An Automated Mobile Crane Selection System for Heavy Industrial Projects

by

Ramtin Azami

Master of Science in Geotechnical Engineering, University of Tarbiat Modares, 2017

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of

Master of Science in Engineering (MScE)

in the Graduate Academic Unit of Civil Engineering

Supervisor: Zhen Lei, Ph.D., Department of Civil Engineering

Examining Board: Jeff Rankin, Ph.D., PEng, Civil Engineering
Lloyd Waugh, Ph.D., PEng, Civil Engineering
Clodualdo Aranas, Ph.D., PEng, Mechanical Engineering

This thesis is accepted by the
Dean of Graduate Studies

THE UNIVERSITY OF NEW BRUNSWICK

December 2021

ABSTRACT

Crane selection for construction projects is a complicated and time-consuming process due to a vast number of parameters and inter-related planning constraints. Selecting the most appropriate cranes can improve the productivity and safety of construction projects as selecting the wrong crane can result in deadly incidents. An algorithm for automatic mobile crane selection in heavy industrial construction projects that employs heuristic and artificial neural network methods, entitled Crane Configurator, is introduced in this research. Crane Configurator consists of two stages: 1) mobile crane configuration selector and 2) mobile crane configuration predictor. The developed application considers the module's features (e.g., weight, length, and width), the project's budget, schedule, and safety. It proposes a crane for the project that satisfies the user's demand. The proposed mobile crane predictor employs mathematical and machine learning techniques, and it leverages the power of relational databases to train an artificial neural network (ANN) to predict the cranes for future projects. Accuracy rates as high as 74% have been achieved for the final results through evaluating the model using the test set from real projects. Lift engineers, project stakeholders, and construction crew can benefit directly or indirectly through proper crane selection, which shortens the project's schedule, improves the job site's safety, and reduces the project's cost.

DEDICATION

This thesis is dedicated with love and respect to my wife, Mahsa, for her unconditional love and support.

ACKNOWLEDGEMENTS

First of all, I would like to express my deepest gratitude to my supervisor, Dr. Zhen Lei, for making this life-changing experience possible. Without his continuous support and guidance throughout my study, it would have been impossible to finish this degree.

I want to thank Mr. Ulrich Hermann and Mr. Travis Zubick from PCL Industrial Management Inc. for supporting and sharing their many years of knowledge and experience.

I am grateful for the support of my colleagues and friends in our research group: Brandon Searle, Alex Caskey, Ala Suliman, and José Daniel Cuéllar Lobo.

Finally, I would like to thank my beloved wife, Mahsa, who has shared her incredible ideas and comments throughout this journey.

Table of Contents

List of Tables

List of Figures

List of Symbols, Nomenclature or Abbreviations

ANN	-	Artificial Neural Network
ATB	-	Anti-Two Block
MLP	-	Multilayer Perceptron Neural Network
OSC	-	Off-site Construction
ReLU	-	Rectified Linear Activation
MLP	-	Multilayer Perceptron Neural Network
RBF	-	Radial Basis Function Neural Network
PNN	-	Probabilistic Neural Network
GR	-	Generalized Regression Neural Networks
SQL	-	Structured Query Language

Chapter 1: Introduction

1.1 Background

Researchers in the construction management field have been seeking effective means of improving the quality and efficiency of construction projects for decades. The off-site construction method, an alternative to conventional site-based construction, makes the desired goal more achievable (Pan and Sidwell 2011). Goodier and Gibb (2007) define off-site construction methodology as "the process of manufacturing and preassembly of certain amounts of building components and modules before their shipment and installation on construction sites." Later, the definition was expanded by Quale and Smith (2017) as "planning, design, fabrication, and assembly of building elements at a location other than their final installed location to support the rapid and efficient construction of a permanent structure." Using either definition, heavy mobile cranes have been an inseparable part of off-site constructions regarding lifting and relocating the modules (Han et al. 2015b). Thus, the extensive use of cranes in the construction industry and the high price rate of renting them make these types of machinery one of the most critical parts of construction processes. Proper and accurate lift planning must be carried out before any actual lifting process to ensure the high productivity of crane operations (Lei 2014). Successful lift planning consists of: (1) crane location; (2) crane selection; (3) support system design, and (4) motion planning of mobile crane operations (Han et al. 2015b). Crane selection, which is the main objective of this research, typically consists of two major phases – the crane type selection followed by the model selection. During the crane type selection phase, crane's type such

as mobile, tower, or all-terrain is determined. Following this phase, the exact model of the crane and configuration need to be selected.

Although some applications have been developed recently to automate lift planning, the primary method for crane model selection is through using lift charts that crane manufacturers provide to lift engineers. Manual crane selection becomes time-consuming, and the optimum crane (e.g., in terms of balancing its size/capacity and cost of usage) may not be selected, since the types and models of cranes keep increasing. Recent advance in computer and software technology resulted in more powerful applications and algorithms to handle and process extensive data in a shorter time. In this regard, a 3D-crane evaluation system was proposed to support mobile crane operation selection efficiently and effectively by verifying and evaluating motions of crane operations in a 3D environment. The proposed system was developed using MAXScript, which allows efficient information exchange, safe and reliable 3D visualization design of mobile crane operations, post-visualization simulations, and representation of digital numeric-based crane lift information (Han et al. 2017). A study of tower crane selections by Huang et al. (2019) proposed a cost-based selection and location optimization model for the optimal selection of tower cranes. The results demonstrate that locating the supply point in the midpoint of the long side of the project site (the building layout was assumed to be rectangular so it has long and short side) results in the lowest cost of the tower crane.

PCL Industrial Management is one of the leading construction companies, which implemented the off-site construction methodology to improve the productivity and efficiency of their projects by shortening project timelines, reducing costs, and improving

workplace safety, particularly in heavy industrial construction projects. Figure 1 shows a heavy modular project that PCL accomplished: the heavy modules were transported using trailers to the job site, and heavy mobile cranes were utilized for lifting the modules and setting them at their final locations. During the past decade, PCL has developed an integrated system to cope with the challenges of heavy mobile crane planning in industrial constructions, such as selecting the location of the cranes, path planning for mobile cranes, and sequence of lifting the modules that are part of the off-site construction method (Lei 2014). The company's central database forms the core of the crane management system. The inputs for this database include (1) crane capacity charts; (2) module's details such as weight, length, and height; (3) crane configurations; and (4) site layout boundaries. There are three main systems within the PCL crane management system: ACPO, ASICO, and CPCP. ACPO stands for Advanced Crane Planning and Optimization, which solved the crane location finding problem (Rick Hermann et al., 2010). ASICO stands for Advanced Simulation in Industrial Crane Operation, which tackles the lifting sequence according to the pre-defined lifting logic (Taghaddos et al. 2012). At the same time, CPCP, which stands for Crane Path Checking and Planning, analyzes the crane motions and generates 4-dimensional animations that show the crane movements (Lei et al. 2013).

One of the challenges that remains unsolved is selecting a crane that can satisfy the project's productivity criteria by reducing the cost overrun, shortening the schedule of the project, and improving the safety of the workers and engineers on the job site. Manual crane configuration selection, which includes reading crane configurations' lift chart, checking the boom length, checking the working radius, and checking the crane's

capacity for each module, is the method that PCL is utilizing to select the suitable crane configuration for each module. This method can be time-consuming and may not be applicable to consider all selection criteria simultaneously.

Figure 1: Modular industrial project (photo courtesy of PCL Industrial Management)

1.2 Motivation

The motivation for this research was to reduce injuries and death during the construction phases due to crane incidents while selecting a crane with sufficient lifting capacity and considering economic parameters. Selecting the wrong crane could increase the cost of the project significantly and may cause fatal injuries.

1.3 Goals and Objectives

This research aims to develop a heuristic algorithm and a neural network algorithm for mobile crane selection in heavy industrial construction projects to eliminate the limitations of the current mobile crane selection methods such as low accuracy and high processing time (Figure 2). The primary objectives of this research are:

1. Develop a heuristic algorithm (i.e., involving or serving as an aid to learning, discovery, or problem-solving by experimental and especially trial-and-error methods) that considers critical design parameters for the mobile crane selection (e.g., the duration of a project, the monthly rate of the crane, and the capacity of the crane), and proposes a list of feasible cranes for each project.
2. Develop an artificial neural network based on historical data to predict the optimum mobile crane configurations for industrial projects.
3. Link both the heuristic and the neural network model so that the user has access to both simultaneously to enhance crane selection decisions.
4. Develop an interface to enable the user to interact with the developed applications.

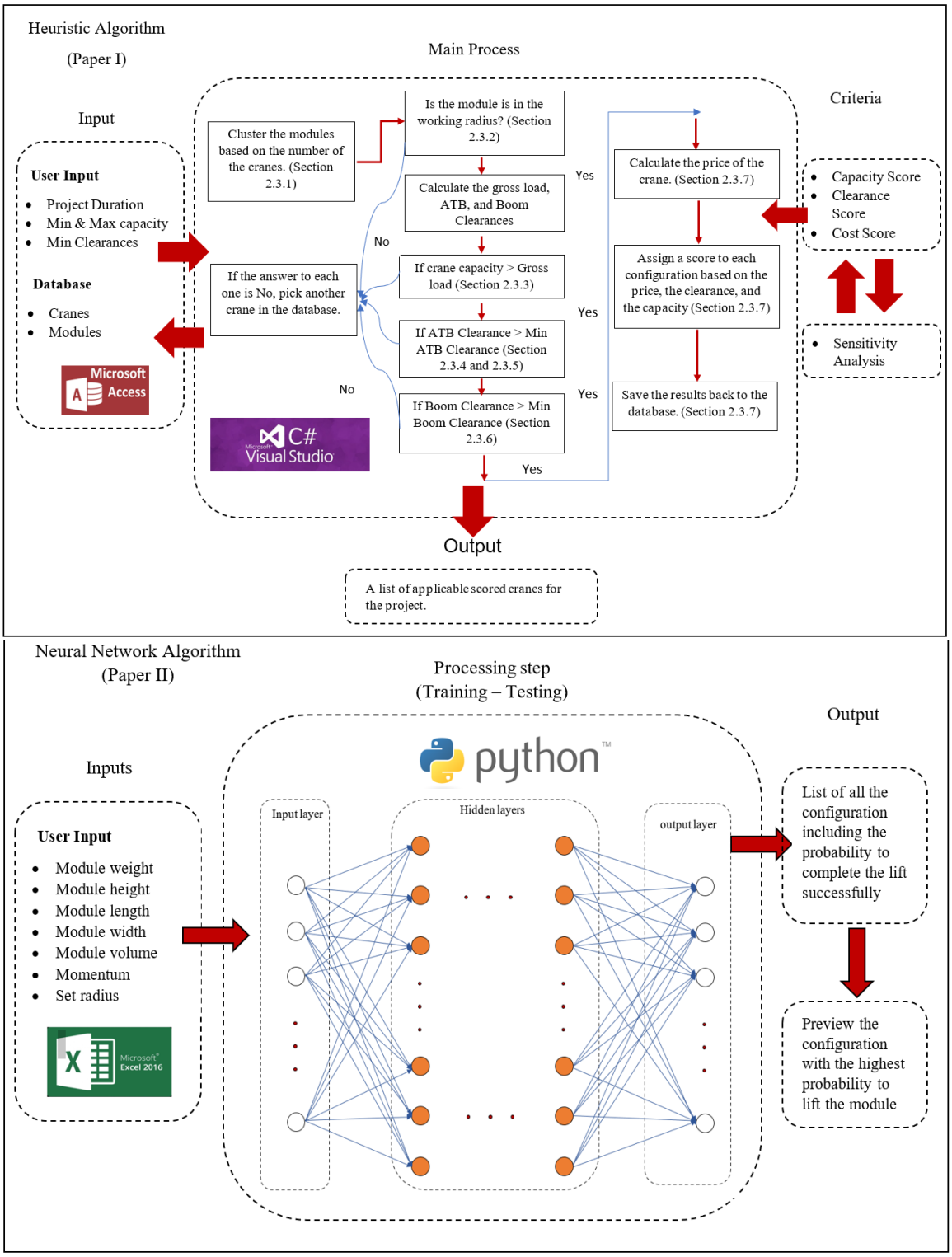


Figure 2: Structure of the proposed algorithm

1.4 Thesis Structure

This research has been written in a paper-based format. Chapter 1 provides an introduction of the research work, objectives, and the contribution of the research. Detailed reviews of related works is presented in both papers (chapters 2 and 3); only a brief explanation of the research context is provided in this chapter to depict the overall trends toward automated mobile crane selection in industrial construction projects.

Chapter 2 contains the first paper, a conference paper that was published at the CSCE annual conference May, 2021 proceeding general conference, in which a heuristic approach was implemented to find all the applicable mobile cranes in the database for construction projects by considering not only the features of the modules and the cranes but the safety parameters as well. Moreover, the algorithm scores each crane configuration based on specifications like the project's duration, the cranes' monthly rate, the cranes' capacity, the cranes' clearances, and the cranes' super-lift presence. A case study was presented in this paper to illustrate the effectiveness of this approach.

Chapter 3 contains the second paper, a journal paper that will be submitted to the *Canadian Journal of Civil Engineering*. This paper presents an entirely different approach from the first paper for selecting crane configuration, which can be applied to any crane. This paper introduces a supervised machine learning technique that trains an artificial neural network and tests the algorithm based on historical data. Once the model is trained and tested, it can be used for future projects to predict the most suitable cranes. The inputs for this model were the module's features, and the outputs were the crane configurations. This neural network was trained and tested with the modules of four real industrial projects.

Finally, chapter 4 includes a summary of the research context and achievements. This chapter was concluded with a brief conclusion of the entire research.

References

Han, S., Bouferguene, A., Al-Hussein, M., & Hermann, U. (Rick). (2017). 3D-Based Crane Evaluation System for Mobile Crane Operation Selection on Modular-Based Heavy Construction Sites. *Journal of Construction Engineering and Management*, *143*(9), 04017060. https://doi.org/10.1061/(asce)co.1943-7862.0001360

Han, S. H., Hasan, S., Bouferguène, A., Al-Hussein, M., & Kosa, J. (2015). Utilization of 3D Visualization of Mobile Crane Operations for Modular Construction On-Site Assembly. *Journal of Management in Engineering*, *31*(5), 04014080. https://doi.org/10.1061/(ASCE)ME.1943-5479.0000317

Hermann, U. (Rick), Hendi, A., Olearczyk, J., & Al-Hussein, M. (2010). An Integrated System to Select, Position, and Simulate Mobile Cranes for Complex Industrial Projects. *Construction Research Congress 2010*, 267–276. https://doi.org/10.1061/41109(373)27

Lei, Z. (2014). *Automated Simulation Model for Crane Motion Planning in Heavy Industrial Projects* (Vol. 1) [PhD Thesis University of Alberta]. https://doi.org/10.1017/CBO9781107415324.004

Lei, Z., Taghaddos, H., Hermann, U., & Al-Hussein, M. (2013). A methodology for mobile crane lift path checking in heavy industrial projects. *Automation in Construction*, *31*, 41–53. https://doi.org/10.1016/j.autcon.2012.11.042

Pan, W., & Sidwell, R. (2011). Demystifying the cost barriers to offsite construction in the UK. *Construction Management and Economics*, *29*(11), 1081–1099. https://doi.org/10.1080/01446193.2011.637938

Taghaddos, H., AbouRizk, S., Mohamed, Y., & Hermann, U. (2012). Simulation-Based

Auction Protocol for Resource Scheduling Problems. *Journal of Construction Engineering and Management*, *138*(1), 31–42. https://doi.org/10.1061/(ASCE)CO.1943-7862.0000399

Chapter 2: Heuristic Method for Automated Mobile Crane Selection in Industrial Projects (Paper 1)[1]

Abstract

Lifting heavy objects on construction sites has always been challenging since many parameters (e.g., module weight and size, project duration, safety, and cost) are involved. Heavy lift planning influences the whole project, especially the project's cost, scheduling, and safety. Proper crane selection as a part of heavy-lift planning has been made by engineers using tabulated crane capacity charts for each module. Capacity charts are organized with details of the cranes (e.g., the length and the angle of the main boom, lifting radius, jib length and angle, and super-lift capacity), making the crane selection process more complicated. This paper proposes a heuristic approach that utilizes machine learning techniques and structured query languages to select the proper crane for a lifting scenario. This approach takes into account the size and weight of all the modules in the project, boom and anti-two block clearance, lifting radius, dimension and weight of the

[1] Azami, R., Lei, Z., Hermann, R., & Zubick, T. (2021). An Automated Mobile Crane Selection System for Heavy Industrial Construction Projects. CSCE 2021 Annual Conference of the Canadian Society for Civil Engineering, proceeding of general construction. Accepted on March 2021 and presented on May 26, 2021

equipment such as hook and rigging, and the cost of the crane per month. The proposed algorithm calculates the boom and anti-two block clearance in three-dimensional (3D) space rather than two-dimensional (2D), which provides more accurate results. This approach enables an automated crane selection process that helps improve planning efficiency and accuracy. Moreover, the algorithm assigns a score to each configuration based on the user input and provides the user with the ideal crane for the project. The parameters that affect the score of each configuration include the percentage capacity of the crane that has been utilized, anti-two block clearance, boom clearance, the monthly rate of the crane, mobilization and demobilization cost, project duration, and super-lift. The weight associated with each criterion has been determined by running sensitivity analysis and consulting with lift engineering experts.

Keywords

Mobile crane, Module, Off-site Construction, Heuristic, Automation

2.1 Introduction

A safer work environment, lowered energy consumption, time savings, quality improvement, and waste reduction have convinced contractors to select off-site construction methods over the past few decades rather than on-site (Han et al. 2020; Jaillon 2009; Mao et al. 2015). One of the challenges during off-site (modularized) construction is lifting modules, especially in heavy industrial projects where the numbers of modules and their dimensions are greater than in other projects. A primary method of installing modules on-site in industrial projects is to utilize mobile cranes. Mobile cranes lift the prefabricated modules from their pick location to the set point. Successful lifts require careful and accurate lift planning (Lei et al. 2013). Project costs might increase,

and schedule conflicts may arise if errors happen in lift planning. Crane location, crane selection, support system design, and mobile crane motion planning (e.g., crane lifting path planning) are all subcategories of the lift planning system (Han et al. 2015a).

In the current crane selection approach, lift engineers usually use tabular charts that are provided by the crane manufacturers, their experience of the past projects, and their engineering judgment to select the best crane for a project. However, these approaches may not lead to the best solution and sometimes fail, which substantially increases the project's cost. In this paper, the authors reviewed the different methods presented by previous researchers for crane selection and proposed an algorithm for selecting the ideal mobile crane by implementing heuristic techniques. This paper is organized as follows: Section 2.2 provides a brief review of the most relevant literature; Section 2.3 describes the methodology adopted in the current study; Section 2.4 provides a detailed case study and validation, and Section 2.5 presents the conclusions of this study.

2.2 Literature Review

Mobile crane planning can be categorized into different subtasks such as (1) crane type and operation location selection (U. Hermann et al. 2010; Huang, Wong, and Tam 2011; Safouhi et al. 2011; Sawhney and Mund 2002), (2) crane lift path planning (Ali, Varghese, and Babu 2005; Chang, Hung, and Kang 2012; Sawhney and Mund 2002), (3) crane productivity improvement and equipment design (Hasan et al. 2010, 2013; Lei et al. 2015), and (4) visual simulation of crane operations (Han et al. 2020; Kang and Miranda 2006; Shih-Chung, Chi, and Miranda 2009; Sydora et al. 2020). Among these, crane

planning components, crane type, and operation location selection aim to determine the most suitable crane under given constraints.

Crane manufacturers produce capacity charts for cranes, from which cranes are selected to operate on construction sites. These capacity charts consist of different types of information based on the lift method. Examples of this information are boom lengths, boom angles to the ground, lifting radii, heights, jib lengths, and offsets from the main boom centerline. Different lifting radii result in different lifting capacities for a crane with a given main boom/jib length. Adding a new section to a lattice boom or extending a hydraulic boom can change the boom length (Al-Hussein, Alkass, and Moselhi 2005). The crane selection process is prone to errors because a typical industrial project usually consists of more than 100 modules, and for each module, a lift plan must be prepared (Lei 2014).

Selecting the crane type and the pick location based on the heaviest module, the largest lift radius, and the lift engineers' experience are alternative methods to use for lift planning. The methods above are neither time-saving nor cost-effective. Therefore, researchers have introduced several methods for selecting and locating the most suitable crane for each project (Safouhi et al. 2011). Single-crane location optimization modeling, which considers the total transportation cost between the crane and construction supportive facilities as the main criterion, was introduced by Rodriguez-Ramos and Francis (Rodriguez-Ramos and Francis 1983). Based on factors such as the site condition, building design, economy, capability, and safety, Hanna and Lotfallah (2004) have introduced a fuzzy logic approach. However, their method lacks the crane model or a specific configuration.

An artificial intelligence technique utilizing a neural network was used by Sawhney and Mund (2002) to develop a crane type selection tool called IntelliCranes. Their method can provide both the type and model of the crane for a project. However, they did not consider the duration of the project or the associated cost with each crane in their research. Researchers have integrated other techniques such as 3D visualization to validate the crane selection results and provide a visual representation. For instance, Al-Hussein et al. (2005) and Moselhi et al. (2004) developed algorithms to select the optimum crane considering the crane capacity and visualize the results using 3D animation.

