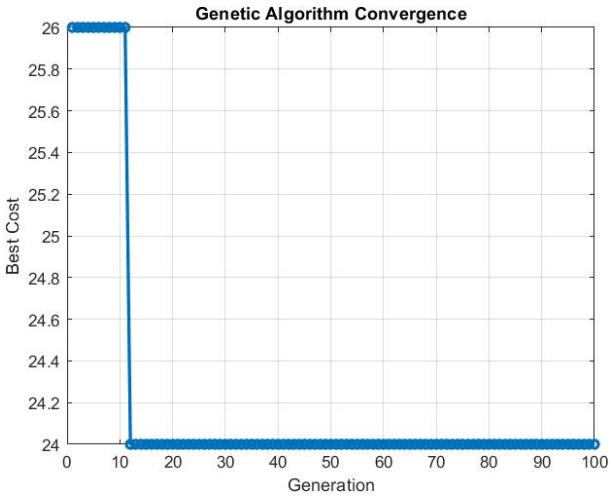


Figure 1. MATLAB Results for Example 1 Reaching Optimum Solution Using 100 Iterations

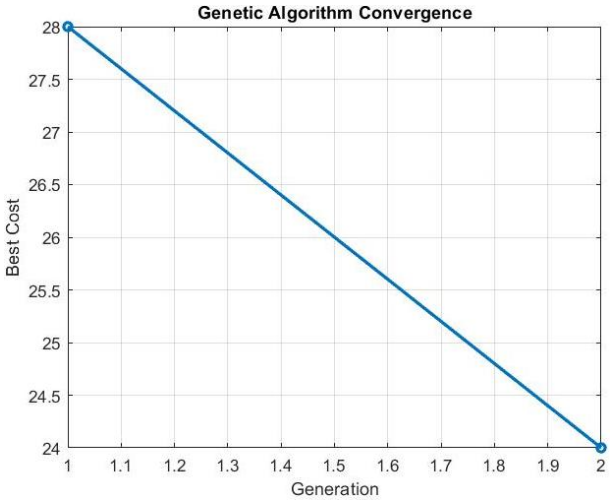


Figure 2. MATLAB Results for Example 01 Reaching Optimum Solution Using Proposed Method

Example 2:

To illustrate the unbalanced assignment problem, consider a scenario where three jobs need to be assigned to four machines. The problem is approached directly using the proposed method to determine the optimal solution for this unbalanced case.

Table 7. Initial Unbalanced Assignment Problem

	$Machine_1$	$Machine_2$	$Machine_3$	$Machine_4$
Job_1	160	220	240	200
Job_2	100	320	260	160
Job_3	100	200	460	250

Step 01: The solution is represented using binary encoding, where 1 indicates a task is assigned to a machine, and 0 indicates it is not.

Step 02: Since the total number of columns exceeds the total number of rows, the assignment problem is unbalanced. To convert it into a balanced assignment problem (with an equal number of rows and columns), a dummy row with zero costs is added to the cost matrix.

Table 8. Initial Unbalanced Assignment Problem

	$Machine_1$	$Machine_2$	$Machine_3$	$Machine_4$
Job_1	160	220	240	200
Job_2	100	320	260	160
Job_3	100	200	460	250

Step 03: For the 4×4 assignment matrix, four initial population solutions are generated. Two solutions assign tasks along the diagonals, while the other two are constructed by selecting the lowest available costs row by row, ensuring unique allocations.

Table 9: Initial Population Selection

String No	Initial Population	Total Cost
1	0001 0010 0100 1000	200+260+200+0 = 660
2	1000 0100 0010 0001	160+320+460+0 = 940
3	0100 1000 0001 0010	220+100+250+0 = 570
4	0010 1000 0100 0001	240+100+200+0 = 540

Step 04: Solutions 3 and 4, having the lowest total costs. It demonstrates the highest fitness values. As a result, these solutions are selected as parents for the next generation through the tournament selection process.

