

## **Warehouse ordering assistance system by a machine-learning robot using heuristic search.**

**JoséOgosi<sup>1</sup>, Jorge Mayhuasca<sup>2</sup>, Ivan Petrlik<sup>3</sup>, Ciro Rodriguez<sup>4</sup>, Roberto Esparza<sup>5</sup>, Henry Bermejo<sup>6</sup>**

<sup>1,3,6</sup> *University Cesar Vallejo*

<sup>2,4,5</sup> *National University Federico Villarreal*

<sup>1</sup>jogosi@ucvvirtual.edu.pe,<sup>2</sup>jmayhuasca@unfv.edu.pe,<sup>3</sup>ipetrlika@ucvvirtual.edu.pe,<sup>4</sup>crodriguez@unfv.edu.pe,<sup>5</sup>mesparza@unfv.edu.pe,<sup>6</sup>hbermejot@ucvvirtual.edu.pe

### **Abstract**

*The heuristic function (n) attempts to guide the search to quickly arrive at the solution. The objective of this study was to implement a heuristic search strategy A \* to solve a real problem. It has the function of finding a solution to the problem using non-rigorous methods such as trial and error, rules of thumb, etc. The algorithm used to solve the problem is efficient. In addition, the results obtained on the console are as expected from the analysis, so there is assurance that the code is working correctly. This research can allow the copy function for modification of the data matrices and the creation of several matrices to cover all the data.*

**Keywords:** *Algorithm, heuristics, classification, matrices, robot.*

### **1. Introduction**

Heuristic (from the Greek word εὐρίσκειν) means "discovery, invention" (the etymology shared with Eureka) occurs in multiple categories of grammar. When used as a noun, it refers to the field of discovery, art, or science. When it shows up as an adjective, it refers to something more specific, such as strategy, rules, syllogism, or reasoning.

In the Amazon example, you want to use a robot to sort inventory in the warehouse. Amazon has three inventories (a table of supplies for sale) in specific locations in the warehouse. The robot is responsible for moving the three inventories to the target location.

The robot can move horizontally and vertically to load or unload inventory. This activity requires that you use a heuristic search strategy A \* to generate a plan that allows your machine learning robot to move the inventory from its initial state to its target state.

The implementation of the A \* algorithm considers the Manhattan distance as a heuristic function. The actual cost (g) of each robot action is 1. The objective of the study was to specify and execute the sequence of actions needed programmatically to achieve the target state in simple notation. Example: "move R row1 column2" or "move R row0 column2" or "load R M1 row0 column2".

### **2. Background, review of the literature, state of the art**

Cab driver geometry was coordinated by Hermann Minkowski on the 19th. Cab driver distances, also known as straight distances, L1 distances or norms, city distances, Manhattan distances, or Manhattan lengths, have variations corresponding to the names of the geometries [1]. The last name implies a grid layout for most roads on Manhattan Island. In other words, the shortest route a car can take between two points in the city is the same distance as the two points on the road shape.

In 2017, they evaluated the efficiency of 82 web services networks of a heuristic search technique with machine learning [2], thus, this dependency has shown great interest in studying relationships between labels and ways to explore them [3], likewise, the application of hybrid heuristics to both complex functions has demonstrated the potential of these algorithms and the importance of recognizing the correct transition moments between searches [4].

In 2018, they compared the heuristic search algorithm to demonstrate the accuracy and advantages of autonomous driving versus conventional motion planning [5]. It was shown that the beneficial

features of robust machine learning are based on heuristic search strategies [6] as well as data encryption [7]. It was also shown that the algorithms avoid guessing the information but use the data it can associate to provide the information [8].

In 2019, metaheuristics are found to be a convenient way to manage and optimize resource flow in large and complex scenarios [9]. Heuristics were presented to handle different combinations in a procurement portfolio [10]. It was also shown that heuristic search provides significant improvements to machine learning concerning the number of nodes traversed through the provided DFS algorithm [11]. Then, classical algorithms are compared with the heuristic search on a real-life dataset [12], as well as it is shown that it is possible to identify the attribute that can solve the problem in classification [13].

In 2020, it is proposed the solution of strong seismic movements with the proposal of metaheuristics [14], also the solution to problems of the orientation of universities is proposed the heuristic search system for students and teachers [15]. Algorithms that reach higher interactions to the rest of algorithms being able to perform it for one of time are also evaluated [16], as well as it is also proposed to strengthen the exploration of the program with heuristic search methods to encode the structures of the neural network module for a task of visual questions [17].