Wu et al. (2012) developed an algorithm for selecting mobile cranes; similar to previous works, many factors were considered in their studies, such as the lifting capacity and geometrical characteristics of the crane. In addition, they have considered the ground bearing capacity, and then the selection is incorporated into a 3D computer-aided system for simulation, design, and rigging calculation purposes. Since each crane manufacturer publishes the cranes' capacity charts and geometrical characteristics in different formats, utilizing these charts in the crane selection process makes it more time-consuming.

Al-Hussein et al. (2000) introduced a comprehensive database called D-CRANE to support an efficient selection of cranes. Another widespread problem that arises during modular construction is the dynamic environment of the job site, which needs to be considered during lift planning. In this regard, Olearczyk et al. (2012) have developed an evolutionary algorithm that reacts to dynamic changing site conditions. Moreover, they have extended the boom clearance algorithm to consider the situation where the crane

boom position is not in the normal plane but the inclined roof shape of the building structure.

By reviewing the literature, the authors realized the gap in the crane selection process as the lack of a decision support system that proposes the most suitable mobile cranes by taking into account the projects' safety and economical parameters simultaneously. The results of this research are beneficial for almost everyone involved in the project since it lowers the cost, shortens the schedule, and improves the safety of heavy industrial projects. Moreover, it can be integrated with another heavy-lift application called Automated Crane Planning and Optimization (ACPO) that has been developed for PCL to automate the process of checking the capacity of a crane and the clearances of the crane (Taghaddos, Hermann, and Abbasi 2018).

2.3 Methodology

The proposed algorithm is shown as a flow chart in Figure 3. The application starts with connecting to the database, which is an MS Access database in this research. However, this database can be any relational database such as Structured Query Language (SQL) or Microsoft Excel. After connecting to the database, the user can interact with the project’s details and the modules already defined in the database or create a new project and add new modules. The main process of the algorithm has been elaborated in the following sections, and all the associated codes written in the C# programming language are presented in Appendix I.

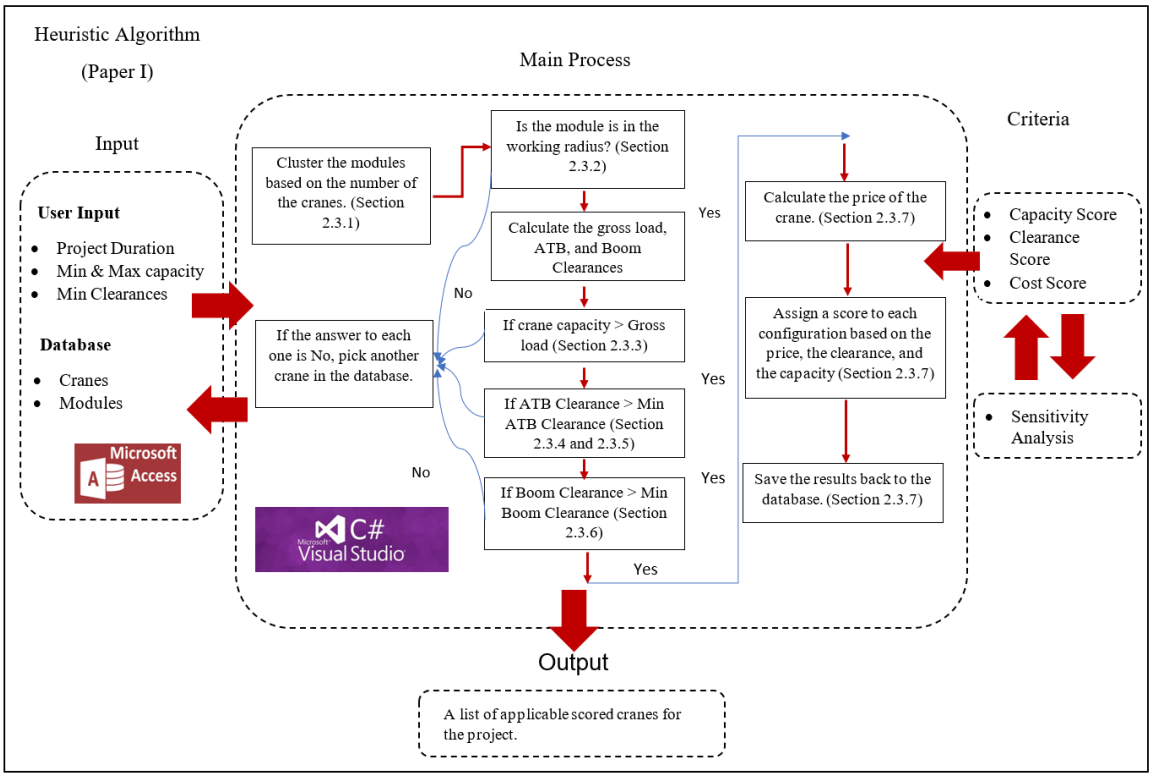


Figure 3: Methodology of the proposed algorithm

2.3.1 Module Clustering

Data clustering has played a significant role in data analysis, and various definitions have been proposed in the literature. One view regards clustering as the segmentation of a heterogeneous population into a number of more homogenous subgroups. Another definition, adopting a bottom-top view, defines clustering as finding groups in a dataset "by some natural criterion of similarity" (Estivill-Castro 2002). Clustering algorithms are considered unsupervised algorithms; the objective is to group a given collection of unlabeled patterns into meaningful clusters. In contrast, supervised algorithms such as classifications are provided with a collection of labeled patterns, and the problem is to label a newly encountered yet unlabeled pattern (Keogh et al. 2001).

Various clustering algorithms have been developed in data science, some of which are more popular due to their advantages. K-means clustering, Mean-Shift Clustering, Density-Based Special Clustering of Applications with Noise (DBSCAN), Expectation-Maximization clustering, and Agglomerative Hierarchical clustering are examples of popular clustering algorithms. All the clustering algorithms mentioned before were considered as a potential option for this research. K-means algorithm, however, was selected after carefully reviewing the advantages and disadvantages of this technique. The time complexity of K-means is O(nkl), and its space complexity is O(k + n), where n is the number of patterns, k is the number of clusters, and l is the number of iterations taken for the algorithm to converge. Although all the algorithms have some ambiguity in some data when clustered, the performance and accuracy of K-means are higher than the other algorithms, especially for large datasets (Abu Abas 2008). Therefore, the K-means algorithm has been selected for this research to cluster the modules.

After implementing this algorithm, the user can cluster the modules based on the number of cranes available on the job site. The first input parameter for the algorithm is the sum of the height of the module and the set elevation, which indicates the overall height that the module needs to be lifted. The second input parameter is the moment generated by the module, which is determined by multiplying the module's weight by the set radius. Finally, the algorithm minimizes an objective function known as means squared error function (see Eq. 1) given as follows:

$$J(V)= \sum_{i=1}^{c} \sum_{j=1}^{c_i} (||x_i - v_i||)^2 \quad (1)$$

Where: $||x_i - v_i||$ is the Euclidean distance between x_i and v_j

c_i is the number of data points in the i^{th} cluster.

c is the number of cluster centers

Figure 4 provides the flowchart for the algorithm used to cluster the modules based on the procedure described above.

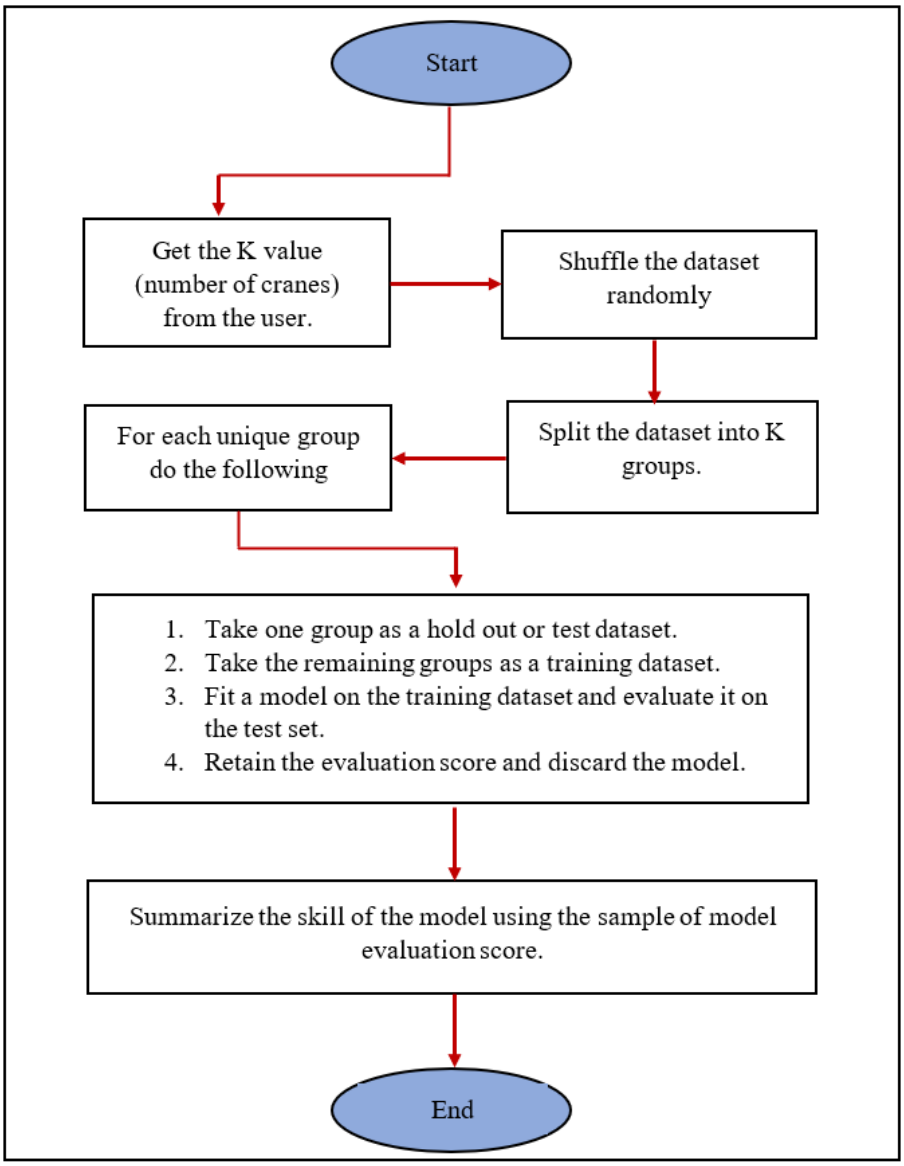


Figure 4: K-means algorithm flowchart

2.3.2 Working Radius

After clustering the modules into different groups, the next step is to check the working radius of the crane to confirm if the cane could reach the lift points and set points (see Figure 5). The working radius has a direct effect on the crane capacity. The boom tip must reach both the lift point and the set point to lift and set the module.

Therefore, the working radius, W_R, must be greater than the maximum of lift and setpoint radius as shown in Eq.2

$$W_R \geq W_R^{'} \quad (2)$$

Where:

$W_R^{'} = \max (R_L, R_S)$

R_L = distance from the center of rotation of the crane to the lifting point

R_S = distance from the center of rotation of the crane to the set point

The required data is retrieved from the database.

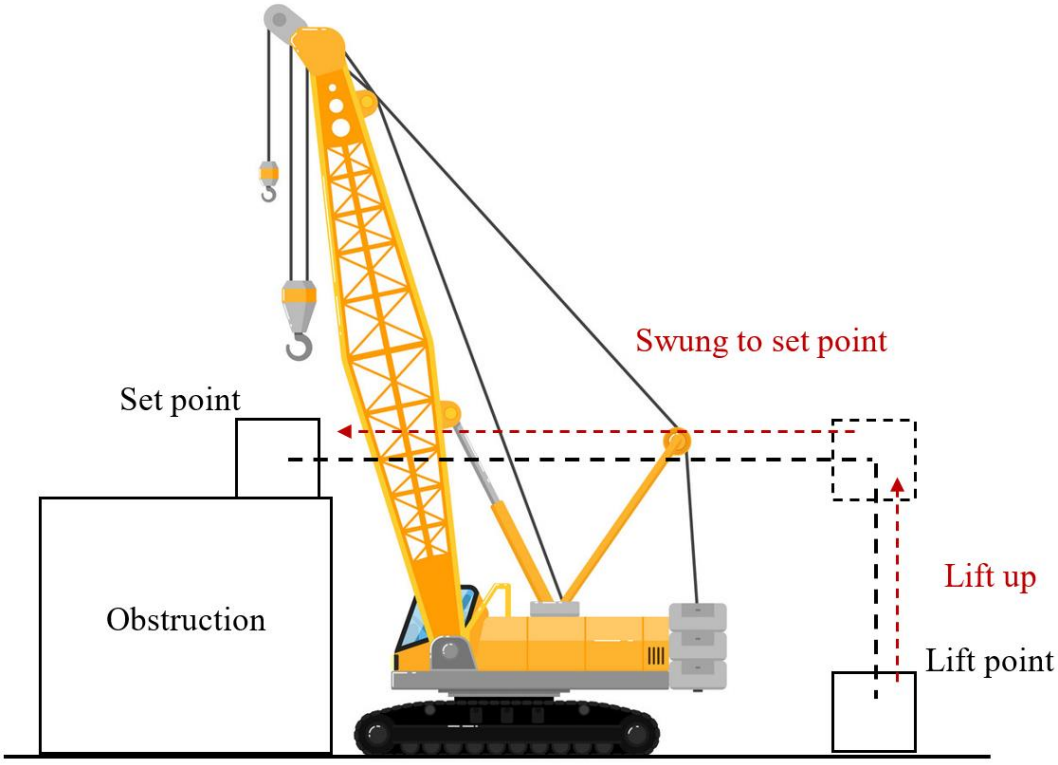


Figure 5: Lift point and set point

2.3.3 Crane Load Capacity Check

The following criterion to check is the gross load, which consists of the weight of the module, the associated rigging, and the hook block. Gross load must not exceed the reduced crane capacity, calculated satisfying Eq. 3 (see Figure 6). The values of these weights are imported from the database or captured through the interface. Section 2.3.3 through 2.3.5 in this paper utilize a methodology similar to the one that was used in this paper (Wu et al. 2011).

$$SF(CC) \geq GL = W_m + W_h + W_r \quad (3)$$

Where:

SF = safety factor

CC = the total crane capacity provided by the crane manufacturer

GL = the gross load

W_m = the module weight

W_h = the hook block weight

W_r = the rigging weight stored in the database.

The term SF in Eq. 3 limits the crane operation to a certain percentage of its capacity, which should be lower than 1. The safety factor varies from project to project, and the user could change it through the interface.

2.3.4 Lifting Height

Lifting height, another influential factor affecting the module lifting process, is calculated for different configurations based on the following methods. Lifting height is also required for checking the clearances (Figure 6).

2.3.4.1 Main Boom Configuration

The height of the main boom shown in Figure 6A has been calculated as follows, while the crane does not consist of any type of jib (see Eq. 4 and Eq. 5).

$$H_1 = \sqrt{\left(L_1^2 + D_1^2\right) - (R-X)^2} \tag{4}$$

$$H = H_1 + Y \tag{5}$$

Where:

X = the distance between the main boom foot and the center of rotation of the crane

Y = the distance between the main boom foot and the ground

L_1 = the length of the main boom

R = the distance between the center of rotation and the lift/set point

D_1 = the main boom offset

The required data is retrieved from the database.

2.3.4.2 Main Boom with Fixed Jib

A Jib is usually added to the main boom to increase the lifting height. Fixed jib refers to a type of jib in which the angle between the main boom and the jib is fixed (see

Figure 6B). However, the angle between the main boom and the horizental can change within a certain range, as defined in Eq. 6 through Eq. 9.

$$\gamma = 180\text{-}\theta\text{-}\tan^{-1}\frac{D_2}{L_2} \tag{6}$$

$$T = \sqrt{L_1^2+\left(L_2^2+D_2^2\right)\text{-}2\times L_1\times\sqrt{L_2^2+D_2^2}\times\cos\gamma} \tag{7}$$

$$H_1 = \sqrt{T^2\text{-}(R\text{-}X)^2} \tag{8}$$

$$H = H_1+Y \tag{9}$$

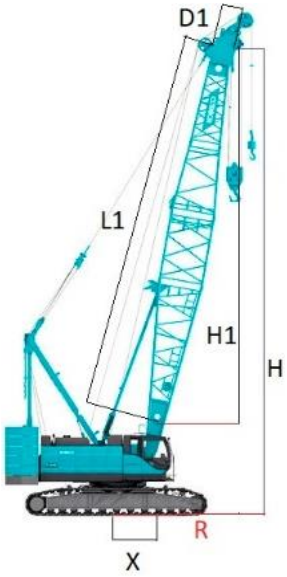
Where:

D_2 = represents the vertical offset of the fixed jib

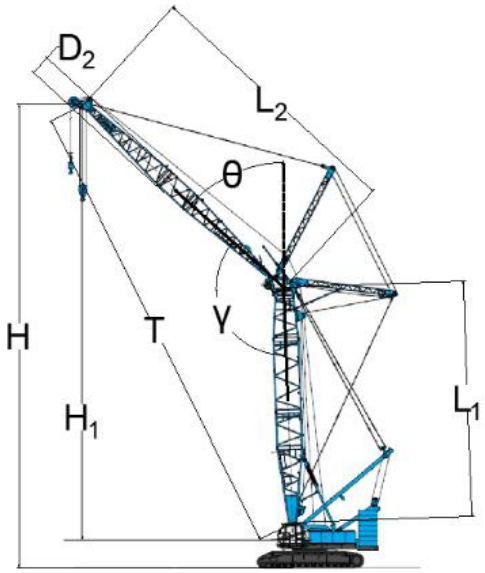
L_2 = fixed jib length

Y = distance between the main boom foot and the ground

The required data is retrieved from the database.



(A) Main boom configuration (photo courtesy of kobelco-cranes.com)



(B) Fixed jib configuration (photo courtesy of deldencranes.co.uk)

Figure 6: Main boom and fixed jib configuration of mobile cranes

2.3.4.3 Main Boom with Luffing Jib

A luffing jib is installed on the main boom for the same purpose as a fixed jib. The difference is that in this situation, the angle between the main boom and the jib is not fixed, and the angle between the main boom and the horizon is fixed (see Figure 7A). In this case, lifting height is calculated as follows (Eq. 10 through Eq. 14):

$$R_1=L_1\times\cos\alpha \tag{10}$$

$$H_1=L_1\times\sin\alpha \tag{11}$$

$$R_3=R-X-R_1 \tag{12}$$

$$H_3=\sqrt{\left(L_3^2+D_3^2\right)-R_3^2} \tag{13}$$

$$H=Y+H_1+H_3 \tag{14}$$

Where:

α = main boom angle

L_3 = luffing jib length

D_3 = vertical offset of the luffing jib

The required data is retrieved from the database.

2.3.5 Anti-Two Block Clearance

Anti-Two bock (ATB) is a piece of equipment hanging from the tip of the boom and prevents the hook block from being hoisted to the boom tip. At the moment that the module touches the ATB, the crane stops working Figure 7B. Therefore, a minimum clearance must be considered between the module and the ATB using Eq. 15.

$$H_{hc}=H + A - ME + GE - H_R - H_M - H_{ATB} \quad (15)$$

Where:

H_{hc} = the clearance between the ATB and the hook block

H = lifting height

A = the distance between the center of rotation of the crane structure and the ground surface

ME = the module elevation

GE = the ground elevation

H_R = the height of the rigging

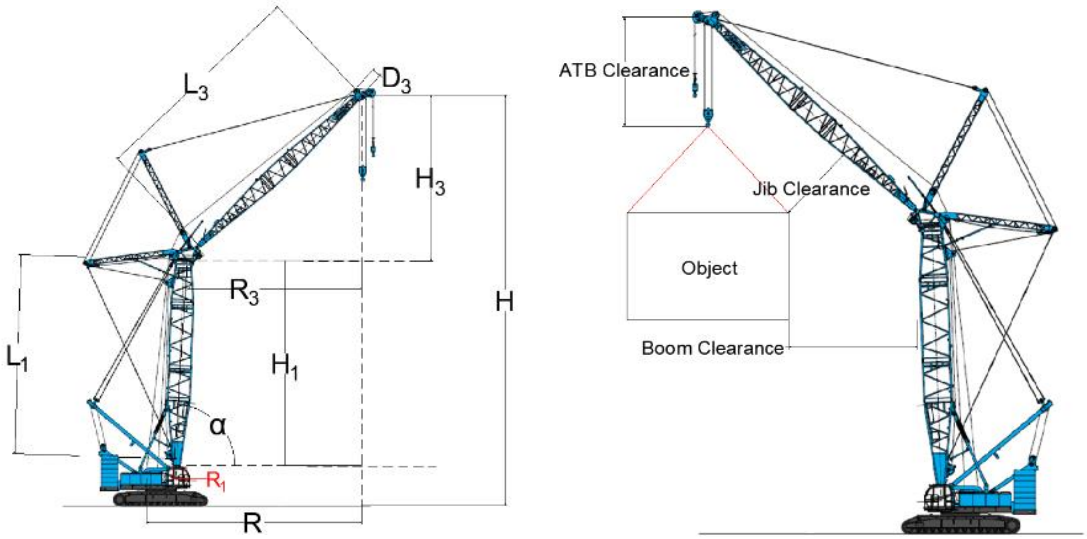
H_M = the module height

H_{ATB} = the distance between the tip of the boom and the ATB devise

All the values are available in the database except the lifting height, which was calculated in the previous steps.