Parent 1 : 0100 1000 0001 0010

Cost : (220) (100) (250) (0)

Parent 2 : 0010 1000 0100 0001

Cost : (240) (100) (200) (0)

In Steps 05, 06, and 07, the row-wise and column-wise arithmetic averages are calculated without considering the zero-cost elements in the dummy row. The penalty costs are then determined by evaluating the difference between the nearest maximum and minimum values relative to these averages, ensuring that zeros in the dummy row are excluded from the calculations.

Table 10. Calculation of Penalty Costs

	M_1	M_2	M_3	M_4	M_1	D_1
J_1	160	220	240	200	205	20
J_2	100	320	260	160	210	100
J_3	100	200	460	250	252.5	210
Dummy	0	0	0	0	-	-
M_1	120	246.7	320	203.3		
D_1	60	100	200	50		

The third row contains the highest difference, and the corresponding lowest cost value is 100. Therefore, in step 08, this value is chosen to perform the crossover between Parent 1 and Parent 2.

Parent 1 : 0100 1000 0001 0010

Parent 2 : 0010 1000 0100 0001

After the crossover process, the resulting offspring (O.S) represent the new solutions with task-resource allocations based on the lowest-cost values and adherence to constraints.

Off Spring 1 : 0100 1000 1000 0010

Off Spring 2 : 0010 0001 0100 0001

In step 09, to avoid identical allocations in the offspring 1, we will use the swap mutation operator.

Off Spring 1 : 0100 1000 1000 0010

After interchanging the binary digits, the newly generated offspring 1 is represented with updated task-resource allocations.

New Off Spring 1 : 0100 0001 1000 0010

Similarly, to avoid identical allocations in the offspring 2, we will use the swap mutation operator again.

Off Spring 2 : 0010 0001 0100 0001

After interchanging the binary digits, the newly generated offspring 2 is represented with updated task-resource allocations.

New Off Spring 2 : 0010 0001 0100 1000

In Step 10, Compute the total cost associated with the newly generated offspring by summing the costs corresponding to the selected task-resource pairs.

Table 11. Total Cost Calculation for New Offspring

String No	New Off Spring				Total Cost
1	0100	0001	1000	0010	220+160+100+0 = 480
2	0010	0001	0100	1000	240+160+200+0 = 600

The transition from the first generation to the second generation yields a better solution compared to the initial generation. With this improvement, the process concludes, reaching the optimal solution.

Table 12. Optimal Solution Achievement

Assignment	Cost
Job1 to Machine2	220
Job2 to Machine4	160
Job3 to Machine1	100
Job4 to Dummy	0

Therefore, after the allocation;

Total minimum cost = 220+160+100+0 = 480

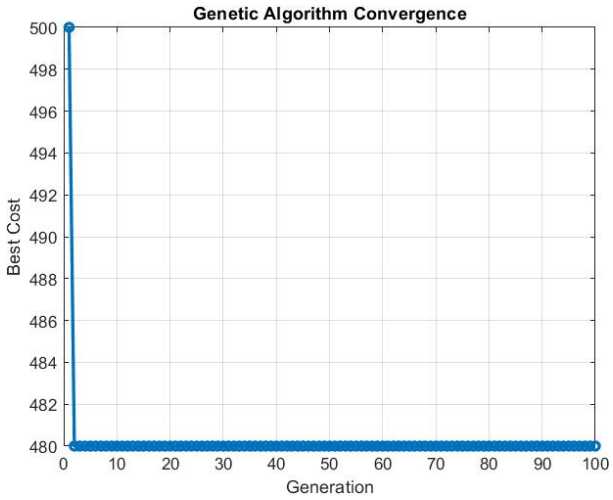


Figure 3. MATLAB Results for Example 2 Reaching Optimum Solution Using 100 Iterations

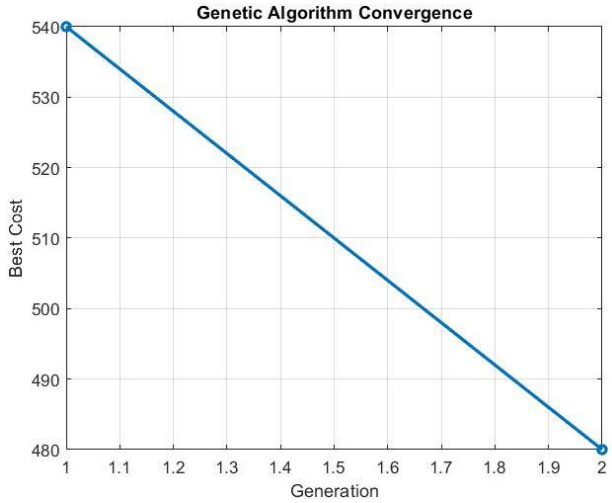


Figure 4. MATLAB Results for Example 2 Reaching Optimum Solution Using Proposed Method