In 2021, they performed validations on gamification and machine learning in achievement-focused video games [18]. Then, the superiority of nonlinear testing to common activation functions as a search guide is demonstrated [19]. Also, the hand-driven trajectory of scripts with sorting and heuristic search was recovered [20].

### 3. Methodology

In this research, the heuristic search strategy A\* will be used to generate a plan that allows the robot to move the inventory from an initial state to a target state.

Initial State: The initial state of the problem will be represented in a 4x4 matrix of characters as follows:

|   | 0  | 1 | 2 | 3  |
|---|----|---|---|----|
| 0 | M1 | # |   | M3 |
| 1 |    | # |   |    |
| 2 | M2 |   | R |    |
| 3 |    |   |   |    |

Figure 1. Initial matrix state.

Where:

- R: represents the robot. Initially, it is located at position [2,2].
- #: represents a wall.
- M1, M2, and M3: represent the three inventories that the robot must move. And they are located at positions [0,0], [2,0], and [0,3] respectively.

Target state

The robot must move the three inventories, M1, M2, and M3, to the following positions: To solve this problem, code steps will be used to understand how each part is boxed and placed at its destination.

|   | 0 | 1  | 2  | 3  |
|---|---|----|----|----|
| 0 |   | #  |    |    |
| 1 |   | #  |    |    |
| 2 |   |    |    |    |
| 3 |   | M3 | M2 | M1 |

Figure 2. Matrix target state.

Tasks requested

Implement algorithm A\* considering the following:

- As a heuristic function, the Manhattan Distance.

- The real cost (g) of each action of the robot is 1.
- The code should be executed and indicate the sequence of actions needed to reach the target state using a simple notation. For example: "move R row1 column2" or "move R row0 column2" or "load R M1 row0 column2".

#### **4. Proposal development**

##### Code analysis

- Copy function: Copies a matrix, this is used because at the moment of traversing a place the robot leaves a trace (-1) in the matrix.
- Function minor distance: Calculates the smallest distance between points, intending to take from one place to another and without traveling more than it should.
- Function find\_not\_resolved: Searches for the positions of the boxes in which they are still resolved, i.e., searches for boxes that are not in place.
- Function calculate\_l\_d\_arrival: Calculates the place of arrival, i.e., searches where the box is located and returns the position.
- Mark function: It marks as if it were a wall since the robot can pass under an object if it is not carrying the boxes, but it cannot pass under it if it is carrying an object, that is why it marks the objects (boxes) to know that they cannot pass through there.
- Path function: it traverses the path from one point to another, i.e., it traverses the places both going and coming.
- Steps in the code to solve the problem.

Each part is where a box is carried and placed at its destination location:

- First, we call the shortest distance function to know where we have to go and make sure that it is one of the not-so-long roads (long road constraint). This function means that we use the closest ones to move as little as possible. Once we know where to go, we mark the place to go from one side to the other.
- Subsequently, we call the function path, i.e., we are going to travel from one place to another (place of origin where the robot is located towards the destination where we must arrive).
- Here we use the copy function to clean up the matrix (since it is filled with numbers at the time of traversing and marking).
- We calculate the arrival position to take it towards its destination.
- We update the robot's position and mark the position where we want to arrive.
- We use the mark function to avoid going through the boxes that are still left on the path.
- We use the path function to go to the destination location.
- Now we use the copy function to clear the array of the path function, from the mark function and update the data.
- We then use the find unresolved function to find out which ones are still unresolved (boxes that are not yet in place). This process is repeated until all the boxes are in place.

The programming language selected was Python. Due to the creation of classes, it was necessary to structure the data as necessary to represent the space between the states and the nodes that were explored in the tree. The developed program is an original work which evidences below the pieces of code worked on:

This means importing NumPy, initializing the variable for iterations, and representing the initial matrix containing data and a pair of methods.

```
import numpy as np
resueltos = 0
stringLab = np.array([[0,1,0,0],
                      [0,1,0,0],
                      [0,0,0,0],
                      [0,0,0,0]])
temporal = np.array([[0,1,0,0],
                     [0,1,0,0],
                     [0,0,0,0],
                     [0,0,0,0]])
laberinto = np.array([[0,1,0,0],
                      [0,1,0,0],
                      [0,0,0,0],
                      [0,0,0,0]])
```

Figure 3.Explanation coding 1.