2.3.6 Boom and Jib Clearance

The minimum distance between the module and the boom or the jib is determined in three-dimensional (3D) space (see Figure 7B). The user can define a minimum clearance through the interface, and the computed value through the algorithm is compared with the user input value by the algorithm. The minimum distance between the module and the crane occurs when the module is at its highest elevation. Therefore, the maximum height needs to be determined through the process explained in the previous sections.



(A) Luffing jib configuration (photo courtesy of deldencranes.co.uk)

(B) Boom and jib clearance (photo courtesy of deldencranes.co.uk)

Figure 7: Configuration and clearances of mobile cranes

Next, the three-dimensional (3D) coordinates of two points on the module (see Eq. 16 and Eq. 17) and two points on the main boom or the jib (see Eq. 18 and Eq. 19) are determined. Each pair of coordinates is used to pass a line through them. Having

determined the coordinates and the lines passed through them, the minimum distance between the module and the boom/jib is the length of the line segment perpendicular to both lines (see Eq. 20 and Eq. 21). The algorithm determines the perpendicular line and its size (see appendix I for the details) by computing the dot product of the two lines passing through the module and the boom where the dot product equals zero (see Eq. 22 and Eq. 23).

$$P_1 = P_{ax} + P_{ay} + P_{az} \tag{16}$$

$$P_2 = P_{2x} + P_{2y} + P_{2z} \tag{17}$$

$$P_3 = P_{3x} + P_{3y} + P_{3z} \tag{18}$$

$$P_4 = P_{4x} + P_{4y} + P_{4z} \tag{19}$$

$$P_a = P_{ax} + P_{ay} + P_{az} \tag{20}$$

$$P_b = P_{bx} + P_{by} + P_{bz} \tag{21}$$

$$(P_a - P_b) \cdot (P_2 - P_1) = 0 \tag{22}$$

$$(P_a - P_b) \cdot (P_4 - P_3) = 0 \tag{23}$$

Where:

P_1 and P_2 = coordinates of the two points on the module

P_3 and P_4 = coordinates of two points on the boom/jib

P_a = coordinate of a point on the line that passes through P_1 and P_2

P_b = coordinate of a point on the line passing through P_3 and P_4 (see Figure 8).

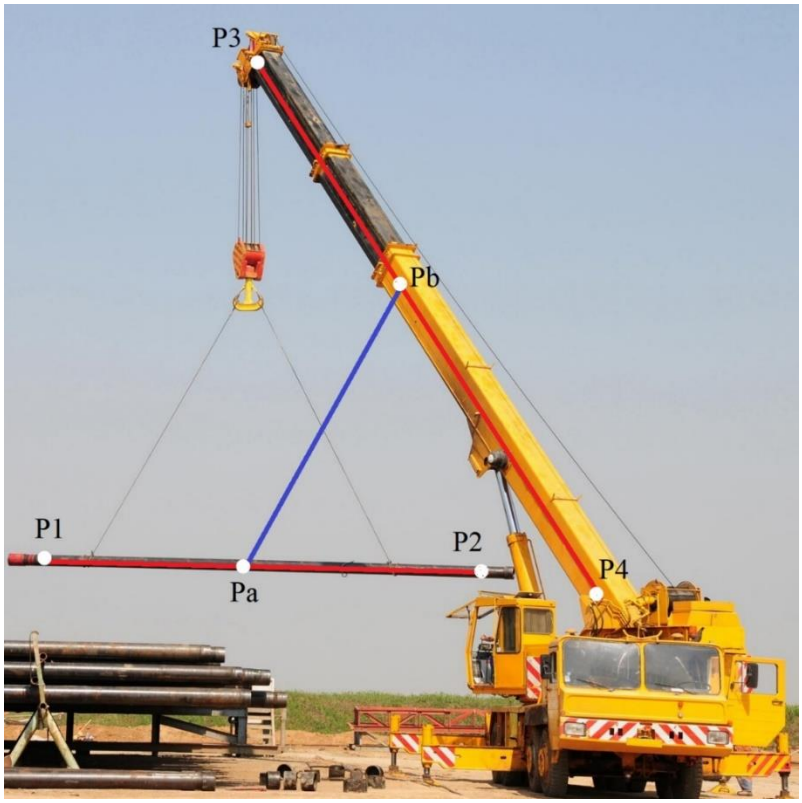


Figure 8: Boom clearance (photo courtesy of bigrentz.com)

2.3.7 Score and Rank the Results

Based on the methodology explained in sections 2.3.1 through 2.3.6, the algorithm displays the result to the user, including all passed and failed configurations and the reasons for each failure. However, this result may not be helpful for the user as it does not indicate which crane is better than the other ones for the project. To tackle this problem, the algorithm first finds the cranes which can lift all the modules selected by the user, and then based on the criteria in Table 1, it scores each configuration. Within the crane selection process for each project, there are quite a few parameters that need to be taken into account, such as cost, project duration, super-lift capacity, boom/jib clearance, ATB clearance, and the percentage capacity, all of which have been considered in this

algorithm. Each parameter receives a weight based on its impact on the project (see Table 1). For instance, the crane's cost has been considered the most important criterion for this case study. Consequently, the impact factor for this criterion has been set to 70 for the minimum price and 35 for the maximum price. In other words, the crane with the minimum price achieves 70 points, and the crane with the maximum price gets 35 points. Any crane with a price between the minimum and maximum prices would get the point between 35 to 70 based on a linear distribution. This criterion helps the least expensive cranes get higher points, increasing their chances of being selected for a project. It is important to mention that the weights associated to each parameter are defined by the user for each project.

The following criterion is the percentage capacity of the crane that has been utilized while lifting modules. This parameter helps to find the right crane from the capacity perspective. In other words, a crane that is not operating by its maximum capacity or a crane that is not unreasonably oversize for lifting a module. For this parameter, any crane with a capacity equal to or lower than the minimum crane capacity defined by the user would get five points. A crane with a capacity equal to the optimum crane capacity or the maximum crane capacity would get the optimum crane capacity score or the maximum crane capacity score, defined in this research as 20 and 10 points, respectively.

Clearance is the next criterion that was considered for ranking the results. All the cranes must have the minimum boom and ATB clearances to be regarded as an applicable crane, and they would not receive any point for the minimum clearances. However, for clearances between the minimum and the optimum value, each crane would get points

between zero and the optimum score through linear distribution. Any clearances equal to or greater than the optimum value would earn the optimum clearance points defined in this research as five points.

Although the super-lift can boost the crane's capacity, it may increase its monthly rate and the number of workers needed for operating the crane. Therefore, the last parameter was related to the super-lift existence. If a crane configuration carries a super-lift, the algorithm assigns zero points to the crane configuration, otherwise ten points. This criterion helps to increase the chances of crane configurations without a super-lift for being selected.

After consulting with crane experts and running sensitivity analysis to determine the effect of each parameter on the crane selection, the following weights (see Table 1) have been assigned to each parameter as the default values. However, the user could change anyone for each project. All the required information in this step (i.e., monthly rate, mobilization and demobilization cost, and project duration) is retrieved from the database. Since the monthly rate of the cranes could be affected by economic situations; therefore, information such as the monthly rate should be updated constantly. The following flowcharts illustrate how the scoring system works (see Figure 9 and Figure 10). More information and examples of the scoring system are provided in the case study section.

Table 1: Ranking parameters and weights

Criteria	Weights
Minimum Crane Capacity	10
Optimum Crane Capacity	20
Maximum Crane Capacity	5
Minimum Boom Clearance	0
Optimum Boom Clearance	5
Minimum Anti–Two Block Clearance	0
Optimum Anti-Two Block Clearance	5
Minimum Price	70
Maximum Price	35
Super - lift	10

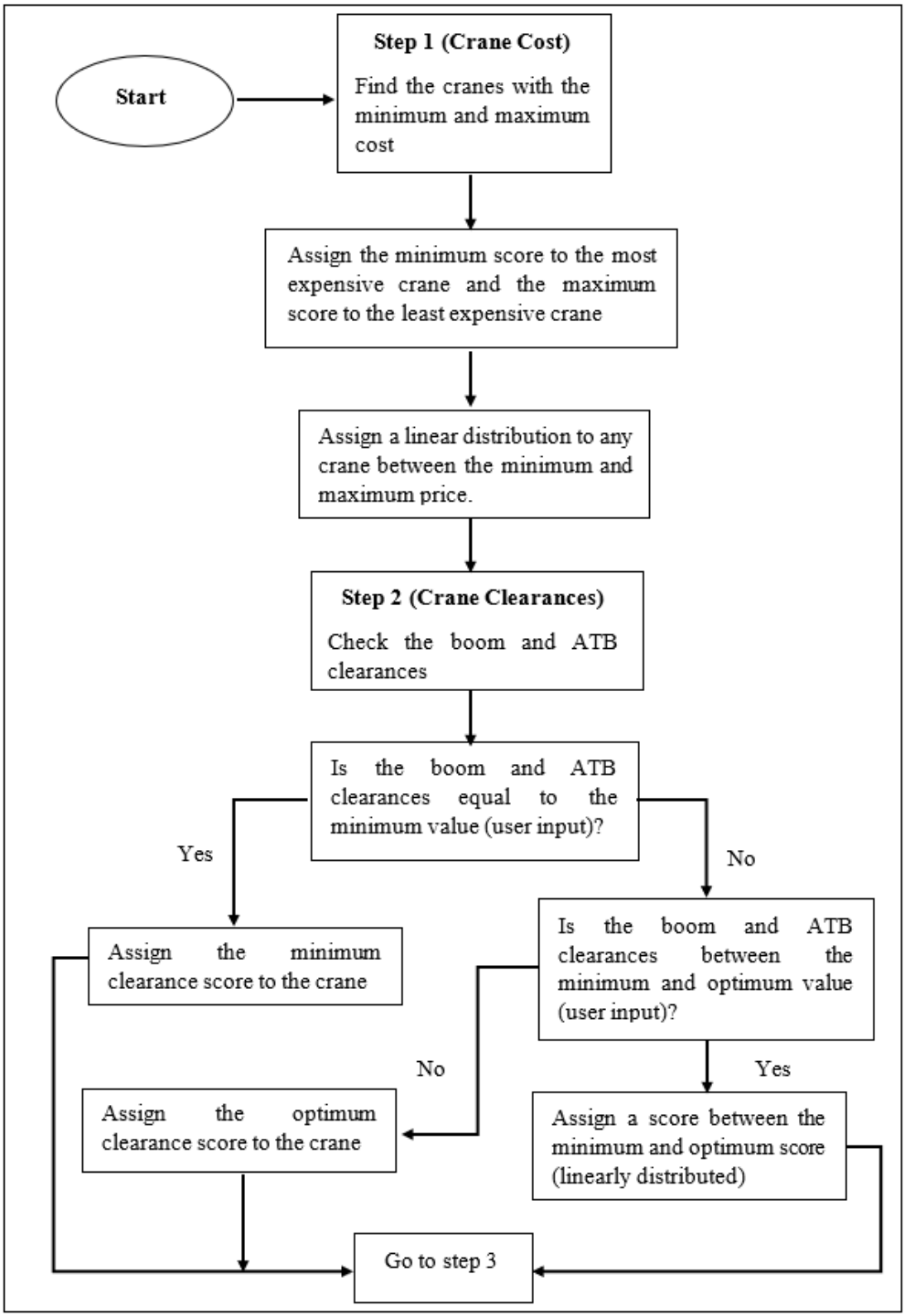


Figure 9: Steps one and two of the scoring system

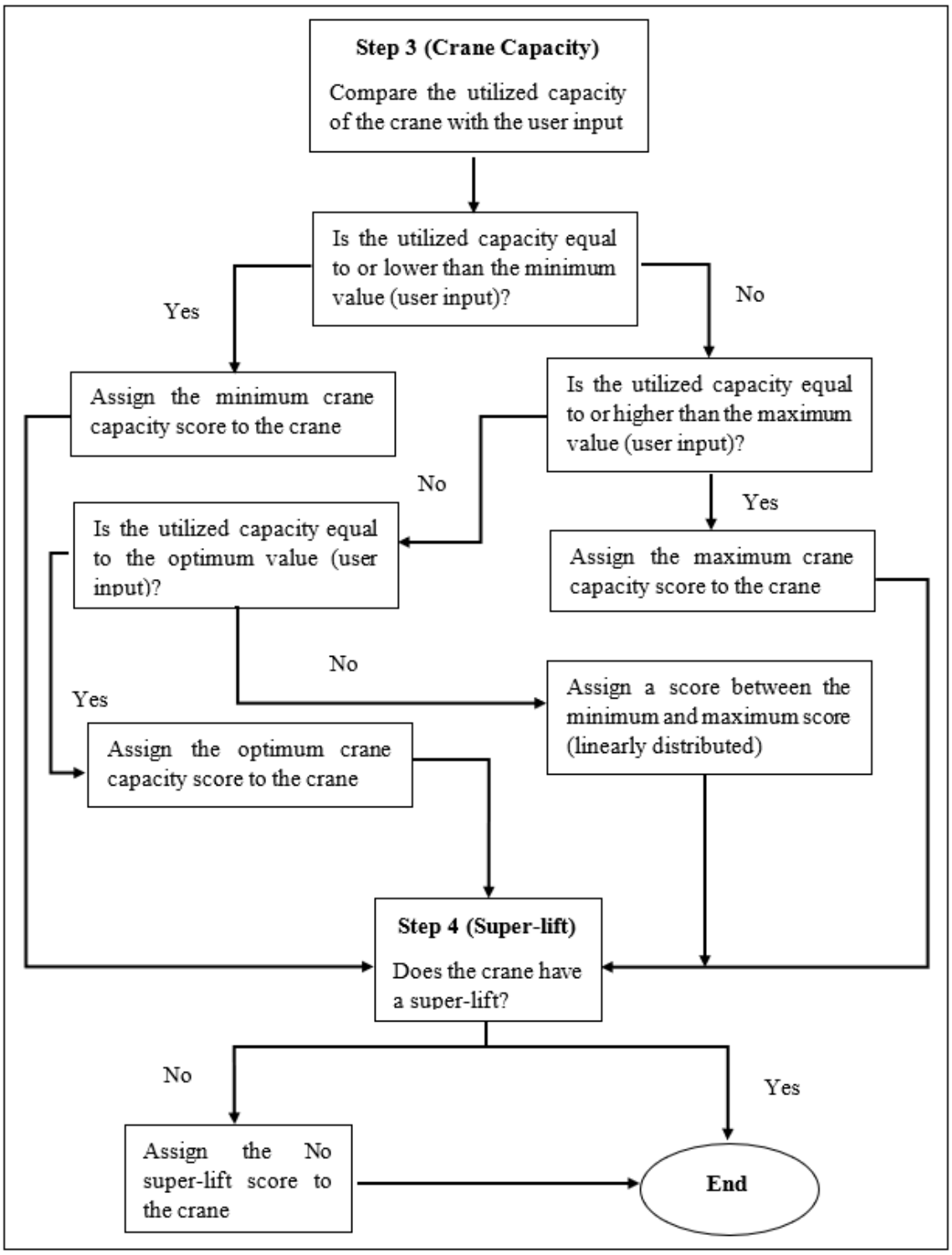
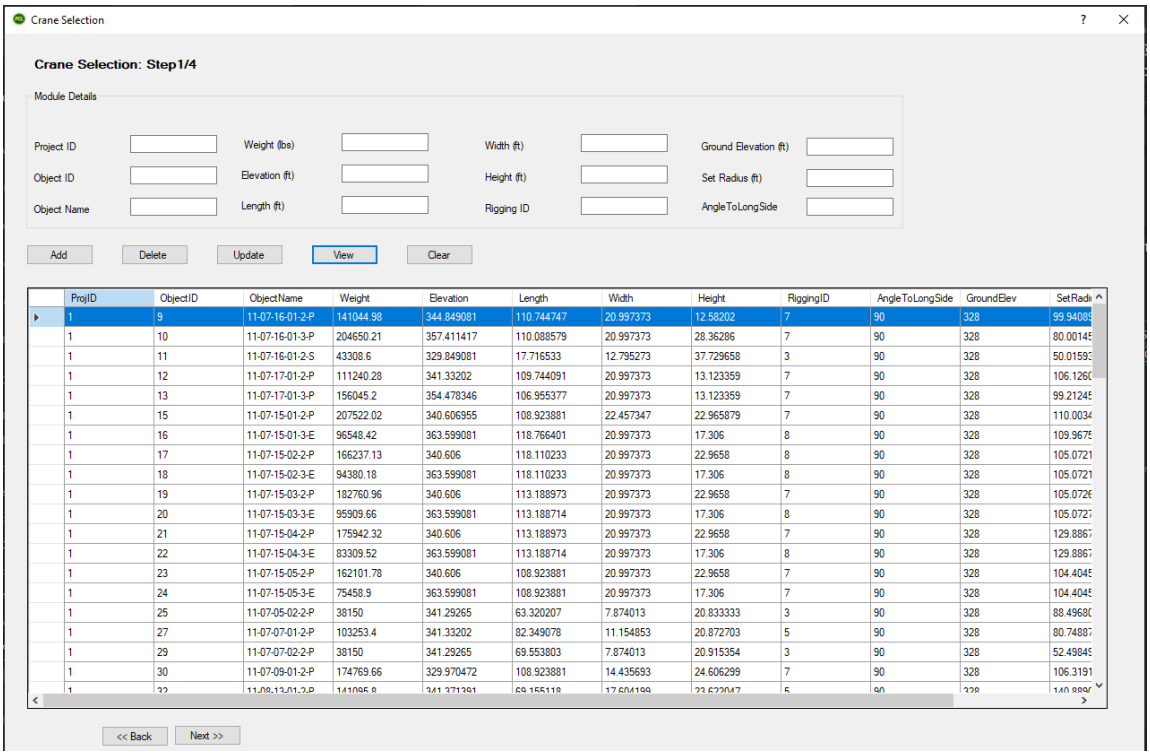


Figure 10: Steps three and four of the scoring system

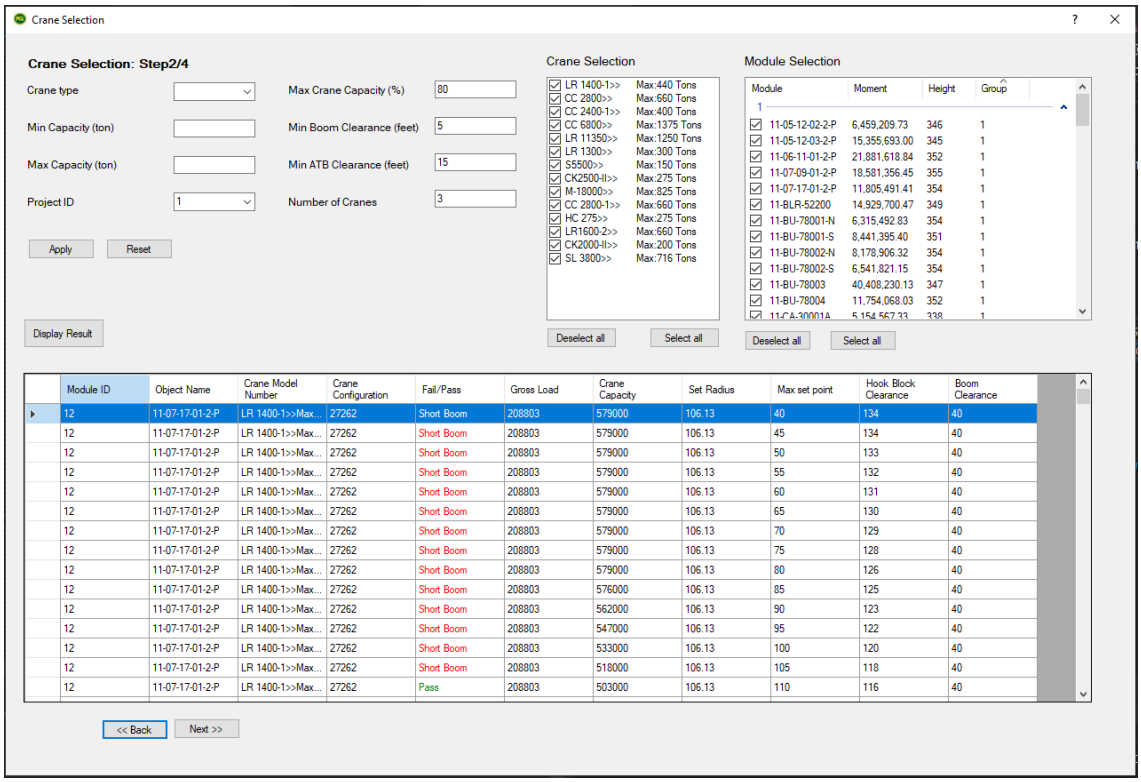
2.4 Case Study

An actual case is presented in this section to illustrate the effectiveness and practicality of the proposed algorithm. This case involves 20 different modules, which have been predefined in the database. In addition, 53 different models of mobile cranes consisting of nearly 6,000 different configurations were available for this project to be selected, all of which are stored in the database, including the geometrical details, capacities, and prices. Figure 11A shows the first step of the application where the user would add/modify the modules using the available options.

In the second step (see Figure 11B), the user would limit the search range using the tools provided. After filling out all the textboxes and the dropdown lists, by clicking on the “Apply” button, all the available cranes and modules associated with the selected criteria would be shown to the user in two different checklist boxes. The user would choose any combinations of the cranes and the modules provided in the checklist boxes. The module selection checklist box in the second step shows how the modules are clustered into different groups using the moment and height as the algorithm's inputs. The number of groups is associated with the number of cranes the user would provide for each project. The moment as the first input is determined by multiplying the weight of each module by the set radius. Crane selection based on the moment generated by the module would be easier for lift engineers than crane selection based on the module’s weight and the set radius individually because the selected crane must overcome the module’s moment. In addition to the moment, height is another input. The total height is defined as the difference between the setpoint elevation and the ground elevation plus the module’s height.