5.2 Discussion

This section illustrates the results obtained from solving both benchmark and randomly generated balanced and unbalanced assignment problems. The proposed modified genetic algorithm approach is compared with the optimal solutions and several existing methods, including the Row Minima Method (RMM), Column Minima Method (CMM), North West Corner Method (NWCM), Least Cost Method (LCM), and Vogel's Approximation Method (VAM).

5.2.1 Benchmark Problem Results

Tables 13 and 14 present the results obtained for several numerical examples of both balanced and unbalanced assignment problems, respectively. These results demonstrate the application and effectiveness of the proposed modified genetic algorithm. The cost matrix data are shown in Appendix A and B, respectively.

Table 13. Comparison of Solutions for Balanced Assignment Problems Using Different Methods

Pr-No	NWCM	LCM	VAM	RMM	CMM	HM	New
1	95	41	41	50	41	41	41
2	31	31	31	31	37	31	31
3	94	56	38	47	55	38	38
4	78	77	72	78	88	72	73
5	18	13	13	14	16	13	13
6	45	26	24	26	32	24	24
7	31	15	9	9	12	9	9
8	408	411	401	404	402	399	400
9	30	25	24	25	30	24	24
10	64	41	41	53	41	41	41
11	330	245	220	220	250	215	220
12	83	60	60	72	60	60	60
13	21	19	19	21	22	19	19
14	26	12	12	12	13	12	12

Figure 5 illustrates the comparison between the solution obtained using the new method and the optimal solution for various balanced assignment problems.

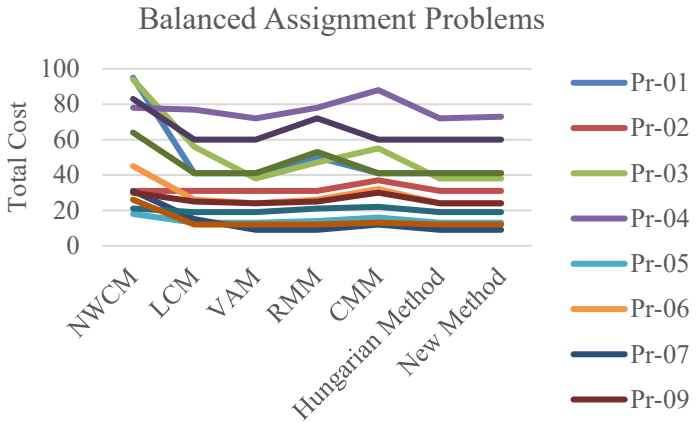


Figure 5. Graphical Comparison of Balanced Assignment Problem Solutions

Table 14. Comparison of Solutions for Unbalanced Assignment Problems Using Different Methods

Problem No	NWCM	LCM	VAM	RMM	CMM	HM	New
1	23	21	16	23	21	14	14
2	16	13	8	10	9	8	8
3	16	8	4	8	4	4	4
4	16	13	9	12	10	9	9
5	51	35	35	44	35	35	35
6	22	19	15	24	15	15	15
7	20	20	16	16	16	16	16
8	16	16	15	16	16	15	15
9	36	36	27	27	27	27	27
10	235	235	238	219	277	219	219

Figure 6 illustrates the comparison between the solution obtained using the new method and the optimal solution for various unbalanced assignment problems.