The next step is to define the search function, in this case, it is requested to copy the matrix if what is really requested is to define a path through which the robot will pass.

```
def copiar(laberinto,stringLab):
    for z1 in range(4):
        for z2 in range(4):
            laberinto[z1][z2] = stringLab[z1][z2]

fff = [[2,1,0,4],
        [0,1,0,0],
        [3,0,5,0],
        [0,6,7,8]]
ff = [[2,1,0,4],
        [0,1,0,0],
        [3,0,5,0],
        [0,6,7,8]]

for i in range(4):
    print(fff[i])

R_lugar_robot = [2,2]
R_lugar_p = [0,0,0]
```

Figure 4.Explanation coding 2.

Indeed, as the objective is to find the distance between the robot and the target point, a function is produced which asks it to do so.

```
def menor_distancia(laberinto,R_lugar_robot, R_lugar_p):
    f = 8
    for z1 in range(4):
        for z2 in range(4):
            if laberinto[z1][z2]==2 or laberinto[z1][z2]==3 or laberinto[z1][z2]==4:
                menorex = abs(z1 - R_lugar_robot[0])
                menory = abs(z2 - R_lugar_robot[1])
                distancia = menorex + menory

            if(distancia<f):
                f = distancia
                R_lugar_p = [z1,z2,laberinto[z1][z2]]
    return R_lugar_p
```

Figure 5.Explanation coding 3.

The search can be checked by the function that searches for what is requested when the robot does not reach the box.

```
def buscar_no_resuelto(laberinto,R_lugar_robot, R_lugar_p):  
    f = 8  
    m = 0  
    for z1 in range(4):  
        for z2 in range(4):  
            m = 0  
            if (laberinto[z1][z2]==2 or laberinto[z1][z2]==3 or laberinto[z1][z2]==4):  
                for z3 in range(resueltos):  
                    if(realizados[z3]==laberinto[z1][z2]):  
                        m = m + 1  
  
            if(m<1):  
                menorex = abs(z1 - R_lugar_robot[0])  
                menory = abs(z2 - R_lugar_robot[1])  
                distancia = menorex + menory  
                if(distancia<f):  
                    f = distancia  
                    R_lugar_p = [z1,z2,laberinto[z1][z2]]  
  
    return R_lugar_p
```

Figure 6.Explanation coding 4.

This involves looking for the place where the box moves, unlike the previous function, it looks for the robot to reach the box.

```
def calcular_1_d_llegada(fff,R_lugar_robot, R_lugar_p,zzz):  
    for z1 in range(4):  
        for z2 in range(4):  
            if(zzz[2]== 10 - fff[z1][z2]):  
                R_lugar_p[0] = z1  
                R_lugar_p[1] = z2  
                R_lugar_p[2] = fff[z1][z2]  
    return R_lugar_p
```

Figure 7.Explanation coding 5.

Once the search for the boxes has been created, the next step is to create the method that marks the walls when the box is a load being transported by the robot.

```
def marcar(laberinto,fff):  
    for z1 in range(4):  
        for z2 in range(4):  
            if(fff[z1][z2] != zzz[2] and fff[z1][z2]<5 and fff[z1][z2]!=0 and fff[z1][z2]!=1):  
                laberinto[z1][z2]=1
```

Figure 8.Explanation coding 6.

Of all the possible paths the robot will use the orientation towards the place to be reached. It is through this function that the robot will walk along the paths demanding to use the solar orientation to refer to the directions on the search axis.

```
def recorrido(i, j):
    if laberinto[i][j] == 3:
        return [(i, j)]

    if laberinto[i][j] == 1:
        return []

    laberinto[i][j] = -1

    if i > 0 and laberinto[i - 1][j] in [0, 3]:          # Norte
        camino = recorrido(i - 1, j)
        if camino: return [(i, j)] + camino

    if j < len(laberinto[i]) - 1 and laberinto[i][j + 1] in [0, 3]: # Este
        camino = recorrido(i, j + 1)
        if camino: return [(i, j)] + camino

    if i < len(laberinto) - 1 and laberinto[i + 1][j] in [0, 3]: # Sur
        camino = recorrido(i + 1, j)
        if camino: return [(i, j)] + camino

    if j > 0 and laberinto[i][j - 1] in [0, 3]:       # Oeste
        camino = recorrido(i, j - 1)
        if camino: return [(i, j)] + camino

    return []
```

Figure 9.Explanation coding 7.