(A) Modules defined in the database



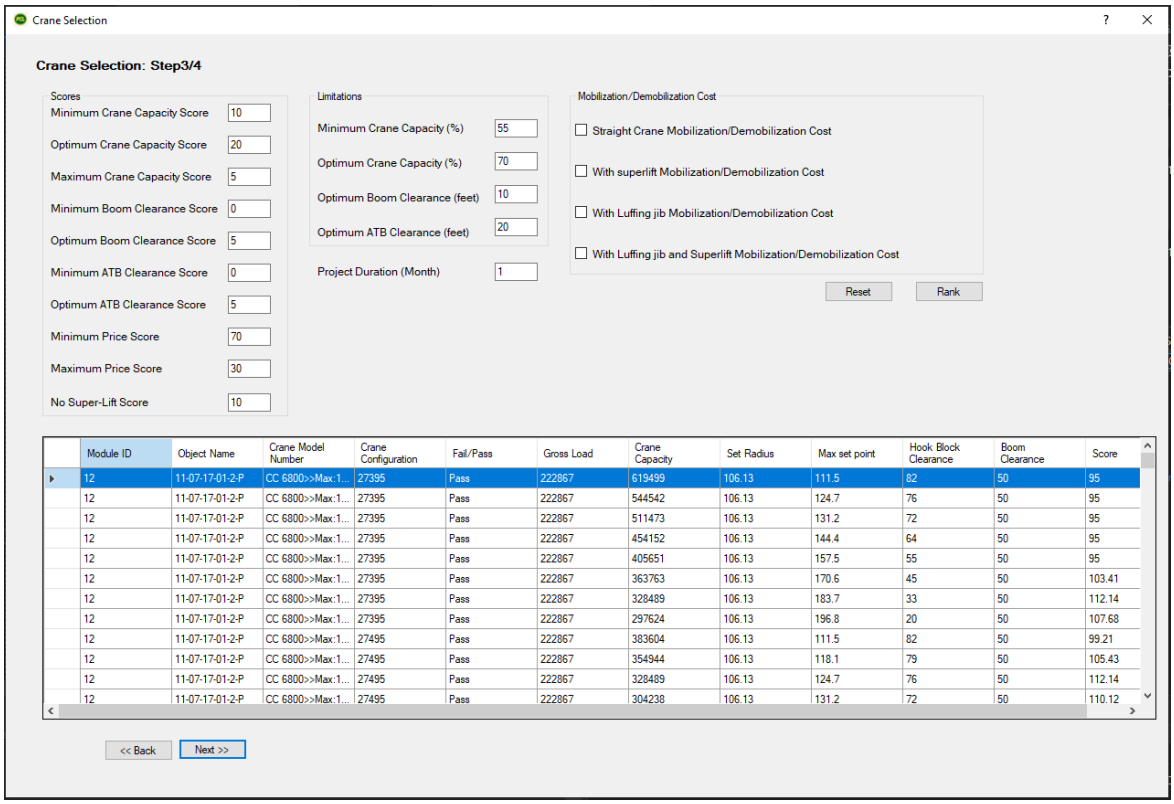
(B) Step two - Module selection

Figure 11: Case study results (Steps one and two)

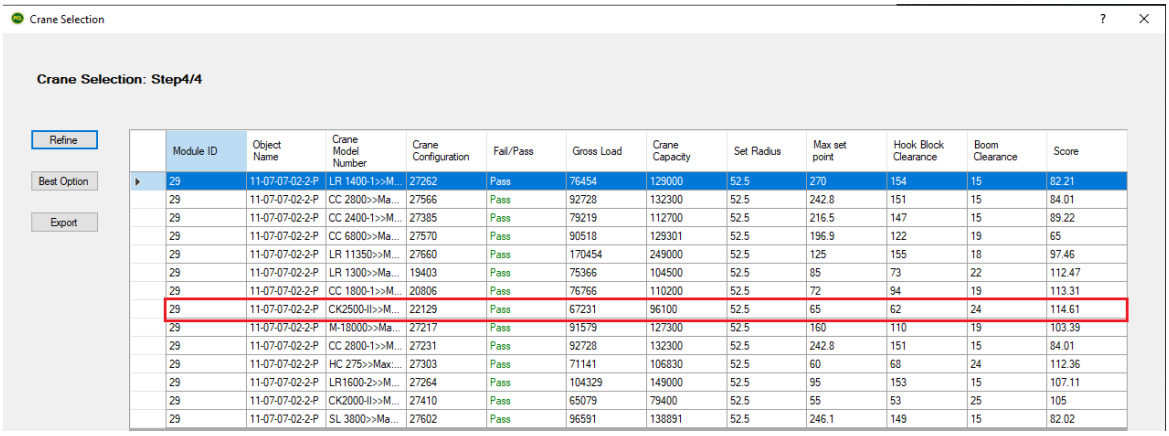
By clicking on the “Display Results,” the preliminary results are shown as a table to the user. This table shows all the details of the cranes and the modules selected by the user. One of the main problems that may arise with this table is that the user would not realize which crane is better than the other ones, as this table shows only “Fail” or “Pass.” To resolve this issue, the user could move forward to the next step, where the scoring system is implemented. Ten criteria have been considered and presented in Figure 12A to attain the desired results. The minimum, optimum, and maximum crane capacity score has been defined to differentiate between the cranes based on their capacity in use. The maximum crane capacity threshold was defined in the second step as it was a required value for the preliminary results. The associated score was defined as 5 points (i.e., if 80% of the crane’s capacity were utilized, that crane would get 5 points). The minimum and the optimum threshold were defined in the third step as 55% and 70%, and the score associated with them are 10 and 20 points, respectively. For any crane with a utilized capacity within these ranges, the scores are distributed linearly. The next score is related to boom and ATB clearance. The threshold for the minimums and the optimums have been defined in the second and third steps, respectively. The minimum thresholds are in the second step for the same reason as the maximum crane capacity. Minimum and maximum price scores are related to the price of each crane. The algorithm first loops through all the cranes and finds the maximum and minimum costs. Then it assigns the associated score to each one, the crane with the maximum price would receive the maximum cost score, and the crane with the minimum price would receive the minimum price score. For the remaining cranes, similar to previous steps, the scores are distributed linearly.

Super–lift is an optional piece of equipment that can be added to some configurations to increase the crane's power and consequently increase the working radius of the crane. However, adding super-lift increases the crane price as it needs ground preparation, workforce, mobilization, and demobilization cost. Therefore, a textbox was provided to enter a score for cranes without a super-lift to differentiate between the cranes with and without super-lift. As the prices of the cranes in the database are per – month, a textbox was created to get the duration of using the cranes per month from the user.

The last part of this step is the cost of mobilization and demobilization of the cranes. This section allows the user to consider the cost of the mobilization and demobilization of the cranes in the projects. Moreover, the user can change the number of mobilization and demobilization occurrences, which affects the project's cost. After considering all the criteria and analyzing them, the results are printed as a table in Figure 12B. This table consists of the configurations of the cranes that received the highest score. Based on this table, the lift engineers can decide to select the configuration with the highest score.



(A) Step three – Score the results



(B) Step four – Propose the configuration with the highest score

Figure 12: Case study results (Steps three and four)

2.4.1 Validation Metrics

To validate the application's results, a comparison between the proposed cranes by the algorithm and the actual cranes that were utilized in the project was made. The details of this comparison are presented in Table 2.

Table 2: Comparison between the actual crane and the proposed crane by the algorithm

	Actual Crane	Proposed Crane
Crane Manufacturer	Liebherr	Demag
Crane model	LR 1300	CC 1800-1
Capacity	300 ton	330 ton
Boom Length	203 ft	216 ft
Super-Lift	No super-lift	No super-lift
Monthly rate	Same	Same
Lift from	Main boom	Main boom

The comparison between the result of the proposed algorithm in this research and the actual crane that lift engineers selected in the past demonstrate that achieved results are valid for several reasons that are explained as follows:

1. The crane proposed by the algorithm has a 10% higher capacity than the actual one because of the safety factor that was considered in the application to make sure that not more than 80% of the capacity of the cranes is utilized for safety purposes.
2. The details of both cranes, such as the boom length, super-lift existence, and the lifting point, are similar between the actual and the proposed crane.
3. The monthly rate of both cranes is almost the same as for the project in the case study, which makes the proposed crane a wise option to consider.
4. All the details of both cranes were discussed with professional lift engineers. Based on their knowledge and experience, the crane selected by the algorithm can

be a better option than the actual one because it has a higher capacity that can perform a safer lift.

2.5 Conclusion

This paper proposed a heuristic approach that automatically selects a mobile crane configuration in heavy industrial construction projects that satisfy users' requirements. An unsupervised machine learning technique, K-means, has been implemented to group the modules based on their moments and heights. Parameters such as the project duration, the cost of the crane, the safety, and the percentage capacity of the cranes were considered in this approach simultaneously, which was carried out in this research for the first time. The lifting capacity check and clearance check are the first two steps of the algorithm. Following these steps, each configuration that passed all the criteria receives a score. The weights that were applied to each parameter were determined based on sensitivity analysis and consulting with lift engineers. The user can change weights for each project.

This application has been developed based on relational databases such as MS Access; however, the code can be modified easily to be used with any other database, such as Microsoft Excel. Although the application has been developed for mobile cranes, it can be modified by the user to be used for other types of cranes as the application is based on a generic method. The algorithm reads the required data from the database and writes the results back to the database, and the runtime could increase if the number of modules or cranes selected by the user were increased. Finally, a comparison was made between the algorithm results and the historical data through a case study to illustrate the application's effectiveness. It was shown that the algorithm could determine even a better solution than the actual selected mobile crane. This application can be improved for

future research to incorporate all types of cranes such as tower and derrick cranes. A 3D visualization can be added to the application for clash detection to increase the project's safety by decreasing uncertainty. In addition, the source code can be optimized to reduce the algorithm's runtime and provide results in a shorter time than the current runtime. This application can be a valuable tool for the engineers, the crew on the job site, and the stakeholders. It provides a tool through which the engineers can look for safety increases and cost and schedule decrease.

Appendix I: C# Source Code

Module Clustering

```
private static double[][] Normalized(double[][] rawData)
        {
            double[][] result = new double[rawData.Length][];
            for (int i = 0; i < rawData.Length; ++i)
            {
                result[i] = new double[rawData[i].Length];
                Array.Copy(rawData[i], result[i], rawData[i].Length);
            }
            for (int j = 0; j < result[0].Length; ++j)
            {
                double colSum = 0.0;
                for (int i = 0; i < result.Length; ++i)
                    colSum += result[i][j];
                double mean = colSum / result.Length;
                double sum = 0.0;
                for (int i = 0; i < result.Length; ++i)
                sum += (result[i][j] - mean) * (result[i][j] - mean);
                double sd = sum / result.Length;
                for (int i = 0; i < result.Length; ++i)
                    result[i][j] = (result[i][j] - mean) / sd;
            }
            return result;
        }

        private static int[] InitClustering(int numTuples, int
        numClusters, int seed)
        {
            Random random = new Random(seed);
            int[] clustering = new int[numTuples];
            for (int i = 0; i < numClusters; ++i)
                clustering[i] = i;
            for (int i = numClusters; i < clustering.Length; ++i)
                clustering[i] = random.Next(0, numClusters);
            return clustering;
        }
        public static int[] Cluster(double[][] rawData, int
        numClusters)
        {
            double[][] data = Normalized(rawData);
            bool changed = true; bool success = true;
            int[] clustering = InitClustering(data.Length, numClusters,
             0);
            double[][] means = Allocate(numClusters, data[0].Length);
            int maxCount = data.Length * 10;
            int ct = 0;
            while (changed == true && success == true && ct < maxCount)
            {
                ++ct;
                success = UpdateMeans(data, clustering, means);
                changed = UpdateClustering(data, clustering, means);
```

```
        }
        return clustering;
    }
    private static bool UpdateMeans(double[][] data, int[]
    clustering, double[][] means)
    {
        int numClusters = means.Length;
        int[] clusterCounts = new int[numClusters];
        for (int i = 0; i < data.Length; ++i)
        {
            int cluster = clustering[i];
            ++clusterCounts[cluster];
        }
        for (int k = 0; k < numClusters; ++k)
            if (clusterCounts[k] == 0)
                return false;
        for (int k = 0; k < means.Length; ++k)
            for (int j = 0; j < means[k].Length; ++j)
                means[k][j] = 0.0;
        for (int i = 0; i < data.Length; ++i)
        {
            int cluster = clustering[i];
            for (int j = 0; j < data[i].Length; ++j)
                means[cluster][j] += data[i][j]; // accumulate sum
        }
        for (int k = 0; k < means.Length; ++k)
            for (int j = 0; j < means[k].Length; ++j)
        means[k][j] /= clusterCounts[k]; // danger of div by 0
        return true;
    }
    private static double[][] Allocate(int numClusters, int
    numColumns)
    {
        double[][] result = new double[numClusters][];
        for (int k = 0; k < numClusters; ++k)
            result[k] = new double[numColumns];
        return result;
    }
    private static bool UpdateClustering(double[][] data, int[]
    clustering, double[][] means)
    {
        int numClusters = means.Length;
        bool changed = false;
        int[] newClustering = new int[clustering.Length];
        Array.Copy(clustering, newClustering, clustering.Length);

        double[] distances = new double[numClusters];
        for (int i = 0; i < data.Length; ++i)
        {
            for (int k = 0; k < numClusters; ++k)
                distances[k] = Distance(data[i], means[k]);
            int newClusterID = MinIndex(distances);
            if (newClusterID != newClustering[i])
            {
                changed = true;
```

```
                    newClustering[i] = newClusterID;
                }
            }
            if (changed == false)
                return false;
            int[] clusterCounts = new int[numClusters];
            for (int i = 0; i < data.Length; ++i)
            {
                int cluster = newClustering[i];
                ++clusterCounts[cluster];
            }
            for (int k = 0; k < numClusters; ++k)
                if (clusterCounts[k] == 0)
                    return false;
            Array.Copy(newClustering, clustering,
            newClustering.Length);
            return true; // no zero-counts and at least one change
        }
        private static double Distance(double[] tuple, double[] mean)
        {
            double sumSquaredDiffs = 0.0;
            for (int j = 0; j < tuple.Length; ++j)
                sumSquaredDiffs += Math.Pow((tuple[j] - mean[j]), 2);
            return Math.Sqrt(sumSquaredDiffs);
        }
        private static int MinIndex(double[] distances)
        {
            int indexOfMin = 0;
            double smallDist = distances[0];
            for (int k = 0; k < distances.Length; ++k)
            {
                if (distances[k] < smallDist)
                {
                    smallDist = distances[k];
                    indexOfMin = k;
                }
            }
            return indexOfMin;
        }
```

Crane Capacity, ATB Clearance, and Boom Clearance Checking

```
public void main_crane (double capacity, double gross_load, double
radius, double setradius, double boom_length, double p_dimension,
double dimension_A, double dimension_B, double module_elevation, double
ground_elevation, double rigging_height,  double module_height, double
two_block_distance, double module_width, double module_length, double
min_hook_block_clearance, ListViewItem module, string crane, DataRow
config)
        {
            OleDbCommand cmd = cnn.CreateCommand();
            cmd.CommandType = CommandType.Text;
            DataRow new_row;
            //check the capacity of the crane
            if (capacity > gross_load)
```

```
{
    //check the radius of the module
    if (radius > setradius)
    {
        try
        {
            //when setradius < radius we use setradius in
        the calculations
            //H will be used for anti-two block clearance
            H = Math.Sqrt(Math.Pow(boom_length, 2) +
            Math.Pow(p_dimension, 2)
            - Math.Pow((radius - dimension_B), 2));
            //H_prime is calculated based on the setradious
        rather than radious, and it will be used for boom
        clearance
            H_prime = Math.Sqrt(Math.Pow(boom_length, 2) +
        Math.Pow(p_dimension, 2)
        - Math.Pow((setradius - dimension_B), 2));
        }
        catch
        {
        MessageBox.Show("Check Your Database Please! Some
        information Is Missing", "Error",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
            cnn.Close();
            return;
        }
        //hook_block_clearanc is a variable that hold the
        distance from the tip of the boom
        //to the anti-two block system and if this amount
        is greater than 20ft or the user input
        // then the clearance is ok
        hook_block_clearance = H_prime + dimension_A -
        module_elevation + ground_elevation
            - rigging_height - module_height -
        two_block_distance;
        Point3D p1 = new Point3D(0, 0,
        dimension_A);//coordinates of the point that boom
        is attached to the suprestructure
        Point3D p2 = new Point3D(setradius - dimension_B,
        0, H_prime + dimension_A);//coordinates of the tip
        of the boom
        Point3D p3 = new Point3D(setradius - dimension_B -
        module_width / 2, module_length / 2,
            H_prime + dimension_A - two_block_distance -
        min_hook_block_clearance - rigging_height);
        Point3D p4 = new Point3D(setradius - dimension_B -
        module_width / 2, -(module_length / 2),
        H_prime + dimension_A - two_block_distance -
        min_hook_block_clearance - rigging_height);
        boom_clearance = minimum_distance_line_to_line(p1, p2, p3,
            p4).Item1;
     //check the clearance between the tip of the boom and anti-two
            block system
     if (hook_block_clearance >= min_hook_block_clearance)
```

```
                {
                    //check the clearance between the boom and the modoule
                    if (boom_clearance >= min_boom_clearance)
                    {
                        new_row = dataset.Tables["capacity_checked"].NewRow();
                        new_row["Module ID"] = module_ID;
                        new_row["Object Name"] = module.Text;
                        new_row["crane model number"] = crane;
                        new_row["crane configuration"] = config["index"];
                        new_row["Fail/Pass"] = "Pass".ToString();
                        new_row["gross Load"] = gross_load;
                        new_row["crane capacity"] = capacity;
                        new_row["Set Radius"] = setradius.ToString("N2");
                        new_row["Max set point"] = radius;
                        new_row["Hook block Clearance"] =
                        hook_block_clearance;
                        new_row["Boom Clearance"] = boom_clearance;
                        dataset.Tables["capacity_checked"].Rows.Add(new_row);
                        cmd.CommandText = "insert into capacity_checked values
                        ('" + module_ID + "', '" + module.Text + "','" + crane
                        + "','" +
                            config["index"] + "','" + "Pass" + "','" +
                        gross_load + "','" + config["capacity (lbs)"] + "','" +
                            setradius + "','" + radius + "','" +
                        hook_block_clearance + "' , '" + boom_clearance + "')";
                        cmd.ExecuteNonQuery();
                    }
else
                        {
                        new_row =
                        dataset.Tables["capacity_checked"].NewRow();
                        new_row["Module ID"] = module_ID;
                        new_row["Object Name"] = module.Text;
                        new_row["crane model number"] = crane;
                        new_row["crane configuration"]=config["index"];
                        new_row["Fail/Pass"]="LowBoomClearance".ToStrin
                        g();
                        new_row["gross Load"] = gross_load;
                        new_row["crane capacity"] = capacity;
                        new_row["Set Radius"] =
                        setradius.ToString("N2");
                        new_row["Max set point"] = radius;
                        new_row["Hook block Clearance"] =
                        hook_block_clearance;
                        new_row["Boom Clearance"] = boom_clearance;

                        dataset.Tables["capacity_checked"].Rows.Add(new
                        _row);
                        cmd.CommandText = "insert into capacity_checked
                        values ('" + module_ID + "','" + module.Text +
                        "','" + crane + "','" +config["index"] + "','"
                        + "Low Boom Clearance" + "','" + gross_load +
                        "','" + config["capacity (lbs)"] + "','" +
```

```
                                setradius + "','" + radius + "','" +
                                hook_block_clearance + "' , '" +
                                boom_clearance + "')";
                                cmd.ExecuteNonQuery();
                                    }
                                }
                                else
                                {
                                    new_row =
                                    dataset.Tables["capacity_checked"].NewRo
                                    w();
                                    new_row["Module ID"] = module_ID;
                                    new_row["Object Name"] = module.Text;
                                    new_row["crane model number"] = crane;
                                    new_row["crane configuration"] =
                                    config["index"];
                                    new_row["Fail/Pass"] = "Low Vertical
                                    Clearance".ToString();
                                    new_row["gross Load"] = gross_load;
                                    new_row["crane capacity"] = capacity;
                                    new_row["Set Radius"] =
                                    setradius.ToString("N2");
                                    new_row["Max set point"] = radius;
                                    new_row["Hook block Clearance"] =
                                    hook_block_clearance;
                                    new_row["Boom Clearance"] =
                                    boom_clearance;
                                    dataset.Tables["capacity_checked"].Rows.
                                    Add(new_row);
                                   cmd.CommandText = "insert into
                                    capacity_checked values ('" + module_ID +
                                    "','" + module.Text + "','" + crane +
                                    "','" +  config["index"] + "','" + "Low
                                    Vertical Clearance" + "','" + gross_load
                                    + "','" + config["capacity (lbs)"] +
                                    "','" +setradius + "','" + radius + "','"
                                    + hook_block_clearance + "' , '" +
                                    boom_clearance + "')";
                                    cmd.ExecuteNonQuery(); }
                            }
                            else
                            {
//when setradius > radius we use radius in the calculations
//H will beused for anti-two block clearance
H = Math.Sqrt(Math.Pow(boom_length, 2) + Math.Pow(p_dimension, 2)
                             - Math.Pow((radius - dimension_B), 2));
//H_prime is calculated based on the setradious rather than radious,
and it will be used for boom clearance
H_prime = Math.Sqrt(Math.Pow(boom_length, 2) + Math.Pow(p_dimension,
2)- Math.Pow((setradius - dimension_B), 2));
//hook_block_clearanc is a variable that hold the distance from the tip
of the boom
//to the anti-two block system and if this amount is greater than 20ft
or the user input then the clearance is ok
```

```
hook_block_clearance = H + dimension_A - module_elevation +
ground_elevation - rigging_height - module_height - two_block_distance;

 Point3D p1 = new Point3D(0, 0, dimension_A);//coordinates of the point
that boom is attached to the suprestructure
Point3D p2 = new Point3D(radius - dimension_B, 0, H +
dimension_A);//coordinates of the tip of the boom
Point3D p3 = new Point3D(radius - dimension_B - module_width / 2,
module_length / 2,H + dimension_A - two_block_distance -
min_hook_block_clearance - rigging_height);
Point3D p4 = new Point3D(radius - dimension_B - module_width / 2, -
(module_length / 2), H + dimension_A - two_block_distance -
min_hook_block_clearance - rigging_height);
boom_clearance = minimum_distance_line_to_line(p1, p2, p3, p4).Item1;
//check the clrearance between the tip of the boom and anti-two block
system
                new_row = dataset.Tables["capacity_checked"].NewRow();
                new_row["Module ID"] = module_ID;
                new_row["Object Name"] = module.Text;
                new_row["crane model number"] = crane;
                new_row["crane configuration"] = config["index"];
                new_row["Fail/Pass"] = "Short Boom".ToString();
                new_row["gross Load"] = gross_load;
                new_row["crane capacity"] = capacity;
                new_row["Set Radius"] = setradius.ToString("N2");
                new_row["Max set point"] = radius;
                new_row["Hook block Clearance"] = hook_block_clearance;
                new_row["Boom Clearance"] = boom_clearance;
                dataset.Tables["capacity_checked"].Rows.Add(new_row);
                cmd.CommandText = "insert into capacity_checked values
                        ('" + module_ID + "','" + module.Text + "','" +
                        crane + "','" + config["index"] + "','" +
                        "Short Boom" + "','" + gross_load + "','" +
                        config["capacity (lbs)"] + "','" +
                        setradius + "','" + radius + "','" +
                        hook_block_clearance + "' , '" + boom_clearance
                        + "')";
                            cmd.ExecuteNonQuery();
                        }
                    }
                    else
                    {
            new_row = dataset.Tables["capacity_checked"].NewRow();
            new_row["Module ID"] = module_ID;
            new_row["Object Name"] = module.Text;
            new_row["crane model number"] = crane;
            new_row["crane configuration"] = config["index"];
            new_row["Fail/Pass"] = "Low Capacity".ToString();
            new_row["gross Load"] = gross_load;
            new_row["crane capacity"] = capacity;
            new_row["Set Radius"] = setradius.ToString("N2");
            new_row["Max set point"] = radius;
            new_row["Hook block Clearance"] = hook_block_clearance;
            new_row["Boom Clearance"] = boom_clearance;
            dataset.Tables["capacity_checked"].Rows.Add(new_row);
```

```
          cmd.CommandText = "insert into capacity_checked values ('" +
module_ID + "','" + module.Text + "','" + crane + "','" +
config["index"] + "','" + "Low capacity" + "','" + gross_load + "','" +
config["capacity (lbs)"] + "','" +setradius + "','" + radius + "','" +
hook_block_clearance + "' , '" + boom_clearance + "')";
                                    cmd.ExecuteNonQuery();
                    }
                }
public (double, Vector3D) minimum_distance_line_to_line(Point3D p1,
Point3D p2, Point3D p3, Point3D p4)
    {
      Vector3D u = new Vector3D(p2.X - p1.X, p2.Y - p1.Y, p2.Z - p1.Z);
      Vector3D v = new Vector3D(p4.X - p3.X, p4.Y - p3.Y, p4.Z - p3.Z);
      Vector3D w = new Vector3D(p1.X - p3.X, p1.Y - p3.Y, p1.Z - p3.Z);
      double a = Vector3D.DotProduct(u, u); // always >= 0
      double b = Vector3D.DotProduct(u, v);
      double c = Vector3D.DotProduct(v, v); // always >= 0
      double d = Vector3D.DotProduct(u, w);
      double e = Vector3D.DotProduct(v, w);
      double D = a * c - b * b; // always >= 0
      double sc, sN, sD = D;  // sc = sN / sD, default sD = D >= 0
      double tc, tN, tD = D;  // tc = tN / tD, default tD = D >= 0
      // compute the line parameters of the two closest points
      if (D == 0) // the lines are almost parallel
            {
                sN = 0.0; // force using point P0 on segment S1
                sD = 1.0; // to prevent possible division by 0.0 later
                tN = e;
                tD = c;
            }
            else
            {  // get the closest points on the infinite lines
                sN = (b * e - c * d);
                tN = (a * e - b * d);
                if (sN < 0.0)
                {  // sc < 0 => the s=0 edge is visible
                    sN = 0.0;
                    tN = e;
                    tD = c;
                }
                else if (sN > sD)
                {  // sc > 1  => the s=1 edge is visible
                    sN = sD;
                    tN = e + b;
                    tD = c;
                }
            }
            if (tN < 0.0)
            {           // tc < 0 => the t=0 edge is visible
                tN = 0.0;
                // recompute sc for this edge
                if (-d < 0.0)
                    sN = 0.0;
                else if (-d > a)
                    sN = sD;
```

```
            else
            {
                sN = -d;
                sD = a;
            }
        }
        else if (tN > tD)
        {       // tc > 1  => the t=1 edge is visible
            tN = tD;
            // recompute sc for this edge
            if ((-d + b) < 0.0)
                sN = 0;
            else if ((-d + b) > a)
                sN = sD;
            else
            {
                sN = (-d + b);
                sD = a;
            }
        }
        // finally do the division to get sc and tc
        sc = (sN / sD);
        tc = (tN / tD);
        // get the difference of the two closest points
        Vector3D dP = w + (sc * u) - (tc * v);  // =  S1(sc) -
        S2(tc)
        double length_dp = dP.Length;
        return (length_dp, dP);   // return the closest distance
    }
```