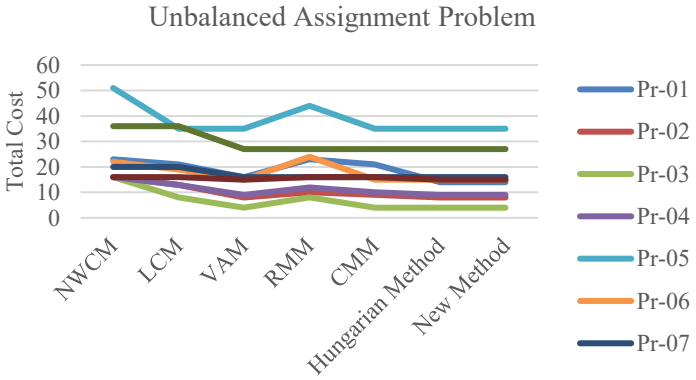


Figure 6. Graphical Comparison of Unbalanced Assignment Problem Solutions

5.2.2 Randomly Generated Problem Results

Tables 15 and 16 present the results obtained for several randomly generated examples of both balanced and unbalanced assignment problems, respectively. These results demonstrate the application and effectiveness of the proposed modified genetic algorithm. The cost matrix data are shown in Appendix C and D, respectively.

Table 15. Comparison of Solutions for Randomly Generated Balanced Assignment Problems Using Different Methods

Problem No	NWCM	LCM	VAM	RMM	CMM	HM	New
1	21	15	12	15	15	12	12
2	38	15	14	14	20	14	14
3	39	23	18	21	31	18	18
4	56	28	17	21	17	15	15
5	197	91	71	70	70	70	70
6	2368	1732	1732	2476	2605	1405	1405
7	765	264	314	254	212	212	212
8	5901	2711	3065	3233	3320	2711	2711
9	130	44	50	44	62	44	44
10	354	160	147	160	199	147	147

Figure 7 illustrates the comparison between the solution obtained using the new method and the optimal solution for various randomly generated balanced assignment problems.

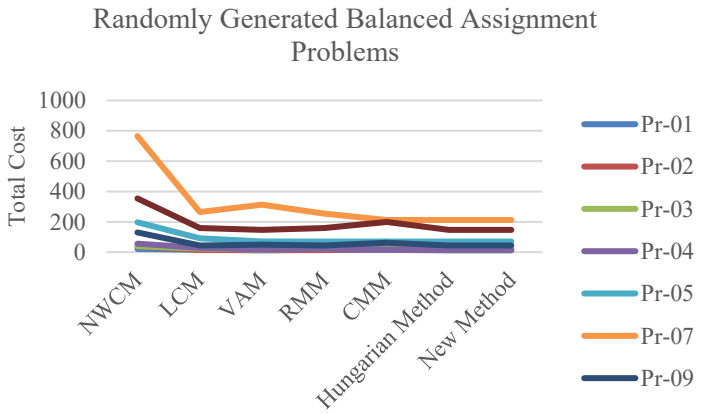


Figure 7. Graphical Comparison of Randomly Generated Balanced Assignment Problem Solutions

Table 16. Comparison of Solutions for Randomly Generated Unbalanced Assignment Problems Using Different Methods

Problem No	NWCM	LCM	VAM	RMM	CMM	HM	New
1	9	9	9	9	9	9	9
2	42	36	29	39	29	29	29
3	43	24	17	31	16	15	15
4	57	21	17	18	21	17	17
5	101	42	44	52	70	42	42
6	39	14	12	27	14	11	11
7	68	21	25	25	28	21	21
8	1785	1675	1299	1196	1159	1159	1159
9	184	146	104	87	220	87	87
10	20	14	12	12	19	12	12
11	128	80	80	98	80	80	80

Figure 8 shows the comparison between the solution obtained using the new method and the optimal solution for various randomly generated unbalanced assignment problems.

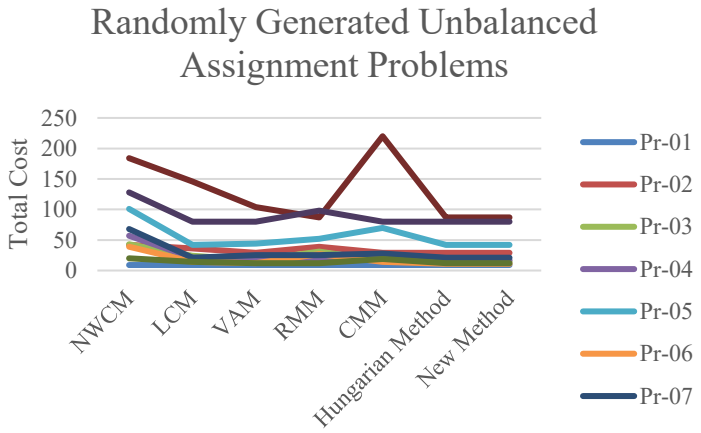


Figure 8. Graphical Comparison of Randomly Generated Unbalanced Assignment Problem Solutions