The first part of the path is defined in the instance pertaining to the distance that separates the robot from the target point. Variables and methods containing each of the states of the declared matrices are defined.

```
zzz = menor_distancia(fff,R_lugar_robot, R_lugar_p)

laberinto[zzz[0]][zzz[1]] = 3

for x in recorrido(2,2) : print(x)

print("llega al lugar")
print("aqui llevara la caja")

copiar(laberinto,stringLab)

fff[zzz[0]][zzz[1]] = 0

zz1 = calcular_1_d_llegada(fff,R_lugar_robot, R_lugar_p,zzz)

R_lugar_robot = [zzz[0],zzz[1]]

laberinto[zz1[0]][zz1[1]] = 3

marcar(laberinto,fff)

for x in recorrido(zzz[0],zzz[1]): print(x)

print("primera caja colocada")
```

Figure 10.Explanation coding 8.

As can be seen in this second part, the method of introducing arrays is at a more internal level than in the first part. These commands have been operated according to the established order which, by itself, inverts the order if the factors of loading and transporting boxes are taken into account, being the expected result without the need of transporting arrays.

```
ff[zz1[0]][zz1[1]] = zzz[2]

resueltos = resueltos + 1
realizados = []
realizados.append(zzz[2])

R_lugar_robot = [zz1[0],zz1[1]]
zzz = buscar_no_resuelto(ff,R_lugar_robot, R_lugar_p)

copiar(laberinto,stringLab)
laberinto[zzz[0]][zzz[1]]=3

for x in recorrido(zz1[0],zz1[1]): print(x)

copiar(laberinto,stringLab)
marcar(laberinto,ff)

print("llega al lugar")
print("aquí llevara la caja")

zz1 = calcular_1_d_llegada(ff,R_lugar_robot, R_lugar_p,zzz)
laberinto[zz1[0]][zz1[1]] = 3

for x in recorrido(zzz[0],zzz[1]): print(x)
print("segunda caja colocada")
```

Figure 11.Explanation coding 9.

Finally, as can also be seen, the sequence used in part 3 has the criteria to support the indices at the boundaries when the next box has been placed supporting the election motif each time resolving negative and positive indices other than the ratios in the initial matrix.

```
ff[zz1[0]][zz1[1]] = zzz[2]

resueltos = resueltos + 1
realizados.append(zzz[2])

ff[zz1[0]][zz1[1]] = zzz[2]
ff[zzz[0]][zzz[1]] = 0

zzz = buscar_no_resuelto(ff,R_lugar_robot, R_lugar_p)
copiar(laberinto,stringLab)

laberinto[zzz[0]][zzz[1]] = 3

for x in recorrido(zz1[0],zz1[1]): print(x)

print("llegar a lugar")
print("aquí llevara la caja")

copiar(laberinto,stringLab)
marcar(laberinto,ff)

zz1 = calcular_1_d_llegada(ff,R_lugar_robot, R_lugar_p,zzz)

laberinto[zz1[0]][zz1[1]] = 3
for x in recorrido(zzz[0],zzz[1]): print(x)

print("tercera caja colocada ")
```

Figure 12.Explanation coding 10.

## 5. Result

Implementing the Manhattan distances, we obtain the following states that lead us to the arrangement of the boxes. The robot encountered the following maze:

```
[2, 1, 0, 4]
[0, 1, 0, 0]
[3, 0, 5, 0]
[0, 6, 7, 8]
```

The robot had to find the box at the coordinates (2,0) but it started at the coordinates (2,2), in order to reach the place, the robot performed the following on the screen:

(2, 2) (1, 2) (0, 2) (0, 3) (1, 3) (2, 3) (3, 3) (3, 2) (3, 1) (2, 1) (2, 0)

The next choice was the number of the place where it had to take the box to store it, this point was located at the coordinates (3,2), for which the robot made the following movements on the screen:

(2, 0) (2, 1) (2, 2) (1, 2) (1, 3) (2, 3) (3, 3) (3, 2)

The execution of these steps means that the first box will have been placed. The next step was to get to the place to carry a second box, the same that was at coordinates (0,3), so the robot accessed the following steps to go to the point from its coordinates (3,2) on the screen:

(3, 2) (2, 2) (1, 2) (0, 2) (0, 3)

Once the robot has reached the point of the box, it can take the box to the coordinates (3,1) to store it in the warehouse at these coordinates, from this coordinate the robot executes the movements through the maze on the screen:

(0, 3) (1, 3) (2, 3) (2, 2) (2, 1) (3, 1)

This means that the second box has been placed, going on to reach the point where it would find the third box at coordinates (0,0) from the robot's current location at coordinates (3,1), for which it performed the following on the screen:

(3, 1) (2, 1) (2, 0) (1, 0) (0, 0)

Finally, i was going to perform the check of its (0,0) coordinates to bring the third box to the target matrix at (3,3) and simply displayed on screen:

(0, 0) (1, 0) (2, 0) (2, 1) (2, 2) (1, 2) (0, 2) (0, 3) (1, 3) (2, 3) (3, 3)

## **6. Discussion**

To solve the problem, the code must allow robot acceleration mechanisms within the program execution, a unique list of functions that modify and suppress the behavior that generates defects in the program performance is created. Also, when explaining the detail of the code development you should use jupyter notebook when you must include comments of the code before and after the modification of the behavior that improves the performance of the robot.

At the moment of copying the matrices, there was the inconvenience that everything was transcribed, that is to say, the same original matrix was copied, where if a data was changed in one matrix, it was also changed in the other one, although the modified matrix had the same data. That is why it was decided to implement the copy function.

Another inconvenience arose when trying to cover all the data of the total matrix in a single matrix, so it was decided to create several matrices to store the states variable according to the necessary code that represents the function that allows storing the matrices with the loaded data.

## **7. Conclusions**

The results obtained in the console are as expected in the analysis so we can confirm that the code for the robot with machine learning works correctly being the order of choice of the functions that influence the decisions of the robot.

The algorithm used for solving the warehouse ordering problem is efficient as far as the help system with the machine learning robot is concerned, the continuation of the changes in the states were stored correctly after the application of several matrices.

The robot was activated with machine learning that receives an initial state achieving with the help of the code based on heuristic search the detection of the paths that playfully are processed by the robot to reach the target state.

## 8. Acknowledgments

To the Federico Villarreal National University, Lima, Peru, for the conference given at the 7th International Conference on the green in Computer Engineering and Technologies, to show the level of research for our inspiration in the writing of this article.

## 9. References

- [1] J. C. P, Leivas. Geometria Euclidiana e do Taxi: um problema concreto e os Registros de Representações Semióticas. *Revista de Educação Matemática*, São Paulo, 16(22), 252-269, (2019)
- [2] M, Kessentini, H, Wang, J. T, Dea & A, Ouni. Improving web services design quality using heuristic search and machine learning. In 2017 IEEE International Conference on Web Services (ICWS) (pp. 540-547). IEEE, (2017).
- [3] D, Mena. Métodos de inferencia basados en búsqueda heurística para cadenas de clasificadores probabilísticos. (2017).
- [4] D, Vera. Algoritmos Heurísticos Híbridos para el diseño de S-Cajas (Doctoral dissertation, Master dissertation, Universidad de La Habana). (2017)
- [5] J, Bernhard, J, R, Gieselmann, K, Esterleand A, Knol. Experience-based heuristic search: Robust motion planning with deep q-learning. In 2018 21st International Conference on Intelligent Transportation Systems (ITSC) (pp. 3175-3182). IEEE. (2018)
- [6] Y, Mohammadi, Y, M, Saeb & A, Penlidis. Heuristic search strategy for transforming microstructural patterns to optimal copolymerization recipes. *Macromolecular Theory and Simulations*, 27(2), 1700088. (2018)
- [7] R, Velvizhi. HEURISTIC SEARCH FOR MACHINE LEARNING. *International Journal of Pure and Applied Mathematics*, 119(12), 10201-10212. (2018)
- [8] S, Shah, S, B, He, B, K, Xu, C, Maung and H, Schweitzer. Solving generalized column subset selection with heuristic search. In Thirty-Second AAAI Conference on Artificial Intelligence. (2018)
- [9] J, Abarca, P, Bernal, S, Nesmachnow and A, Trejo. Metaheurísticas. *Inventio. La génesis de la cultura universitaria en Morelos*. (2019)
- [10] L, Jariego. Funciones de adquisición heurísticas para Optimización Bayesiana (Bachelor's thesis). (2019)
- [11] B, Kucharski, A, Deihim and M, Ergezer. Machine Learning Based Heuristic Search Algorithms to Solve Birds of a Feather Card Game. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 33, No. 01, pp. 9656-9661). (2019)
- [12] J, Mattiev, and B, Kavšek. Using constrained exhaustive search vs. greedy heuristic search for classification rule learning. In *Computer Science Research Conference* (Vol. 43).
- [13] S, Zúñiga. Desarrollo de un algoritmo de clasificación de secuencias de ADN, basado en localizar regiones conservadas y una técnica de búsqueda heurística (Master's thesis). (2019)
- [14] A, Puris, P, Novoa and Oviedo. Desarrollo de metaheurísticas poblacionales para la solución de problemas complejos. (2020)
- [15] J, Quiroz and K, Ramírez. Sistema de búsqueda heurística para la localización de las principales oficinas de la Universidad Nacional Pedro Ruiz Gallo utilizando el algoritmo A Star. (2020)
- [16] A, Valero. Estrategias de aprendizaje automático aplicadas a videojuegos (Doctoral dissertation, Universitat Politècnica de València). (2020)
- [17] Y, Mohammadi, M, Saeb and A, Penlidis. Heuristic search strategy for transforming microstructural patterns to optimal copolymerization recipes. *Macromolecular Theory and Simulations*, 27(2), 1700088. (2018)
- [18] X, Shi, X, J, Chen, J and L, Wang. Heuristic Search for Activation Functions of Neural Networks Based on Gaussian Processes. In 2021 International Joint Conference on Neural Networks (IJCNN) (pp. 1-8). IEEE. (2021)
- [19] T, Wang, and C, Liu. Handwriting Trajectory Recovery from Off-Line Multi-Stroke Characters by Deep Ordering Prediction and Heuristic Search. In 2021 IEEE International Conference on Multimedia and Expo (ICME) (pp. 1-6). IEEE. (2021)