References

Abu Abas, O. (2008). Comparison between data clustering algorithms. *TechnologyThe International Arab Journal of Information*, *5*(3).

Al-Hussein, M., Alkass, S., & Moselhi, O. (2005). Optimization Algorithm for Selection and on Site Location of Mobile Cranes. *Journal of Construction Engineering and Management*, *131*(February), 168–175. https://doi.org/10.1061/~ASCE!0733-9364~2005!131:5~579!

Ali, M. S. A. D., Varghese, K., & Babu, N. R. (2005). Collision Free Path Planning of Cooperative Crane Manipulators Using Genetic Algorithm. *Journal of Computing in Civil Engineering*, *19*(2), 182–193. https://doi.org/10.1061/(ASCE)0887-3801(2005)19

Chang, Y. C., Hung, W. H., & Kang, S. C. (2012). A fast path planning method for single and dual crane erections. *Automation in Construction*, *22*, 468–480. https://doi.org/10.1016/j.autcon.2011.11.006

Estivill-Castro, V. (2002). Why so many clustering algorithms. *ACM SIGKDD Explorations Newsletter*, *4*(1), 65–75. https://doi.org/10.1145/568574.568575

Han, S. H., Hasan, S., Bouferguène, A., Al-Hussein, M., & Kosa, J. (2015). Utilization of 3D visualization of mobile crane operations for modular construction on-site assembly. *Journal of Management in Engineering*, *31*(5), 1–9. https://doi.org/10.1061/(ASCE)ME.1943-5479.0000317

Han, S. H., Lei, Z., Hermann, U., Bouferguene, A., & Al-Hussein, M. (2020). 4D-based Automation of Heavy Lift Planning in Industrial Construction Projects. *Canadian Journal of Civil Engineering*, 1–37. https://doi.org/10.1139/cjce-2019-0825

Hasan, S., Al-Hussein, M., Hermann, U. H., & Safouhi, H. (2010). Interactive and dynamic integrated module for mobile cranes supporting system design. *Journal of Construction Engineering and Management*, *136*(2), 179–186. https://doi.org/10.1061/(ASCE)CO.1943-7862.0000121

Hasan, S., Bouferguene, A., Al-Hussein, M., Gillis, P., & Telyas, A. (2013). Productivity and CO2 emission analysis for tower crane utilization on high-rise building projects. *Automation in Construction*, *31*, 255–264. https://doi.org/10.1016/j.autcon.2012.11.044

Hermann, U., Hendi, A., Olearczyk, J., & Al-Hussein, M. (2010). An integrated system to select, position, and simulate mobile cranes for complex industrial projects. *Construction Research Congress 2010: Innovation for Reshaping Construction Practice - Proceedings of the 2010 Construction Research Congress*, 267–276. https://doi.org/10.1061/41109(373)27

Huang, C., Wong, C. K., & Tam, C. M. (2011). Optimization of tower crane and material supply locations in a high-rise building site by mixed-integer linear programming. *Automation in Construction*, *20*(5), 571–580. https://doi.org/10.1016/j.autcon.2010.11.023

Jaillon, L. C. (2009). *The evolution of the use of prefabrication techniques in Hong Kong construction industry*. https://scholars.cityu.edu.hk/en/publications/the-evolution-of-the-use-of-prefabrication-techniques-in-hong-kong-construction-industry(d5773a2a-6f12-4a6a-9238-e8304a33506c).html

Kang, S. C., & Miranda, E. (2006). Planning and visualization for automated robotic crane erection processes in construction. *Automation in Construction*, *15*(4), 398–

414. https://doi.org/10.1016/j.autcon.2005.06.008

Keogh, E., Chakrabarti, K., Pazzani, M., & Mehrotra, S. (2001). Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. *Knowledge and Information Systems*, *3*(3), 263–286. https://doi.org/10.1007/PL00011669

Lei, Z. (2014). *Automated Simulation Model for Crane Motion Planning in Heavy Industrial Projects* (Vol. 1) [PhD Thesis University of Alberta]. https://doi.org/10.1017/CBO9781107415324.004

Lei, Z., Taghaddos, H., Han, S. H., Bouferguène, A., Al-Hussein, M., & Hermann, U. (2015). From autoCAD to 3ds max: An automated approach for animating heavy lifting studies. *Canadian Journal of Civil Engineering*, *42*(3), 190–198. https://doi.org/10.1139/cjce-2014-0313

Lei, Z., Taghaddos, H., Hermann, U., & Al-Hussein, M. (2013). A methodology for mobile crane lift path checking in heavy industrial projects. *Automation in Construction*, *31*, 41–53. https://doi.org/10.1016/j.autcon.2012.11.042

Mao, C., Shen, Q., Pan, W., & Ye, K. (2015). Major Barriers to Off-Site Construction: The Developer's Perspective in China. *Journal of Management in Engineering*, *31*(3), 04014043. https://doi.org/10.1061/(ASCE)ME.1943-5479.0000246

Olearczyk, J., Al-Hussein, M., Bouferguene, A., & Telyas, A. (2012). 3D-modeling for crane selection and logistics for modular construction on-site assembly. *Congress on Computing in Civil Engineering, Proceedings*, *2012*, 445–452. https://doi.org/10.1061/9780784412343.0056

Rodriguez-Ramos, W. E., & Francis, R. L. (1983). Single crane location optimization. *Journal of Construction Engineering and Management*, *109*(4), 387–397.

https://doi.org/10.1061/(ASCE)0733-9364(1983)109:4(387)

Safouhi, H., Mouattamid, M., Hermann, U., & Hendi, A. (2011). An algorithm for the calculation of feasible mobile crane position areas. *Automation in Construction*, *20*(4), 360–367. https://doi.org/10.1016/j.autcon.2010.11.006

Sawhney, A., & Mund, A. (2002). Adaptive Probabilistic Neural Network-based Crane Type Selection System. *Journal of Construction Engineering and Management*, *128*(3), 265–273. https://doi.org/10.1061/(ASCE)0733-9364(2002)128:3(265)

Shih-Chung, K., Chi, H.-L., & Miranda, E. (2009). Three-Dimensional Simulation and Visualization of Crane Assisted Construction Erection Processes. *Journal of Computing in Civil Engineering*, *3801*(March), 99–109. https://doi.org/10.1061/(ASCE)0887-3801(2009)23

Sydora, C., Lei, Z., Siu, M. F. F., Han, S. H., & Hermann, U. (2020). Critical lifting simulation of heavy industrial construction in gaming environment. *Facilities*, *39*(1), 113–131. https://doi.org/10.1108/F-08-2019-0088

Taghaddos, H., Hermann, U., & Abbasi, A. B. (2018). Automated Crane Planning and Optimization for modular construction. *Automation in Construction*, *95*(July), 219–232. https://doi.org/10.1016/j.autcon.2018.07.009

Wu, D., Lin, Y., Wang, X., Wang, X., & Gao, S. (2011). Algorithm of crane selection for heavy lifts. *Journal of Computing in Civil Engineering*, *25*(1), 57–65. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000065

Chapter 3: Neural Network Method for Mobile Crane Prediction in Industrial Projects (Paper 2)[1]

Abstract

The majority of the lift engineers use manual methods to select a mobile crane for their construction projects through lift charts provided by crane manufacturers. This process can be time-consuming and complicated. Artificial neural networks as powerful supervised machine learning techniques can be utilized to process large amounts of historical datasets and predict results on unseen datasets with high accuracy. In this research, the authors aim at developing a neural network to predict the most suitable mobile cranes for heavy industrial construction projects. The neural network model takes as input the features of the modules that need to be lifted and predicts the most suitable mobile crane configuration for the project. Various techniques such as K-fold cross-validation and sensitivity analysis were implemented to increase the accuracy and shorten the model's runtime. For training and validating the model, the historical data of four actual industrial projects was used, and the results of the cross-validation test set showed

[1] Azami, R, Lei, Z, Hermann, R, Zubick, T (2021). "Prediction of Mobile Crane Configurations for Heavy Industrial Construction Projects Using Artificial Neural Network" To be submitted to the Canadian Journal of Civil Engineering.

70% accuracy. This model can be trained for any other type of cranes and achieve the required results by changing the training and test sets.

Keywords

Crane selection, heavy construction, mobile crane, neural network, machine learning

3.1 Introduction

Off-site construction has been prevalent in the past few decades, and it is not new to the construction industry. Efficiency, predictability, safety, and sustainability are among the benefits that off-site construction has brought to the construction industry (O'Connor et al. 2014). Pipe racks and vessels are examples of prefabricated offsite modules in factories, assembled in module yards, and shipped to the site for installation. The industry adopted this process to achieve high efficiency and reduce waste by eliminating on-site work (Lei 2014). Cranes are widely used in the off-site construction industry to improve construction efficiency and facilitate the on-site assembly of prefabricated components (Hasan 2013). Many cranes have been designed and produced by crane manufacturers as a result of the increase and diversification of the lifting needs of the construction industry. Crane selection for construction projects is a crucial and complex task. The construction operations' time, cost, and safety can be significantly influenced by selecting the appropriate crane (Mund 1999).

The United States construction industry has higher injury and fatality rate than other sectors due to its complicated and constantly changing nature (Rivara and Alexander 2007). In the United States, the construction industry accounts for 19.4% of workplace fatalities and 12.3% of occupational injuries and illnesses, even though construction workers represent only 4.8% of the U.S. workforce (Abudayyeh and Al-

Battaineh 2003). Improper crane selection, lack of proper crane support system, or failure to calculate support reactions have been the root causes of a significant portion of crane accidents (Hasan 2013). It has been shown by Neitzel et al. (2001) that among numerous factors involved in construction accidents, cranes contribute to as many as 33% of total deaths and injuries experienced in the industry. Mobile cranes have resulted in over 84% of fatalities related to crane use (Beavers et al. 2006).

During the assembly process of prefabricated modules, heavy mobile cranes are used as the main lifting equipment. Because of the competitiveness of the construction industry, contractors need to analyze the capacity and capability of critical resources in order to reduce the cost and shorten the duration of construction (Hasan 2013). A typical industrial site where mobile cranes are used to lift the completed modules is shown in Figure 13. The rental cost and crew fees of mobile cranes make them expensive to use. For example, for industrial projects in Alberta, Canada, a dual-crane lift of a single vessel (Demag CC1800 and Manitowoc 4100) costs CAD $320,000, including the rental cost, mobilization/demobilization, and ground preparation (Lei 2014). Therefore, cranes should be utilized efficiently to ensure that the projects are successful and productive and avoid budget overruns, schedule delays, and safety issues (Hermann et al. 2011 and Lei, 2014).

Figure 13: Typical module in industrial projects (photo courtesy of PCL Industrial Management)

In practice, the lifting schedules of cranes are managed according to the demand, urgency, and priority of the tasks. Nevertheless, the capacities and locations of cranes can significantly restrict their coverage of service. Consequently, crane planning has a significant impact on construction sequencing, scheduling, budgeting, and safety. Therefore, a proper and feasible crane arrangement is a crucial prerequisite for the success of a construction project. In the past decades, the selection of cranes and their location on the job site is often made manually and largely dependent on the design of the building. This method, manual planning through trial and error based on the site layout, is inefficient and error-prone in today's construction projects, where projects become more and more complex (Chen 2016; Hasan 2013).

Practitioners in the crane rental and construction industries have cooperated in developing structured methods and software tools for crane selection due to the central role of cranes in construction operations (Mund 1999). Even though lift planning and crane selection have received considerable attention from practitioners in recent decades, the developed approaches to select the best possible mobile crane do not necessarily result in optimum crane selection. Thus, new techniques need to be created, or the current systems should be improved (Hasan 2013).

To select the most suitable mobile cranes, this research developed an artificial neural network model which predicted the ideal crane by considering the most relevant features of the modules and the projects. The model has been trained and tested with historical data from real industrial projects. This paper is structured as follows: section 3.2 to 3.4 background, context, and hypothesis; section 3.5 neural network literature review; section 3.6 methodology, case study, and validation; section 3.7 conclusion and future work.

3.2 Background

Construction sequence, scheduling, budgeting, and safety are notably impacted by crane planning, a challenging task due to the complex trade-offs between the involved parameters (e.g., crane size, lifting sequence, path planning). Previously published studies related to crane planning are mainly focused on the following areas: (1) crane selection; (2) optimum crane location; (3) crane path planning; (4) the usage and coordination of multiple cranes, and (5) simulation and visualization of crane operation (Chen 2016).

As part of crane planning, crane selection consists of two main phases: crane type selection and crane model selection. The former is concerned with the type of the crane

(e.g., lattice boom, hydraulic boom, fixed, mobile), and the latter is for determining the exact model of the crane (Mund 1999). The parameters that affect the crane type selection often result from the environmental, organizational, market, and industry conditions rather than decision-making process (Shapira and Schexnayder 1999). Research shows that the primary factors affecting crane type selection can be categorized as follows: (1) type of use; (2) duration on the site; (3) construction height; (4) site spaciousness; (5) terrain topography; (6) soil stability; (7) construction aspect ratio; (8) crane relocation on the site; and (9) site accessibility. The second phase involves mainly selecting a crane that can fulfill the physical requirements of the lifts to be carried. Therefore, the factors that affect the model selection are summarized as: (1) building height; (2) load dimension; (3) the maximum load to be lifted; (4) the maximum radius at which a load has to be placed; and (5) the depth of placement (Sawhney and Mund 2002).

In this regard, a fuzzy logic approach was proposed by Hanna (1999) to select the type of crane. In this method, the expert's vague knowledge of the suitability of crane types in various project conditions is translated into fuzzy sets and fuzzy rules. A fuzzy inference engine can quantitatively identify the best crane type for a project based on the description of the project's condition that an expert provided in words (Hanna and Lotfallah 2004). This system is not capable of performing the crane model selection for a specific task.

In 1993 an application was developed, called PRECISE, that minimized the number of moves required by a mobile crane to erect steel structures. Optimum path and the erection sequence were selected for the crane; however, it is limited to only single-story steel structures and the use of mobile cranes on site (Raynar and Smith 1993).

CRANES is a rule-based expert system for selecting tower cranes to construct multi-story buildings, which requires the user to input the building dimension, the load distributions, and the crane location. This system's limitations are that the crane's location needs to be chosen in advance, and the system is limited to tower cranes only (Cooper 1987). The module CRANE proposed by Gary (1985) can consider the implications of the building's shape, the load distribution, and the possible crane's location. This system requires changing the number and the size of the selected crane(s) during the project construction, which is not practical and acceptable to the contractors. It is based on mathematical calculation rather than knowledge (Gray and Little 1985).

LOCRANE was developed by Warszawski (1990) as a test case for applying the expert system methodology to construction planning tasks. It was limited to the selection of cranes for a given building. This knowledge base system cannot combine substitute transportation methods (e.g., equipment pumps, hoists, carts) with cranes for optimization and capturing interrelationships between crane selection and other construction tasks. CRANE ADVISER has been developed by Al-Hussein (1995) to integrate a knowledgebase and algorithm programs to assist in the crane selection for high-rise building projects. Expert knowledge, heuristics, and rules of thumb related to crane selection are contained in a knowledge-based module.