6. Conclusions

In conclusion, solving assignment problems is a fundamental task in various fields, with the Hungarian Algorithm being a widely recognized approach for achieving optimal solutions. However, this study explores the potential of Genetic Algorithms as an innovative and adaptable alternative, particularly for complex or large-scale problems. Genetic Algorithms are characterized by their iterative nature, inspired by the process of natural selection, which enables continuous improvement of solutions across generations. This study specifically employed a crossover strategy focused on selecting the lowest cost corresponding to the largest penalty difference and used swap mutation operators to optimize the exploration of solutions while avoiding identical outcomes. These techniques allowed for efficient progression toward optimal or near-optimal solutions. The proposed method was tested on examples of both balanced and unbalanced assignment problems, demonstrating its flexibility and effectiveness. In unbalanced scenarios, dummy rows or columns were added to balance the cost matrix, and the Genetic Algorithm seamlessly adapted to generate high-quality solutions. The proposed method has been validated through rigorous numerical examples, showing its capacity to deliver optimal or near-optimal solutions for both benchmark problems and randomly generated assignment problems. Comparisons with existing methods and optimal solutions demonstrate the superior accuracy and efficiency of the modified Genetic Algorithm. An iterative refinement process highlights the capability of Genetic Algorithms to handle a wide range of assignment problem variants effectively. While traditional methods like the Hungarian Algorithm are valuable and reliable, the adaptability and performance of Genetic Algorithms suggest their potential as a promising alternative for modern optimization challenges. The results of this study underscore the advantages of using Genetic Algorithms for solving assignment problems and provide a foundation for future research and applications in complex decision-making processes.

References

Abdur Rashid. (2017). *An alternative approach for solving unbalanced assignment problems. Jahangirnagar University Journal of Science, 40*(2), 1–12.

Afroz, H. D., & Hossen, D. M. A. (2017). *New proposed method for solving assignment problem and comparative study with the existing methods. IOSR Journal of Mathematics, 13*(2), 84–88. https://doi.org/10.9790/5728-1302048488

Amponsah, S. K., Otoo, D., Salhi, S., & Quayson, E. (2016). *Proposed heuristic method for solving assignment problems. American Journal of Operations Research, 6*(6), 436–441. https://doi.org/10.4236/ajor.2016.66040

Deng, W., Xu, J., & Zhao, H. (2019). *An improved ant colony optimization algorithm based on hybrid strategies for scheduling problem. IEEE Access, 7*, 20281–20292. https://doi.org/10.1109/ACCESS.2019.2897580

Dil Afroz, H. (2023). *An alternative proposed system for solving assignment problems. Quest Journals Journal of Research in Applied Mathematics, 9*(1), 2394–0735. www.questjournals.org

Dil, H., & Anwar, M. (2024). *Divide column and subtract one assignment method for solving assignment problem.* January, 289–297. https://doi.org/10.9790/5728-1705010107

Ekanayake, E. M. U. S. B. (2021). *Novel method for solving balance and unbalance assignment problem using new approach for ant colony algorithm. Mathematical Modelling and Applications, 6*(4), 101. https://doi.org/10.11648/j.mma.20210604.13

Hashim, Z. K., & Shiker, M. A. K. (2023). *A new technique to solve the assignment problems. AIP Conference Proceedings, 2414*(February 2023). https://doi.org/10.1063/5.0114724

Haupt, R. L. (1995). *An introduction to genetic algorithms for electromagnetics. IEEE Antennas and Propagation Magazine, 37*(2), 7–15. https://doi.org/10.1109/74.382334