**Authors**



**José Ogosi**

Systems and Computer Engineer, Master in Information Technology Management, Doctorate in Systems Engineering At The Federico Villarreal National University, Teacher And Thesis Advisor At The technological University of Peru, At The Federico Villarreal National University, At The Cesar Vallejo University, And At The University of Sciences and Humanities. Diploma in Public Management. Responsible conduct in research. Specialist in Information Technologies and Optimization Algorithms, In Public Management, Interests in It Projects in Software Development, Optimization Algorithms, Bpm, Cloud Technology. Developed applications in PHP, ANDROID, JAVA, and PYTHON and institutional portals.



**Jorge Mayhuasca**

Doctor in Engineering, Master in Systems Engineering, Industrial Engineer, Research Professor, Senior Lecturer appointed to the Faculty of Industrial and Systems Engineering. Director of the Academic Department of Systems Engineering. Director of the Professional School of Industrial Engineering. Member of the Scientific Committee of the School of Industrial and Systems Engineering. President of the quality committee of the Professional School of Systems Engineering.



**Ivan Petrlik**

Doctor in Systems Engineering, Master in Systems Engineering. Research Professor RENACYT – UCV - CONCYTEC. Software Analyst and Programmer with more than 18 years of experience in national and private companies. Member of the digital literacy commission of the College of Engineers of Peru. Member of the Information Technology and Communications Committee. Member of the Community of Knowledge R+D+I+Systems of the Faculty of Industrial Engineering and Systems of the Universidad Nacional Federico Villarreal and Member of the Circle of Ambassadors of Peace - Geneva - Switzerland



**Ciro Rodriguez**

Professor at the Software Engineering School of the Universidad Nacional Mayor de San Marcos and also at the Computer Science and Graduate School of the Universidad Nacional Federico Villarreal, with science studies at the Abdus Salam International Center for Theoretical Physics (ICTP) and at the United States Particle Accelerator School (USPAS).



**Roberto Esparza**

He is an Industrial Engineer with a Master's Degree in Industrial Engineering with a mention in production and a Ph.D. in Engineering from the Universidad Nacional Federico Villarreal. He is a Senior Lecturer and is currently director of the Central Quality Office and president of the Institutional Licensing Commission of the UNFV.



**Henry Bermejo**

Doctor in Public Management and Governance, Master in Systems Engineering with a mention in Information Technology, Master in University Teaching. University Professor with more than 20 years of experience in different National and State Universities. Collegiate and active member of the College of Engineers of Peru - La Libertad. Post Degree in Human Resources. Specialist in Networks and Communications - CISCO. Chief in the Information Technology Area in the Company BUSINESSOFT Tecnología de Sistemas and with wide participation in Telecommunications companies.