An expert system, entitled NEXPERT, was combined with geographical information system by Varghese (1992) to optimize the route selection when large objects are moving from the pick to set location to optimize the route selection. Wind speed, rental charges, lift radii, weights, and dimensions of the heaviest load have been considered in SELECTCRANE. This system was developed by Hanna (1994) to

recommend the type of crane to the user. IntelliCRANES, which can assist with both crane type and model, was developed by Sawhney and Mund (2001). The system mainly includes two modules: (1) a neural network-based crane type selection module and (2) a knowledge-based expert system module for model selection. This system has some limitations, such as the number of cranes that can be selected as output is limited to only eight, and simultaneous crane selection for more than one crane for a construction site is not possible (Mund 1999).

Another system developed by Sohn et al. (2014) for tower crane selection considers all the costs used in the economic analysis. It converts them into the net present value for an accurate comparison. The minimum cost solution, the lateral support structure, the foundation design, and the individual components required are the final output of this system. HELPS2 is a system that can assist in overall lift planning from the preliminary planning stage to the detailed planning stage, final evaluation, and selection. This system was developed by Hornady et al. (1992), and the outputs increase in detail as the lifting plan evolves. HELPS2 optimizes the lifting plan according to the objectives of cost, reliability, safety, and performance. However, it still depends on the user to choose step by step (Hornaday et al. 1993).

Moselhi et al. (2004) developed a 3D modeling crane selection system that searches in its database for all technically feasible cranes according to the module's dimension, weight, and location. The cranes retrieved from the databases can satisfy the specified clearance between the crane, the lift, and the adjacent buildings (Moselhi, Alkass, and Al-Hussein 2004). A 3D computer-aided application was developed by Wu et al. (2011) to select mobile cranes for heavy lifts. This application can consider the

lifting capacity, clearance between the boom or jib to equipment, and ground bearing pressure. Although this system can find all the feasible cranes, it does not specify which one is the optimum solution, and the user needs to select the crane manually.

Artificial Neural Network has been recognized as a powerful and effective tool in construction industry by many researchers in the past few years. This method has the ability to predict the results in a shorter time with higher accuracy AL-Zwainy and Aidan (2017). For example, neural network has been used to make the lifting capacity of virtual mobile cranes as realistic as possible. Since the data from load charts of the mobile cranes were limited, an artificial neural network was trained to predict the lifting capacity based on the real-time states of the boom length, the load radius, and the counterweight of the virtual mobile crane. The results showed relatively small deviation between the neural network predicted values and the ground truth values in the load charts, which did not provide significant impacts on the lifting capacity for most of the states (Roysson et al. 2021).

Collecting pavement distress information by using a UAV with high-resolution camera and feed them into neural network models to predict pavement conditions was the main objective of a research by Zhu et al. (2022). They have used the acquired data to train three neural networks, namely Faster R-CNN, YOLOv3, and YOLOv4, and compared their performances. The results indicate that YOLOv3 had the best performance with 56.6% accuracy on test sets. The proposed model in this research can be used for non-destructive automatic pavement conditions inspection. Moreover, Cheng and Wang (2018) trained a convolutional neural network using 3,000 images collected from closed-circuit television to identify both the type and location of pipe defects in

sewer systems. The model proposed in this research achieved mean average precision of 83%. Machine learning techniques can even be used for determining residual values of heavy construction equipment. Three different models including Modified Decision Tree (MDT), LightGBM, and XGBoost were trained and tested by Shehadeh et al. (2021). The results of each model were evaluated based on Prediction Accuracy, Mean Absolute Error (MAE), Mean Squared Error (MSE), Mean Absolute Percentage Error (MAPE), and Coefficient of Determination (R^2) metrics. It was indicated in this research that MDT had the highest prediction accuracy of 92.84%. This approach can be an invaluable tool for equipment sellers, buyers, and owners.

A construction cost estimation model was developed which takes into account the economic variables and indexes. Historical data from 372 low- and midrise buildings were used to verify the results of the model. Cost estimation errors of the previously proposed models such as Back Propagation Neural Networks (BPNNs) and Support Vector Machines (SVMs) are much higher than the proposed model in this research (Rafiei and Adeli 2018). Recently, several machine learning models such as k-nearest neighbors (KNN), random forest (RF), artificial neural networks (ANNs), and support vector machine (SVM) have been developed to predict deterioration of tunnels. The results indicate that deterioration of tunnels is significantly impacted by18 variables, among which width has the greatest impact on the prediction of deterioration of tunnels. RF model was identified as the best model with 85.38% accuracy compare to other models (Ahmed et al. 2021). In addition, association rule mining in combination with Bayesian network was implemented to determine defects occurrence probabilities and the relationships between the defects. The hybrid machine learning model implemented in

this research can evaluate the risk of defects, and a model called Swiss cheese model (SCM) was utilized to determine the priority order of the defects (Fan 2020).

A unique approach was implemented by Moon and Munira Chowdhury (2021) for training a neural network to predict the 28-day concrete strength. They used 3-day concrete strength as the prior information in the neural network training instead of using random weights and bias to reduce overfitting and improve the 28-day concrete strength prediction. The results of the prior information based neural network (PI-NN) showed significant improvements compared to the conventional methods. Tijanić et al. (2020) have utilized artificial neural networks to estimate road construction costs at conceptual phase of the project life cycle where limited information is available. Three different models, namely multilayer perceptron, general regression neural network, and radial basis function neural network, were implemented, and the results demonstrated 87% accuracy in the cost estimation of the studied project by using the general regression neural network. In another research, Ilbeigi et al. (2020) utilized artificial neural network to reduce the energy consumption of the buildings in Iran through finding the most effective parameters of the building on energy consumption. Furthermore, genetic algorithm was implemented to optimize the energy consumption of the building. The results showed 35% decrease in energy consumption through using this approach.

As labor costs make up 30 to 50 percent of the today's construction project costs, construction labor productivity is usually considered as the total project productivity. Goodarzizad et al. (2021) proposed to measure the construction labor productivity. First the influential factors were extracted from the literature and then by implementing an artificial neural network with three layers and six processing nodes in the hidden layer the

construction labor productivity was estimated. The results of this study show that labor experience and skill is the most crucial factor among all the factors with 25% influence on the results.

A study carried out by Leśniak and Juszczyk (2018) proposed an artificial neural network to estimate the overhead costs of the construction projects due to the fact that the current methods are either time-consuming or inaccurate. Multilayer perceptron neural networks were developed and trained using 143 cases of completed construction projects. The proposed method can process the overhead construction costs at early stage of the construction investment process with satisfactory precision. More recently, Wang et al. (2022) proposed an automated vision-based method for productivity analysis of cable crane transportation to eliminate the weakness of the traditional manual method which is time-consuming and tedious. In this approach a Faster region-based convolutional neural network (Faster R-CNN) was implemented to improve the feature extraction capability of the proposed model. The mean average precision of the proposed model was 98.01%. Convolutional neural network was implemented in another research by Chenet al. (2021) to identify soft-story buildings at the city scale. In this approach, features of the buildings were extracted automatically from (3D) point could data through a moving box, and then deep neural network was used to identify soft-story buildings.

In summary, the common techniques that have been used for crane selection can fall into two categories: (1) 3D simulation and visualization and (2) statistic and machine learning. All the methods or the systems that have been developed have some limitations regarding the input/output data or the criteria that were considered. Thus, the author aimed to improve the mobile crane model selection approach by simultaneously

considering module dimensions, weights, and safety in this research. Researchers have found several facts that neural networks are more effective than the traditional estimation methods. Neural networks can be implemented to predict future results even in the absence of data and information (AL-Zwainy and Aidan 2017). Artificial neural networks can significantly reduce the processing time of the information and provide preliminary estimates and, consequently, the cost of data processing reduces significantly (Naik and Radhika 2015). Having considered all the advantages of using neural networks mentioned in the literature, flexibility, and tendency to predict and classify all types of data better than any other method, this approach has been selected to develop a mobile crane classifier. Moreover, two different approaches (i.e., a heuristic and an artificial neural network) have been combined to improve processing time and accuracy. Despite all the advantages of this methodology, it has some limitations regarding the dataset required for training and testing the model and the machine required for running the model, which may not be available for all the projects. Consequently, training a model with a small or medium size dataset results in predictions with low accuracy rate. In addition, increasing the number of hidden layers to increase the accuracy could impact the processing time of the model significantly (Figueroa et al., 2012; Rácz et al., 2021). Therefore, this approach may not be the best option when enough historical data and machines with high computational power are not available.

3.3 The Research Context

The proposed neural network for automation of mobile crane selection in heavy industrial projects is part of a more extensive study that aims to automate crane planning for industrial projects. It primarily consists of four stages: lift planning (feasibility and

sequencing), crane type and model selection (the main contribution of this paper), crane positioning, and crane path planning.

The crane model selection involves a neural network trained and tested by historical data to validate the results. The training dataset could be raw data from past projects or the results from the previously developed algorithm by the authors (Azami et al. 2021) (see chapter 2).

3.4 Crane Model Selection

The selection of a mobile crane's model for industrial projects is not a straightforward process. Many projects exist in which both tower (fixed) and mobile cranes constitute practical solutions. There are, however, construction projects where site and job conditions mandate a specific model of crane to be used (Shapira and Glascock 1996). Further, the factors affecting the selection of the crane model, such as lifting capacity, working radius, monthly rate, mobilization/demobilization cost, and safety, often do not have a linear relationship, and considering all these factors is a complex process. Therefore, a heuristic algorithm has been developed to select the mobile crane for each project. At the same time, it considers the essential factors simultaneously (the details of this algorithm can be found in the previous paper).

Although the heuristic approach considers all the key factors, the processing time is the weakness of this method due to the non-linearity of the relationships and the great number of cranes and modules in the database. Consequently, the results of the heuristic approach, which were the most suitable mobile cranes for the previous industrial projects, were used to train an Artificial Neural Network (ANN). The advantage of the ANN method is that it is trained and tested once, and it can be used for future projects for

predicting the results; in other words, it does not need to be trained for each project, so the processing time has shortened significantly. The ANN approach and the implementation are explained in detail in the methodology sections. The methodology section is followed by a case study, in which the employment and performance of the model are illustrated. At the end, a brief conclusion and discussion for future works are presented.

3.5 Neural Network Background

Machine learning systems perform based on a simplified model of the biological neurons are called Neural Network (Hykin 1999). In the past few decades, the popularity and usefulness of neural networks for classification, clustering, pattern recognition, and prediction in many disciplines have increased (Abiodun et al. 2018; Aljarah et al. 2018; Almonacid et al. 2017). In medical applications various forms of neural networks have been utilized by researchers to diagnose diseases and classify biomedical information such as heart diseases and diabetes (Esteva et al. 2017). Capabilities of neural networks to to process a large amount of data in a short period of time are the primary reasons of the success of these models (Faris et al. 2016).The internal parameters of the neural networks can change by themselves in the same way as the biological neural network changes itself to perform some cognitive tasks (such as recognizing faces). Human brains are full of neurons, and neurons are cells in the brain. A single neuron in human brain shown in Figure 14 consists of different parts. The neuron has a number of input wires which are called the Dendrites. The dendrites are responsible to receive inputs from other locations. Also, a neuron has an output wire called the Axon which sends signals to the other

neurons. The Nucleus is located between the dendrites and the axon which is the processing section of the neuron and responsible for analysing data.

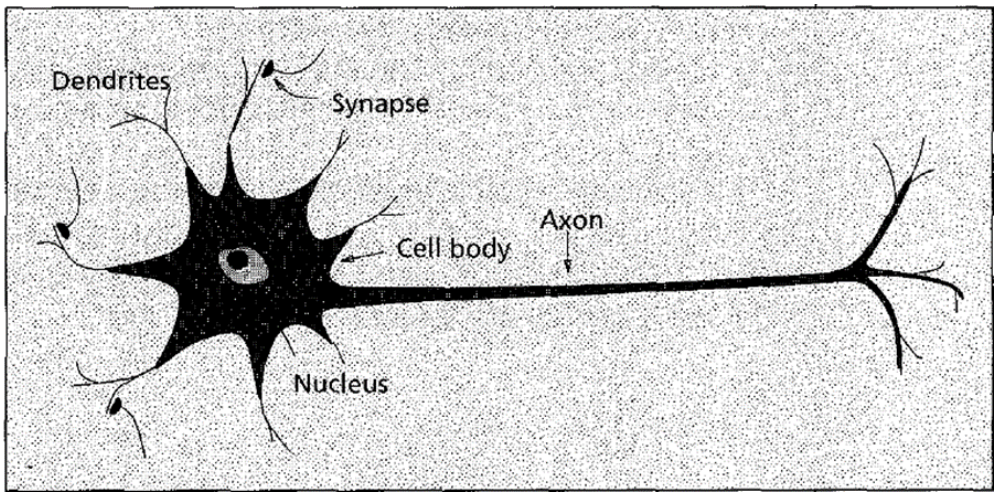


Figure 14: Biological neuron (reproduced from Jain et al., 1996)

The topology of a neural network is predefined and composed of a bunch of neurons. Depending on the task that the network has to learn, some possible topologies are kept constant. However, for some applications such as robotics, the topology itself can change dynamically. Therefore, the topology can be considered as a parameter. The type and intensity of the information exchanged are determined by the weights associated with the connection among neurons (Angelini et al. 2008).

Jain et al. (1996) stated that characteristics such as nonlinearity, high parallelism, robustness, fault and failure tolerance, learning, ability to handle imprecise and fuzzy information, and generalization made ANNs an attractive approach for problem-solving. Possessing such characteristics makes artificial networks desirable as (1) nonlinearity allows a better fit to the data; (2) noise-insensitivity provides accurate prediction in the presence of uncertain data and measurement errors; (3) fast processing and hardware

failure-tolerance achieved by high parallelism; (4) the system updates its internal structure by learning and adaptivity; (5) unlearning data is made possible by generalization. The development of a mathematical algorithm will enable ANNs to learn by mimicking information processing and knowledge acquisition in human error (Mohammadhassani et al. 2013).

3.5.1 The Neural Network Model

ANNs are constructed from three types of layers, and each layer is composed of simply interconnected elements called nodes or processing elements that act as microprocessors Figure 15. The nodes on the input layer, shown on the left side of Figure 15, corresponding to the biological original dendrites. The connections on the output side (see right side of Figure 15) corresponding to the axon and transmit the output. Providing connection weights and activation functions or thresholds within the nodes, the synopsis is mimicked. The activation of the nodes results from the sum of the weighted inputs and can be negative, zero, or positive. This is due to the synaptic weights, which represent excitatory synapses when positive or inhibitory ones when negative (Mund 1999). The yellow and blue circles in the hidden layers represent the activation nodes and are usually noted as θ. Each node is connected to every node from the next layer and each connection (arrows) has a particular weight. Each node's impact on node in the next layer can be seen as the weight.

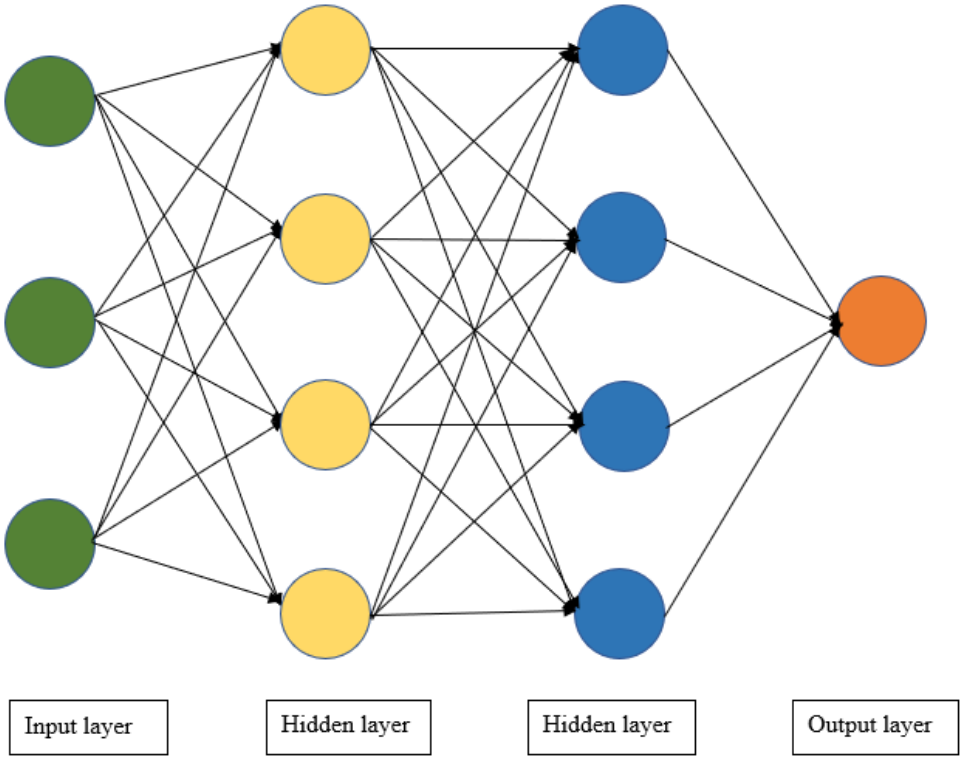


Figure 15: Neural network structure

To demonstrate how the weights and activation functions work, we consider the top yellow node, which has been connected to all the previous nodes (green nodes). The values of all the previous nodes (green) are multiplied by their associated weights and summed, resulted in the value of the top yellow node. The yellow node has a predefined activation function that defines if the node is "activated" or how "active" it will be, based on the summed value shown in Figure 16. The additional node with value one is called the "bias" node. A bias value allows the activation function to shift to the left or right, and it helps to get a better fit for the data.

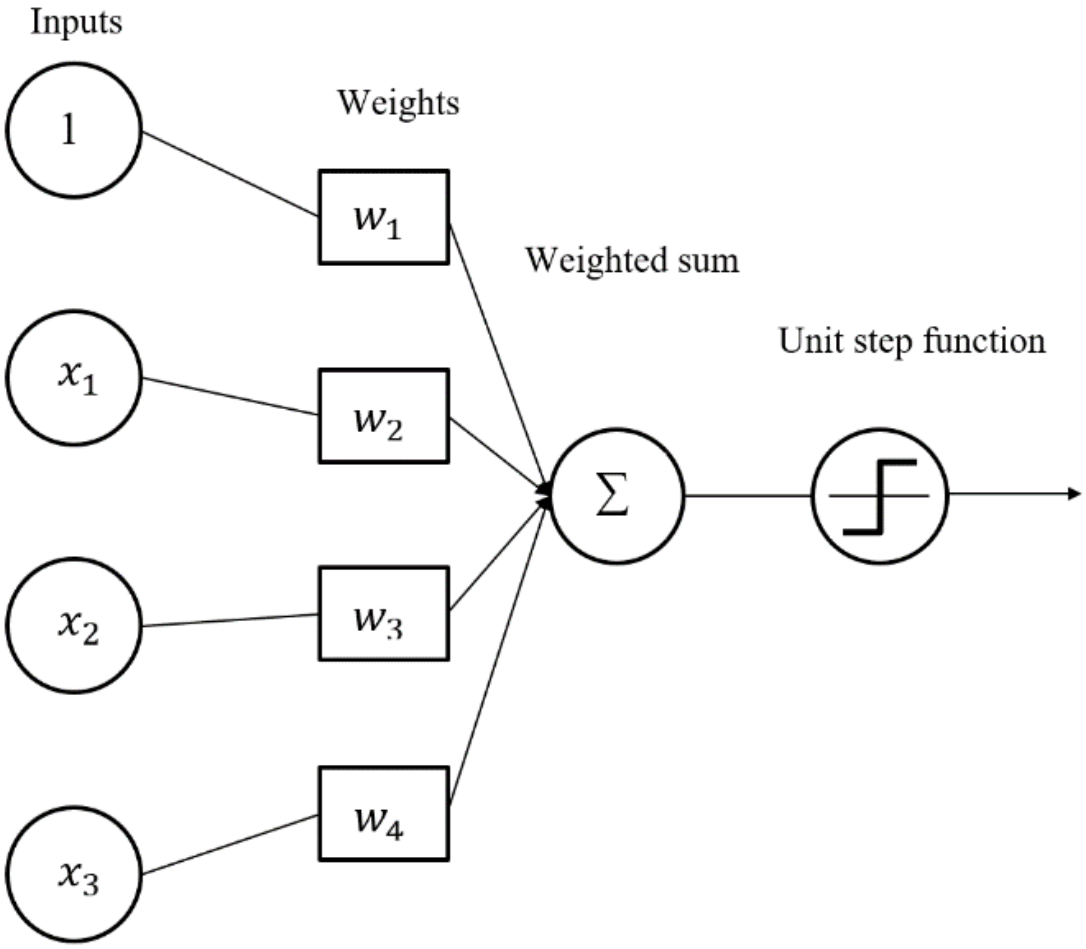


Figure 16: Neural network node activation

3.5.2 Activation Function

In a neural network, the activation function defines whether a given node should be activated or not based on the weighted sum. There are various activation functions such as binary step function, linear function, and sigmoid, where each one has some advantages and disadvantages. Therefore, the user should select the most suitable one for the NN model. Unless an activation function was implemented in a neural network, the output of the NN model would be a simple linear function which is just a polynomial of degree one. The activation function differentiates between neural network and linear

regression as it helps the neural network learn and recognize complex mapping from data (Sharma et al. 2020). In this research, Rectified Linear Activation (ReLU) function has been selected.