Idriss Mohamed, E., Mahjoub Mohammed, E., Mohamed Idriss, El., & Mohamed Hussein, E. (2015). *Application of linear programming (assignment model). International Journal of Science and Research, 4*(March), 3–7. www.ijsr.net

Khandelwal, A. (2018). *An amalgamated approach for solving unbalanced assignment problem. Malaya Journal of Matematik, 6*(2), 321–325. https://doi.org/10.26637/MJM0602/0003

Kore, G. B. (2012). *A new approach to solve an unbalanced assignment problem. International Journal of Physics and Mathematical Sciences, 2*(1), 46–55. http://www.cibtech.org/jpms.htm

Kumar, A. (2013). *Encoding schemes in genetic algorithm. Kurukshetra International Journal of Advanced Research in IT and Engineering, 2*(3), 1–7. www.garph.co.uk

Lakshmi, V. T. (2020). *A new revised zero's to one's method for solving assignment problem.* June, 2–7. https://doi.org/10.13140/RG.2.2.27089.58728

Majumdar, J., & Bhunia, A. K. (2006). *Elitist genetic algorithm approach for assignment problem. Advanced Modeling and Optimization, 8*(2), 1–15.

Mona Gothi, R. P. A. B. P. (2023). *Optimal solution to the assignment problem. Annals of Mathematics and Computer Science, 16*(July), 112–122.

Murugesan, R., & Esakkiammal, T. (2020). *TERM – a very simple and efficient method to solve assignment problems. Applied Mathematical Sciences, 14*(17), 801–809. https://doi.org/10.12988/ams.2020.914275

Pierobom, J. L., Delgado, M. R., & Kaestner, C. A. A. (2016). *Particle swarm optimization applied to task assignment problem.* 1–8. https://doi.org/10.21528/CBIC2011-16.4

Rahman, M., & Islam, Z. (2022). *A new approach to the branch and bound method for solving assignment problems. IOSR Journal of Mathematics, 18*(6), 13–17. https://doi.org/10.9790/5728-1806011317

Rai, N., & Khan, A. J. (2019). *A brief review on classic assignment problem and its applications. IOSR Journal of Engineering (IOSRJEN), 9*(9), 2278–8719. www.iosrjen.org

Ramesh, G., Sudha, G., & Ganesan, K. (2020). *Method of finding an optimal solution for interval balanced and unbalanced assignment problem. IOP Conference Series: Materials Science and Engineering, 912*(6), 1–9. https://doi.org/10.1088/1757-899X/912/6/062031

Rao, S. S., & Srinivas, M. (2016). *An effective algorithm to solve assignment problems: Opportunity cost approach. International Journal of Research in Engineering and Technology, 6*(1), 48–50.

Senbagam, K., & Saranya, L. (2022). *Enhancement technique for an unbalanced assignment problem in an intuitionistic fuzzy domain. Journal of Fuzzy Systems and Applications, 7*(5), 37–46.

Singh, S., College, R. J. C. D. A. V., & Punjab, D. (2018). *New proposed approach for solving assignment problem and a comparative study with Hungarian method. International Journal of Interdisciplinary Research and Innovations, 6*(3), 717–721. www.researchpublish.com

Stoykova, S., & Spasov, V. (2019). *Choice of fitness functions and parameter settings in genetic algorithms for analysis of induction motors. IOP Conference Series: Materials Science and Engineering, 618*(1). https://doi.org/10.1088/1757-899X/618/1/012024

Wulan, E. R., Devi, A. R., & Nuraiman, D. (2019). *The comparative analysis of Hungarian assessment, matrix ones assignment and alternate MANSI method in solving assignment problem. Journal of Physics: Conference Series, 1402*(7). https://doi.org/10.1088/1742-6596/1402/7/077090

Younas, I., Kamrani, F., Bashir, M., & Schubert, J. (2018). *Efficient genetic algorithms for optimal assignment of tasks to teams of agents. Neurocomputing, 314*, 409–428. https://doi.org/10.1016/j.neucom.2018.07.008