The binary step function works based on a threshold that the user defines, and if the weighted sum is greater than the threshold, the node will be activated or otherwise.

if $(z > \text{threshold}) \rightarrow$ active the node (value 1)

if $(z < \text{threshold}) \rightarrow$ do not "active" the node (value 0)

Sigmoid function Eq. 24, the most widely used function, is non-linear and continuously differentiable with a smooth S-shaped function Figure 17. Sigmoid function transforms the values in the range 0 to 1. It is defined as follow:

$$\text{Sigmoid}(x) = \frac{1}{1+\exp(-x)} \tag{24}$$

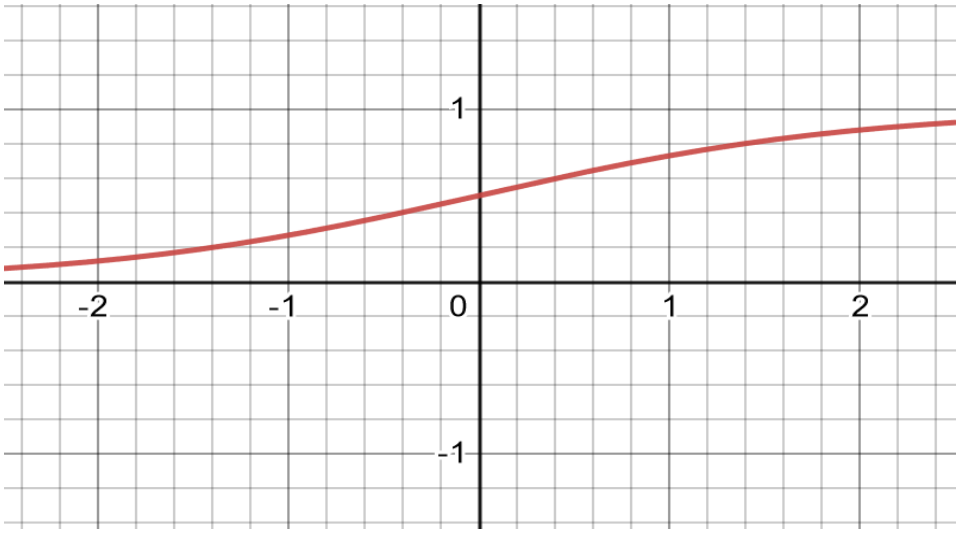


Figure 17: Sigmoid fnction

As it is not in the scope of this research, for more details about other types of activation functions, readers are encouraged to read this paper (Sharma et al. 2020).

3.5.3 Types of Neural Networks

A neural network can be of several types; this section briefly reviews the most common types.

3.5.3.1 Multilayer Perceptron Neural Network (MLP)

MLPs employ one or several hidden layers in the ANNs as opposed to single-layer perceptron (SLP) which only has one input and one output layer, and SLPs are considered as the simplest form of ANNs (Faris et al. 2016). Due the limited number of layers, SLPs are not able to handle nonlinear data efficiently (Ojha et al. 2017). Therefore, MLPs are the most commonly used class of neural networks due to its fast operation, ease of implementation, and smaller training requirements (Kocyigit et al. 2008; Mund 1999; Subasi 2007; Übeyli 2009). An example of MLP consisting of input, hidden, and output layers were shown in Figure 15. The hidden layer processes and transmits the input data to the output layer. MLPs are feedforward networks trained with backpropagation. There is no analytical method to determine the number of hidden layers and the neurons in MLP. The more complex the relationship between the input data is, the greater the number of hidden layers and neurons required. Although Ward Systems Group (1996) proposed to use one hidden layer with the number of processing elements equal to the result of Eq. 25, the only way to find the best combination of the hidden layers and neurons is trial and error (Azadi and Karimi-Jashni 2016; Azadi and Sepaskhah 2012; Hazarika et al. 1997; Oğulata et al.2009; Puig-Arnavat et al. 2013; Subasi and Erçelebi 2005).

$$PE = \frac{(x+y)}{2} + \sqrt{z} \tag{25}$$

Where:

x = the number of input processing elements

y = the number of output processing elements

z = the number of training examples.

Random weights define the connections between the layers, and then using Eq. 26 the value for each processing node is determined.

$$Z_j = \sum_{i=1}^{m} \omega_{ij} x_i + b_j \tag{26}$$

Where:

m = the total number of inputs

x_i = denotes the input variable i

b_j = the bias term

ω_{ij} = the connection weight.

Following the calculation of Z_j, an activation function such as sigmoid, which was defined in Eq. 1, can be employed in MLP. Therefore, the output of neuron j can be obtained by Eq. 27:

$$\hat{y} = \text{Sigmoid}(Z_j) = \frac{1}{1 + \exp\left(\sum_{i=1}^{m} \omega_{ij} x_i + b_j\right)} \tag{27}$$

Having set the number of the hidden layers, the processing elements, and concluding the output of all the neurons, the backpropagation minimizes the prediction error. The error is determined by running all the training cases through the network and comparing the output with the ground truth. Different error functions can be implemented to combine the differences, where the most common error function is Sum Squared Error.

In this study, MLP has been implemented with various hidden layers from one to five and 60 to 5,000 nodes for each layer (Aljarah et al. 2018; Heidari et al. 2019).

3.5.3.2 Radial Basis Function Neural Network (RBF)

RBFs are feedforward networks developed in 1988 by Lowe. RBFs are supervised algorithms that usually contain one hidden layer, and it is useful for non-linear networks. Due to their simple architecture compared to other models, they are faster in training and providing better results (Montazer and Giveki 2015). The training time in this model is considerably low as it uses only one hidden layer, which can be considered an advantage. Moreover, the Gaussian activation function is employed in RBFs rather than standard sigmoid which is recommended for MLP. As opposed to MLP, when the input data set includes many features, RBFs are not the best option to use since they are sensitive to the curse of dimensionality (Mund 1999). Although this model can be trained fast, random selection of the centers and widths of the RBF units makes this model less efficient (Montazer and Giveki 2015).

3.5.3.3 Probabilistic Neural Network (PNN)

PNNs are multilayer feedforward neural networks built using Parzen's approach to devise a family of probability density function estimators (Parzen 1962). PNNS can solve complex scientific and engineering problems such as classification of labeled stationery data patterns, unsupervised algorithms that work with unlabeled data sets, signal processing applications, dealing with waveforms as data patterns, image classification and recognition, earthquake magnetiude prediction (Mohebali et al. 2020). Figure 18 displays the architecture of a PNN that recognizes K = 2 classes while it can be

extended to any number of K classes. Each hidden node in Class k corresponds to a Gaussian function centered on its associated feature vector in the kth class.

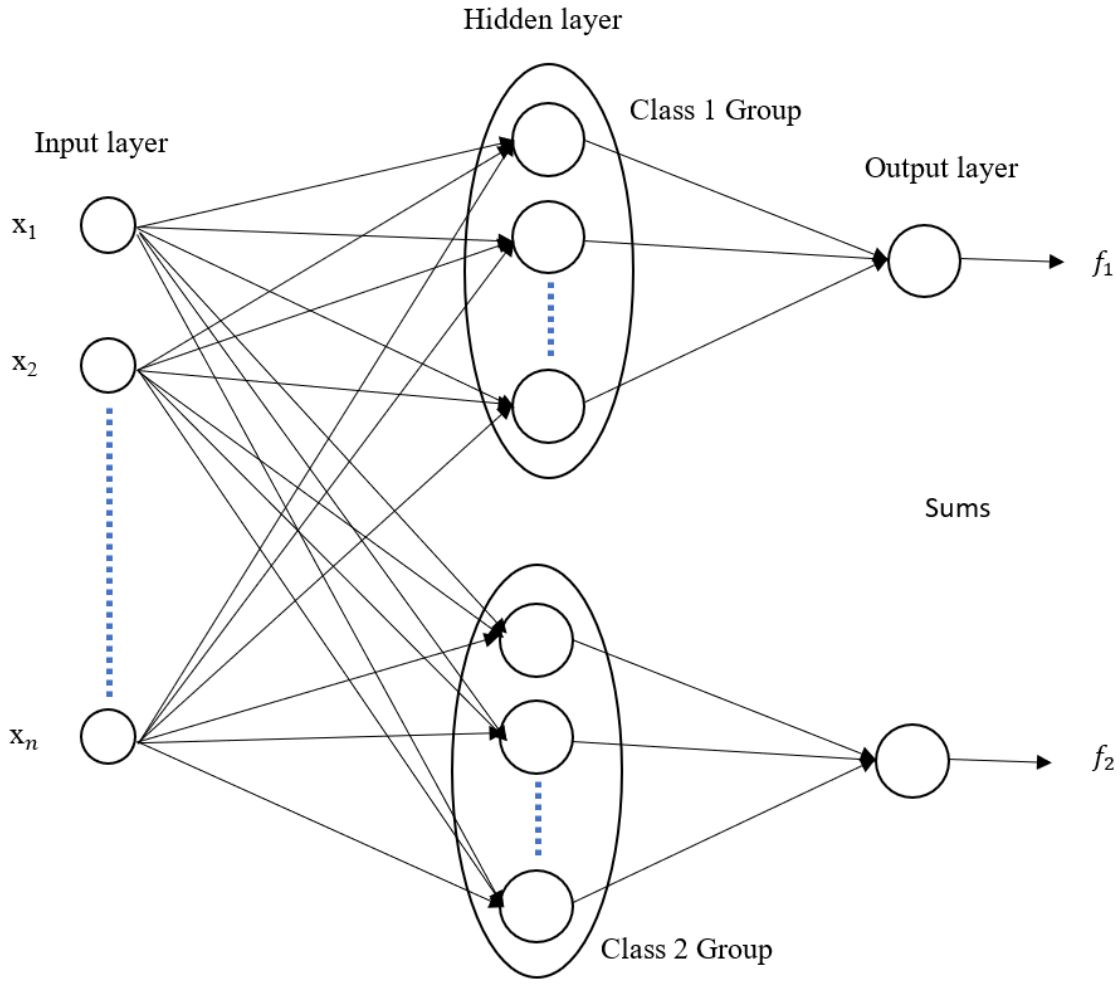


Figure 18: Probabilistic neural network

3.5.3.4 Generalized Regression Neural Networks (GRNN)

GRNNs, which generally proposed by (Specht 1991), are feedforward ANNs GRNNs work the same way PNNs work; however, GRNNs produce continuous values rather than classification. Additionally, GRNNs can train quickly on sparse data sets, although they are slow in execution on large datasets (Mund 1999). GRNNs have been

proved to be a competent tool for predicting engineering problems (Bowden et al. 2006; Firat and Gungor 2009; Ozturk and Turan 2012).

3.5.4 Training Neural Networks

The procedure of modifying the synaptic weights of the ANN in an orderly fashion to attain the desired design objective is called training (Haykin 2019). Training an ANN is a complex and essential task in supervised machine learning models. In most applications, this involves optimizing the objective function while maintaining the networks' ability to generalize; in other words, correctly identifying unseen inputs is the desired design objective (Wing Lun Chiu 2002). The goal is achievable by exposing the ANN to numerous examples or cases. Training and testing datasets must be collected beforehand and contain data for input and output variables.

Moreover, training the network with invalid data will result in incorrect predictions. Subsequently, the user cannot expect optimum results from such ANN. Therefore, in this research, training data have been acquired using the authors' heuristic approach to ensure that the near-optimum data has been used to train the ANN model. Many learning approaches have been developed and proposed, such as the Genetic Algorithm, Hebbian Rule, and Widrow-Hoff rule. For Multilayer Perceptron, a descendant of the Widrow-Hoff rule called the Backpropagation Algorithm is the best-known learning rule. In backpropagation, the network typically starts with random weightings on its connections/synapses. The neural network is repeatedly exposed to a set of training data. The known output of the training data presented is used to adjust the neural network's connection weights incrementally. This process is achieved by

calculating the gradient vector of the error surface, a multidimensional mapping of the neural network's overall squared error during the training process—the gradient vector points in the direction of the steepest descent from the current point. A minor adjustment in that direction would decrease the error, and many such adjustments would eventually achieve the goal of finding a minimum in the error surface (Mund 1999).

Although large adjustments may converge on a solution faster than small adjustments, they may also overstep the solution. In practice, the adjustment size is proportional to the slope/gradient of the error surface and the learning rate. A constant that defines the size of the adjustment to the connection weights of the neural network is called the learning rate. The correct setting for the learning rate is application-dependent and is usually chosen by experiment.

A major problem with this approach is that it does not actually train the neural network to minimize the error it will make when presented with new cases but trains it to minimize the error on the training set (Bishop 1995). The most significant manifestation of this distinction is the problem of over-learning or over-fitting. A network with more connection weights can model a more complex function but is prone to over-fitting. A network with fewer connection weights would perform a better generalization but may not be sufficiently powerful to model the underlying function (Networks and Trajan Software Ltd Co. Durham 1999).

The description of the backpropagation algorithm is as follows (Mund 1999):

1. Set the connection weights randomly.
2. The input x and output y for a given iteration j are presented as vectors (Eq. 28 and Eq. 29):

$$x_j=[x_0,x_1,x_2,\ldots,x_{n-1}] \quad (28)$$

$$y_j=[y_0,y_1,y_2,\ldots,y_{m-1}] \quad (29)$$

Where:

n = number of input elements

m = number of output elements

3. For each layer L, the network output y_{nl} is calculated and passed to the next layer (Eq. 30):

$$y_{nl}=f\left[\sum_{i=0}^{i=n-1} w_i x_i + b_i\right] \quad (30)$$

4. Working from back to front, the weight is adjusted starting at the output layer (Eq. 31):

$$w_{ij}(n+1) = w_{ij}(n) + \eta\delta_{nj}(n) \quad (31)$$

Where:

$w_{ij}(n)$ = connection weight from nod i to node j at iteration n

η = the gain term/learning rate

δ_{nj} = the error propagated from node j in iteration n

3.5.5 Testing Neural Networks

Testing the network evaluates its ability to extract a feasible solution when the inputs are unknown and not trained to the network. In other words, there are different methods for selecting the test dataset, such as Train Test Split, K fold, and Leave One Out. Train Test Split works based on a random selection where some portions of the dataset are selected randomly for training and the rest for a test. The ratio of the test set

can be between 10%-20% of the dataset. Train Test Split is easy to implement due to simplicity even when the dataset is small and the network may not be trained and tested based on the whole dataset.

On the other hand, the K-fold technique is designed to ensure that every observation from the original dataset has the chance of appearing in the training and test set. The dataset is divided into K separate portions, and during each iteration one section is held for testing, and the rest is considered for training. The value of K, which is user input, could be any amount between two and the length of the dataset. Figure 19 and Figure 20 show a schematic drawing of dividing the dataset and a detailed step-by-step algorithm, respectively, to better illustrate how the K-fold technique works. Leave One Out is a special case of the K fold where the value of K is always equal to the length of the dataset. Thus, the learning algorithm is applied once for each instance, using all other instances as a training set and using the selected instance as a single-item test set.

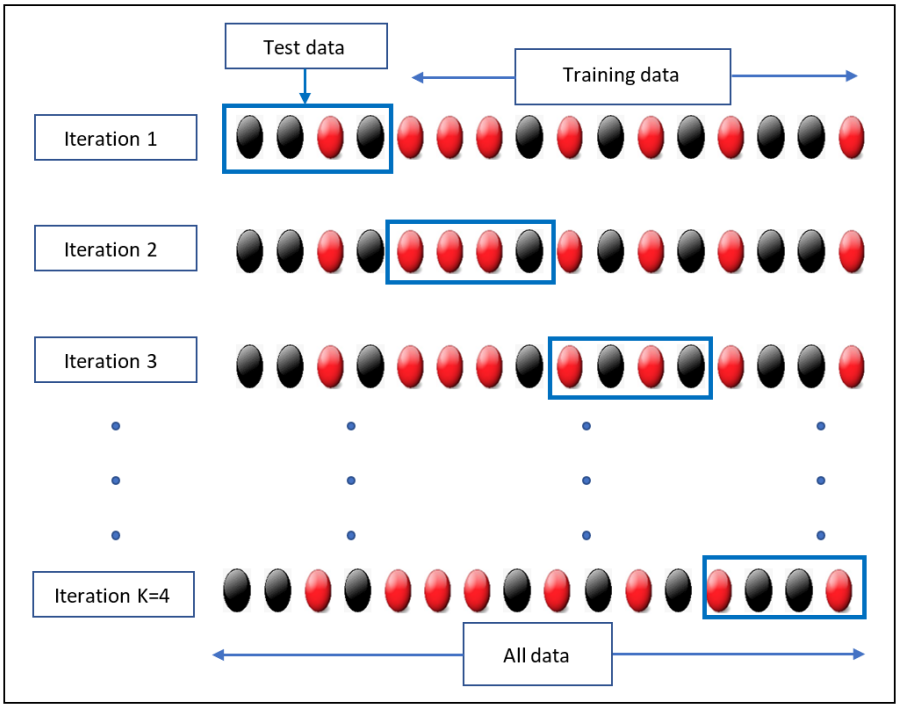


Figure 19: K-Fold iteration

In this research, K fold has been selected for testing the neural network with K equals five. In every iteration of the learning process, 20% of the instances in the dataset are selected for testing and 80% for training.

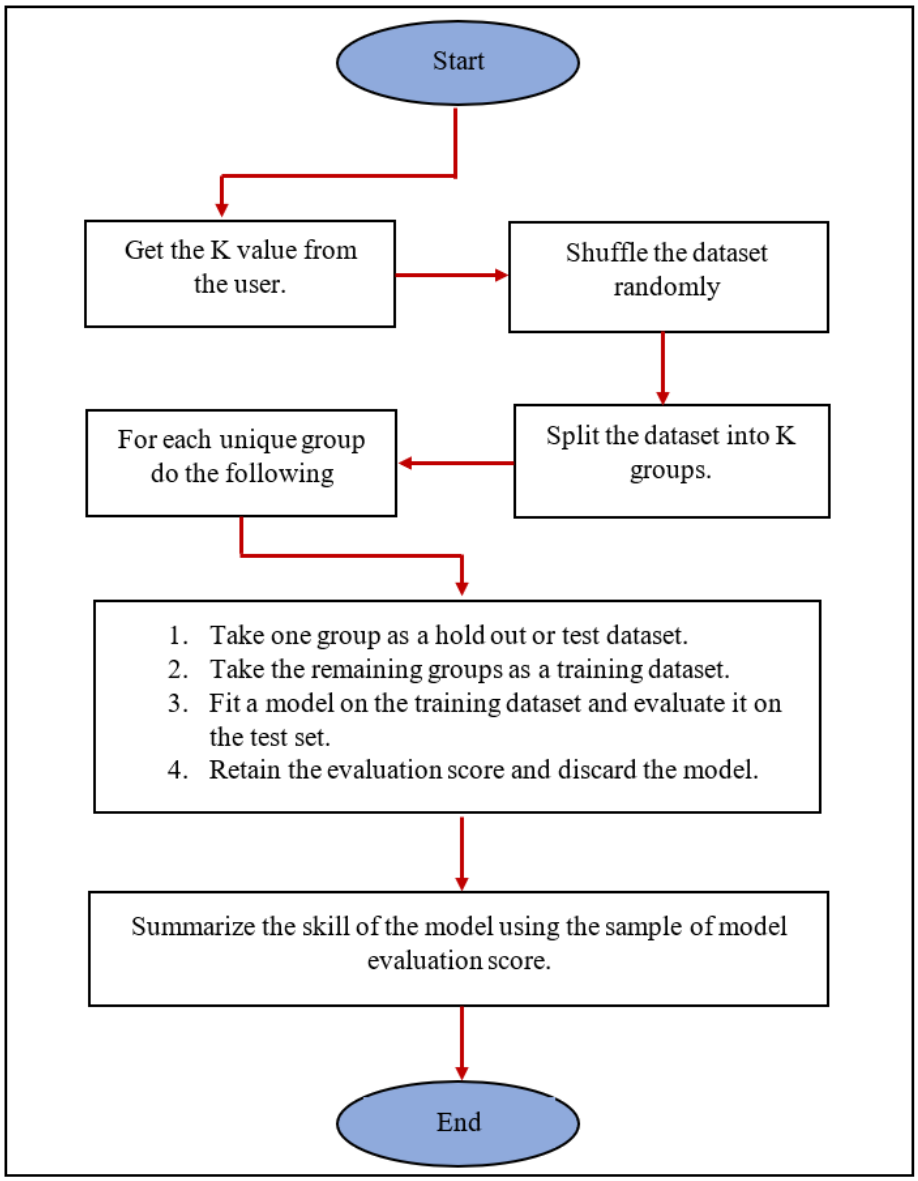


Figure 20: K-Fold algorithm

3.6 Methodology

As mentioned before, Multilayer Perceptron Neural Network has been selected in this research due to its high accuracy and low runtime compare to other models for multiclass clasification. MPLs are feedforward networks trained with backpropagation, which is able to separate data into a specified number of output categories. The input

layer of the MLP will contain ten processing elements, which equals to the number of the features of the modules such as weight, length, width, and height. Likewise, the number of processing elements in the output layer will be equal to the number of the crane configurations in the dataset, which in this case were 58 processing elements. One hidden layer with 60 processing elements was selected for the first trial based on the Ward Systems Group (1996) advice, and then the model was modified by increasing the number of hidden layers and processing elements.

Microsoft Visual Studio 2019, Microsoft Access and Excel 2016, and Python 3.8 were utilized to create a user interface, connect a database to the model, and develop the neural network model. The model, which was developed using Python 3.8, reads the input data from an Excel file that includes the training data Figure 21. Following the training step, the model needs to be tested to verify that the model can generalize everything that has been learned. The complete code written in Python and the training set can be found in the appendix II.