Appendices

Appendix A. Information for the Balanced Assignment Problems

No	Cost Matrix
1	[27 15 22 12; 4 21 31 25; 6 29 22 3; 25 12 18 25](Amponsah et al., 2016)
2	[7 9 11 8 10; 6 5 10 9 7; 12 7 6 10 8; 18 8 12 7 11; 17 12 14 11 6](Amponsah et al., 2016)
3	[18 24 12 6 30 7 13; 11 7 14 22 20 15 8; 10 29 8 28 24 7 25; 8 23 21 16 9 19 17; 4 2 24 18 4 14 9; 18 2 23 9 9 30 12; 3 7 3 26 9 17 11](Amponsah et al., 2016)
4	[7 9 11 8 10 12 15 13 14; 6 5 10 9 7 11 13 18 15; 12 7 6 10 8 9 12 17 13; 18 8 12 7 11 10 14 16 19; 17 12 14 11 6 8 10 15 18; 14 9 13 12 5 7 8 19 16; 8 10 15 10 12 18 17 20 19; 11 13 16 17 14 15 21 12 17; 15 14 9 16 13 6 20 21 11](Amponsah et al., 2016)
5	[9 2 7 8; 6 4 3 7; 5 8 1 8; 7 6 2 4](Rahman & Islam, 2022)
6	[12 8 7 15 4; 7 9 1 14 10; 9 6 12 6 7; 7 6 14 6 10; 9 6 12 10 6](Ekanayake, 2021)
7	[8 4 2 6 1; 0 9 5 5 4; 3 8 9 2 6; 4 3 1 0 3; 9 5 8 9 5](Afroz & Hossen, 2017)
8	[85 75 65 125 75; 90 78 66 132 78; 75 66 57 114 69; 80 72 60 120 72; 76 64 56 112 68](MONA GOTHI, 2023)(Dil Afroz, 2023)
9	[5 7 9 10; 12 8 5 6; 6 9 11 9; 7 13 8 6](Hashim & Shiker, 2023)
10	[8 26 17 11; 13 28 4 26; 38 19 18 15; 19 26 23 10](Dil & Anwar, 2024)
11	[35 22 60 41 27 52 44; 51 39 42 33 65 47 58; 25 32 53 41 50 36 43; 32 28 40 46 3 55 49; 43 36 45 63 57 49 42; 27 18 31 46 35 42 34; 48 50 72 59 43 64 58](Majumdar & Bhunia, 2006)
12	[12 30 21 15; 18 33 9 31; 44 25 24 21; 23 30 28 14](Dil Afroz, 2023)
13	[2 5 5 4; 8 6 7 9; 4 5 8 7; 6 7 6 5](Singh et al., 2018)
14	[10 5 10 15; 3 9 15 3; 10 7 3 2; 2 3 4](Ramesh et al., 2020)

Appendix B. Information for the Unbalanced Assignment Problems

No	Cost Matrix
1	[5 2 4 2 5 ;2 4 7 6 6 ;6 7 5 8 7 ;5 2 3 3 4 ;8 3 7 8 6 ;3 6 3 5 7](Hashim & Shiker, 2023)
2	[3 6 2 6; 7 1 4 4; 3 8 5 8; 6 4 3 7; 5 2 4 4; 5 7 6 2](Abdur Rashid, 2017)
3	[5 2 6 2; 1 7 4 4; 5 0 3 0; 2 4 5 1; 3 6 4 5; 3 1 2 4](Kore, 2012)
4	[3 6 2 6; 7 1 4 4; 3 8 5 8; 6 4 3 7; 5 2 4 3; 5 7 6 4](Kore, 2012)
5	[9 26 15; 13 27 6; 35 20 15; 18 30 20](Singh et al., 2018)
6	[6 5 1 6; 2 5 3 7; 3 7 2 8; 7 7 5 9; 12 8 8 6; 6 9 5 10](Khandelwal, 2018)
7	[5 7 11 6 5; 8 5 5 6 5; 6 7 8 2 4; 10 4 8 2 4](Murugesan & Esakkiammal, 2020)
8	[3 4 7 6; 4 6 8 9; 5 7 4 6; 7 9 10 3](Rao & Srinivas, 2016)
9	[21 14 7; 15 10 5; 15 10 5; 12 8 4](Rao & Srinivas, 2016)
10	[43 82 80 78; 64 84 103 94; 94 106 108 92(Senbagam & Saranya, 2022)]

Appendix C. Information for the Randomly Generated Balanced Assignment Problems

No	Cost Matrix
1	[9 1 9 4 10; 3 3 2 4 3; 5 7 4 1 8; 7 2 3 1 8; 8 7 2 8 4]
2	[5 7 5 2 5 9; 3 8 9 3 5 2; 8 6 4 7 6 8; 5 2 4 7 6 8; 1 4 7 9 4 5;2 6 2 5 3 10]
3	[12 8 10 1 14 13 2; 6 1 14 8 8 15 11; 7 3 3 2 11 9 1; 13 12 5 1 5 11 14; 14 4 10 9 1 5 4; 10 3 4 3 7 10 7; 9 5 7 14 10 2 11]
4	[7 11 1 13 11 15 3 14; 14 5 3 11 6 8 8 12; 14 1 2 3 7 10 15 7; 7 15 11 3 15 1 6 7; 8 15 14 3 12 1 1 2; 8 1 12 8 15 9 3 8; 13 11 4 5 6 2 14 1; 1 12 14 10 1 13 12 4]
5	[21 42 45 43 14 20 49 41 12; 38 10 46 36 19 4 38 4 39; 40 4 14 24 37 30 37 5 21; 24 42 49 35 34 30 14 35 46; 43 43 24 6 32 6 46 9 23; 38 7 45 20 6 9 24 20 18; 30 42 28 35 11 45 47 7 9; 3 8 37 38 23 37 18 27 2; 28 36 14 33 47 21 24 3 2]
6	[726 114 733 431 870; 874 181 390 363 909; 777 997 648 893 759; 956 753 426 761 558; 793 753 178 235 52]
7	[178 176 6 19 159 36; 50 79 88 143 191 118; 115 22 198 5 115 142; 95 77 182 18 166 82; 89 185 64 22 179 3; 78 34 9 68 113 113]
8	[1525 1612 1507 403 389 441; 945 567 1181 1220 557 457; 1290 194 1601 163 1286 571; 1342 989 1301 924 1786 931; 155 341 1050 1439 717 1583; 1665 246 1080 1051 1540 567]
9	[40 47 12 2; 7 15 2 17; 31 2 34 30; 38 26 39 41]

10	[54 51 14 90 22; 40 92 43 93 17; 45 51 64 22 59; 17 40 23 78 18; 54 90 64 48 66]

Appendix D. Information for the Randomly Generated Unbalanced Assignment Problems

No	Cost Matrix
1	[9 10 3 7 7; 7 9 8 3 6; 1 8 10 5 1; 3 2 9 2 8]
2	[6 17 13 9; 9 5 11 10; 10 14 17 11; 12 3 16 14; 13 11 10 19]
3	[7 14 2 4 6; 15 9 15 9 3; 15 4 9 8 1; 9 11 15 11 5; 1 9 7 8 7; 9 2 5 8 2]
4	[8 9 13 13 3 3; 2 18 5 6 9 11; 6 4 9 12 14 15; 6 9 9 11 5 6; 5 6 12 3 11 20]
5	[14 30 14 26 30 15 15; 3 12 28 17 18 18 20; 29 9 13 12 10 9 21; 14 27 6 25 17 5 12; 15 18 30 15 27 9 6; 26 11 18 18 5 10 26]
6	[8 7 1 10 2 10; 10 3 5 4 5 3; 4 4 8 7 3 4; 6 6 4 4 3 7; 1 8 8 5 6 1; 7 4 9 9 1 10; 1 4 2 5 3 7]
7	[36 16 18 12 12; 4 1 6 28 30; 2 24 7 17 10; 6 20 14 9 2; 7 28 4 30 15; 3 28 8 16 24]
8	[993 278 877; 598 315 939; 458 754 477; 816 328 423]
9	[45 72 78 16 94; 4 58 55 41 46; 17 88 63 77 4; 83 63 98 18 83]
10	[7 3 9 10 6; 7 4 5 2 6; 4 10 6 5 8; 3 4 5 3 3]
11	[38 44 27 26; 44 30 62 62; 39 60 16 43; 8 29 60 44; 58 63 39 44]