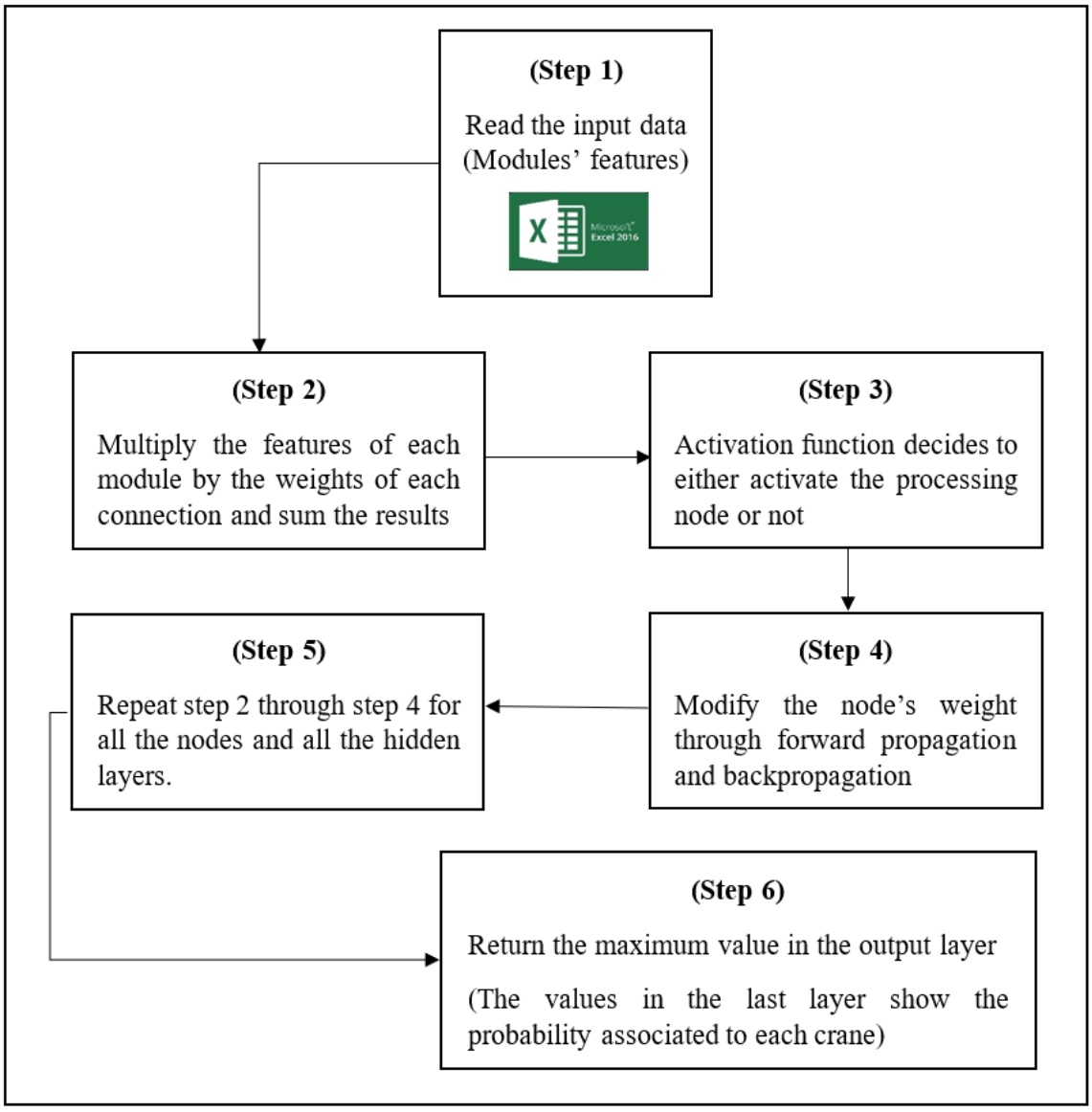


Figure 21: Neural network methodology

3.6.1 Testing and validation

A case study including a heavy industrial project was used to illustrate how the crane configurator works. Crane configuration selection was performed for four projects and validated with the historical data of PCL Industrial Management Inc. The results and details of the case study are described in the following paragraphs.

The total number of modules needed to be lifted was 630, weighing between 12 klb and 530 klb, and the momentum range falls between 1 klbf.ft and 40 klbf.ft. Table 3 shows five records that are included in the dataset.

The columns ‘label’ in Table 3 and Table 4 show the cranes that have been selected for lifting modules with the features presented in the feature columns. The values in Table 3 are the raw data, and each column has a different characteristic, which is called a feature in data science. Because real-world data generally contains noise, missing values, and maybe unusable format, which cannot be directly used for machine learning models, preprocessing helps enhance the quality of the data to promote the extraction of meaningful insights from the data.

Data preprocessing includes four main stages: data quality assessment, data cleaning, data transformation, and data reduction. In this research, after careful consideration based on the quality of the available dataset, data normalization was the only preprocessing that was needed to be done before feeding the model with the training dataset. Normalization helps scale the data within a range to avoid building incorrect machine learning models and executing data analysis. Data normalization can be done through different techniques, which are explained in the following paragraphs.

1. Minimum-Maximum: this method uses the minimum and maximum values in the dataset. The normalized values are calculated as follows (Eq. 32):

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{32}$$

Where x = the value of the feature in the dataset, x_{min} = the minimum record in that feature, and x_{max} = the maximum record in that feature.

This method is a suitable choice when both of the following conditions are met:

(a) The approximate upper and lower bounds of the data with few or no outliers are known.

(b) The data are approximately uniformly distributed across a range.

2. Maximum: this method uses the greatest value in the dataset (Eq. 33).

$$x' = \frac{x}{x_{max}} \tag{33}$$

3. Square root: this method uses the sum of all values in the dataset (Eq. 34).

$$x' = \frac{x}{\sqrt{\sum_{i=1}^{n} x_i}} \tag{34}$$

4. Logarithmic: logarithmic scaling computes the log of the values to compress a wide range to a narrow range (Eq. 35).

$$x' = \log x \tag{35}$$

5. Standard score: this method is a variation of Minimum-Maximum that represents the number of standard deviations away from the mean. In this method, normalized attributes are less affected by the outliers; therefore,a standard score has been implemented to normalize the dataset so that the value in each feature follows the standard deviation(Eq. 36 to Eq. 38) (Ahn et al. 2020).

$$x' = \frac{x-u}{s} \tag{36}$$

$$u = \frac{\sum_{i=1}^{N} x_i}{N} \tag{37}$$

$$s = \frac{\sqrt{\sum_{i=1}^{N} (x_i - u)^2}}{N} \tag{38}$$

Where:

x' = the normalized feature

u = the average of values

x_i = record in the dataset

N = the total number of the records in the dataset

s = the standard deviation of the features.

Table 4 shows the normalized values from Table 3. For instance, the following calculation demonstrates data normalization for the first record in column "Weight" in Table 3:

u (Average of column "Weight.")= $\frac{141{,}044+43{,}308+94{,}380+75{,}458}{4} = 88{,}547.50$

s (Standard deviation of column "Weight.") $= \frac{\sqrt{\sum_{n=1}^{N} (x_i - u)^2}}{N} = 40{,}856.86$

$x' = \frac{141{,}044 - 88{,}547.5}{40{,}856.86} = 1.28$

Table 3: Examples of the records in the dataset

Weight (klb)	Height (ft)	Length (ft)	Gross load (klb)	Moment (klbf. ft)	Set Radius (ft)	Label
141.04	12	110	266.19	14,096.16	100	LR1400
43.30	38	18	103.03	2,166.12	50	CK-2500
94.38	17	118	222.81	9,916.73	105	CC-2400
75.45	17	109	222.81	9,916.73	105	M-18000

Table 4: Examples of normalized data

Weight	Height	Length	Gross load	Moment	Set Radius	Label
1.28	-0.86	0.51	1.02	1.17	0.43	LR1400
-1.10	1.70	-1.72	-1.65	-1.59	-1.72	CK-2500
0.14	-0.41	0.71	0.31	0.20	0.64	CC-2400
-0.32	-0.41	0.49	0.31	0.20	0.64	M-18000

3.6.2 Validation Metrics

To measure the performance of a model, various metrics are available such as precision and recall, defined as (Eq. 39 and Eq. 40):

$$\text{Precision} = \left(\frac{\text{TP}}{\text{TP+FP}}\right) \tag{39}$$

$$\text{Recall} = \left(\frac{\text{TP}}{\text{TP+FN}}\right) \tag{40}$$

Where: TP (true positives) and FP (false positives) are the numbers of objects correctly and incorrectly predicted, respectively. Likewise, FN (false negative) is the number of objects incorrectly predicted.

For instance, TP is a correctly recognized crane model for lifting modules in this research, while FN is an incorrectly recognized no crane model in the dataset for lifting the modules. After testing, each output was examined and compared with the ground truth. However, precision and recall are more useful for binary classification rather than

multiclass classification. Thus, in this research, recall and precision have not been implemented.

The cost function plays a key role in adjusting a neural network's weights to create a better fitting machine learning model. Specifically, during forward propagation, the neural network runs on a training dataset and generates outputs. These results and the target labels are compared, and the loss function calculats a penalty for any deviation between the target label and the neural network's outputs. During backpropagation, the partial derivative of the loss function is calculated for each trainable weight of the neural network, and the partial derivatives adjust the weights. Backpropagation iteratively adjusts the trainable weights of a neural network to produce a model with lower loss (Ho and Wookey 2020).

The loss function evaluates "How badly the algorithm predicts the results." The loss function would output a higher number if the predictions were completely wrong. In contrast, if the model predicted correctly, it would output a lower number. The loss function helps to understand the learning rate of the model. It includes various approaches for different models, namely, regression loss function, binary classification loss function, and multi-class classification loss function (Ho and Wookey 2020).

Due to the fact that Cross-Entropy loss function maximize the likelihood of the correct prediction given the ground truth in the training set, it has been implemented in this research (Li et al. 2020). Cross-Entropy loss function is a subcategory of the multi-class classification loss function and a generalization of binary cross-entropy loss. The number of bits required to transmit a randomly selected event from a probability distribution is called Entropy. Cross-entropy builds upon the idea of entropy from

information theory and calculates the number of bits required to represent or transmit an average event from one distribution compared to another distribution. In this approach, the model's performance, whose output is a probability value between 0 and 1, is measured. As long as the predicted probability converges to the actual label, the cross-entropy decreases. The Standard binary cross-entropy loss function is given by Eq. 41:

$$J = -\frac{1}{M}\sum_{m=1}^{M}\left[y_m \times \log\left(h_\theta(x_m)\right) + \left(1 - y_m\right) \times \log\left(1 - h_\theta(x_m)\right)\right] \quad (41)$$

Where:

M = the number of training examples

y_m = the target label for training example m

x_m = the input for training example m

h_θ = the model with neural network weights θ.

The first term, $y_m \times \log\left(h_\theta(x_m)\right)$, disincentivizes probabilistic false negatives during training. For example, if a training example has target one and the output of the machine learning model was 0.4, therefore, we would say that there is a probabilistic false negative of 60%. In other words, from a Bayesian perspective, the model has 60% confidence in the wrong result, or from a Frequentist perspective, the model would be wrong 60% of the time. The loss function penalizes this 60% by returning the value -log (0.4) = 0.39. In the perfect case, if the binary classifier outputs 1, then it is completely accurate for the training example, and the loss is -log (1) = 0. The same logic applies to the second term and probabilistic false positives (Ho and Wookey 2020).

Accuracy is another metric for evaluating classification models. It is defined as follows (Eq. 42):

$$\text{Accuracy} = \frac{\text{Number of correct prediction}}{\text{Total number of prediction}} \tag{42}$$

For binary classification, accuracy can also be calculated in terms of positive and negative as follows (Eq. 43):

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{43}$$

3.6.3 Research Findings

In this case study, Multilayer Perceptron Neural Network (MLP) was implemented. The first step was to define the network parameters such as the number of the elements of the input layers, the number of the hidden layer and its elements, and the number of the output layer's elements. The number of the input elements must be equal to the number of the features of the modules in the dataset, which in this case, there were ten different features. The number of the hidden layer and its elements were selected as 1 and 60, respectively.

Since the value of K for the K-fold has been set to five in this research, the dataset was divided into five portions. One portion consists of 20% of the dataset was selected randomly for the test, and four other portions consist of 80% of the remaining for training the model. Therefore, the following plots in Figure 22 show the model's accuracy and loss for each fold of the dataset during the training. The number of iterations needs to be found by experiment, and it is a trade-off between the running time of the algorithm and the accuracy of the predicted results. Although the accuracy of the model may increase

due to increasing the number of iterations, high variance or overfitting could happen, which could prevent the model from being able to generalize.

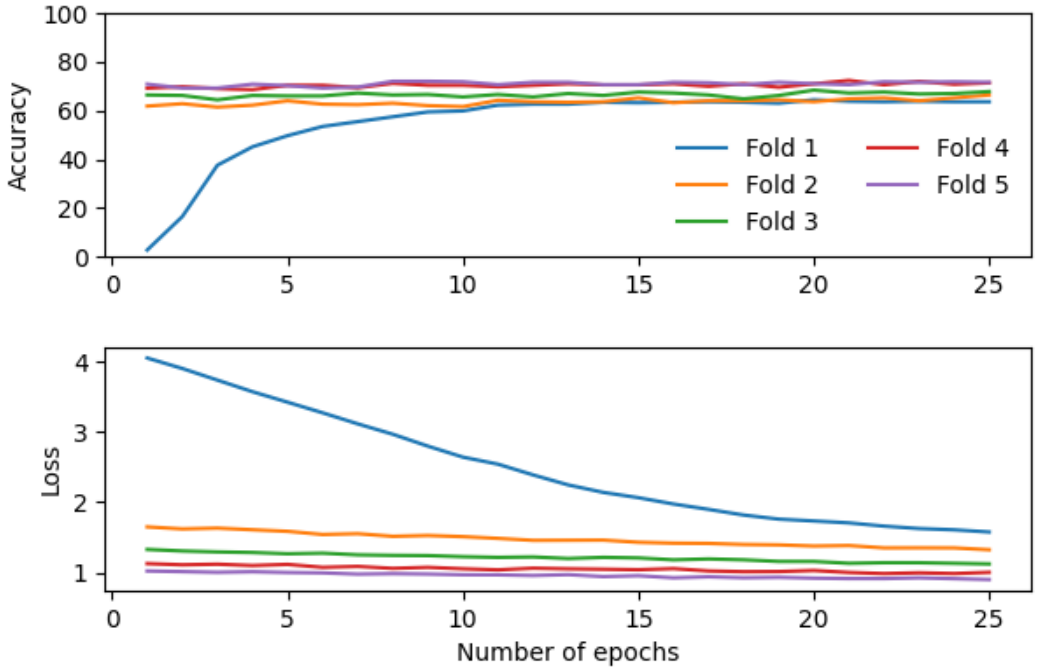


Figure 22: Accuracy and loss of the model with one hidden layer and 60 nodes on the training set

Having trained the model using the training dataset, the model must be tested by the test set so that the accuracy and generalization of the model on unseen data can be evaluated. Table 5 shows the test results of the model trained in Figure 22. Likewise, the training and testing results of a model with two hidden layers and 60 processing nodes in each hidden layer are shown in Figure 23 and Table 6.

Table 5: Accuracy of the model including one hidden layer and 60 nodes tested with the available test set

Fold	Accuracy of the test set (%)
1	61.90
2	72.20
3	68.25
4	62.98
5	68.00
Average accuracy	66.61

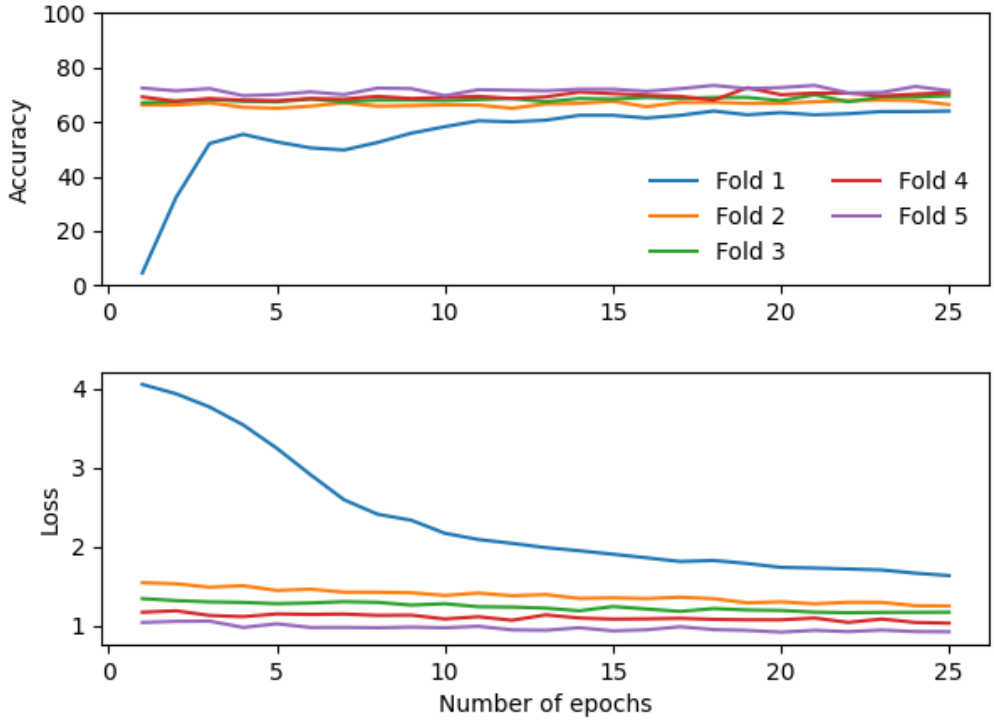


Figure 23: Accuracy and loss of the model including two hidden layers and 60 nodes in each layer on the training set

Table 6: Accuracy of the model including two hidden layers and 60 nodes tested with the available test set

Fold	Accuracy of the test set (%)
1	58.73
2	65.87
3	68.25
4	70.63
5	72.00
Average accuracy	67.10

The same dataset was used to train and test other MLP models with one and two hidden layers and 600 and 1200 processing elements to better understand the model's performance and find an optimum model. The evaluations of these models are presented in Figures 24 to 27 and Tables 7 to 10, respectively. Evaluating these plots and tables show that by increasing the number of hidden layers and the processing nodes, the accuracy of the model on the training set and test set increases. However, the rate of increasing the accuracy, which increases either by increasing the hidden layer or the processing nodes, decreases gradually. Therefore, the optimum model needs to be found by trial and error.

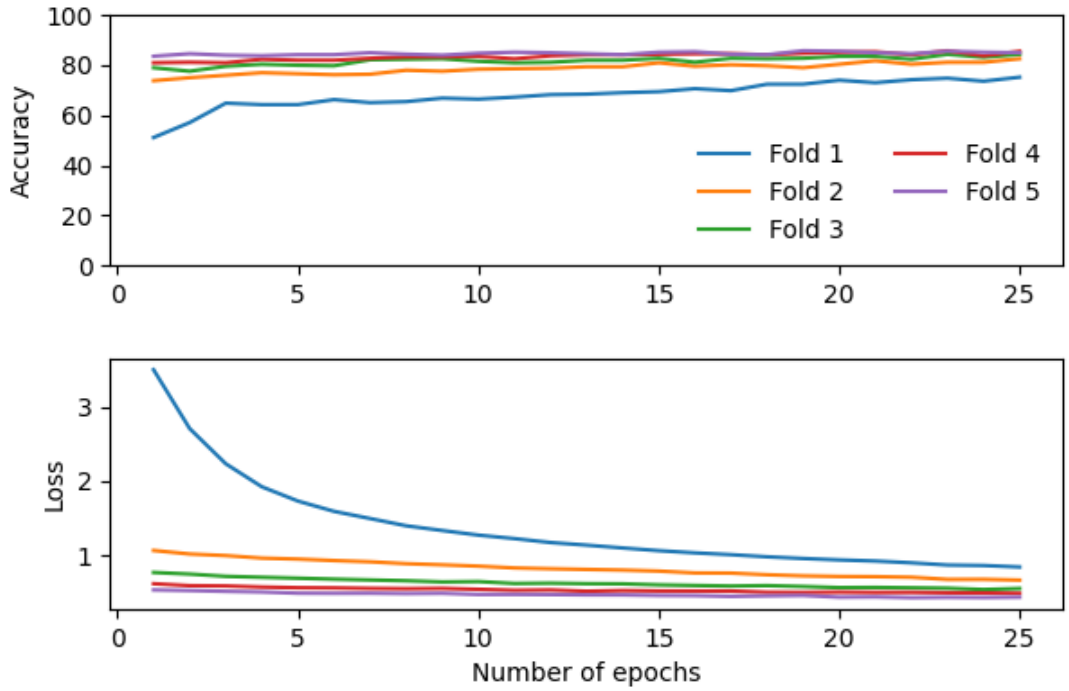


Figure 24: Accuracy and loss of the model including one hidden layer and 600 nodes on the training set

Table 7: Accuracy of the model including one hidden layer and 600 nodes tested with the available test set

Fold	Accuracy of the test set (%)
1	59.52
2	66.66
3	77.77
4	71.43
5	80.00
Average accuracy	71.10

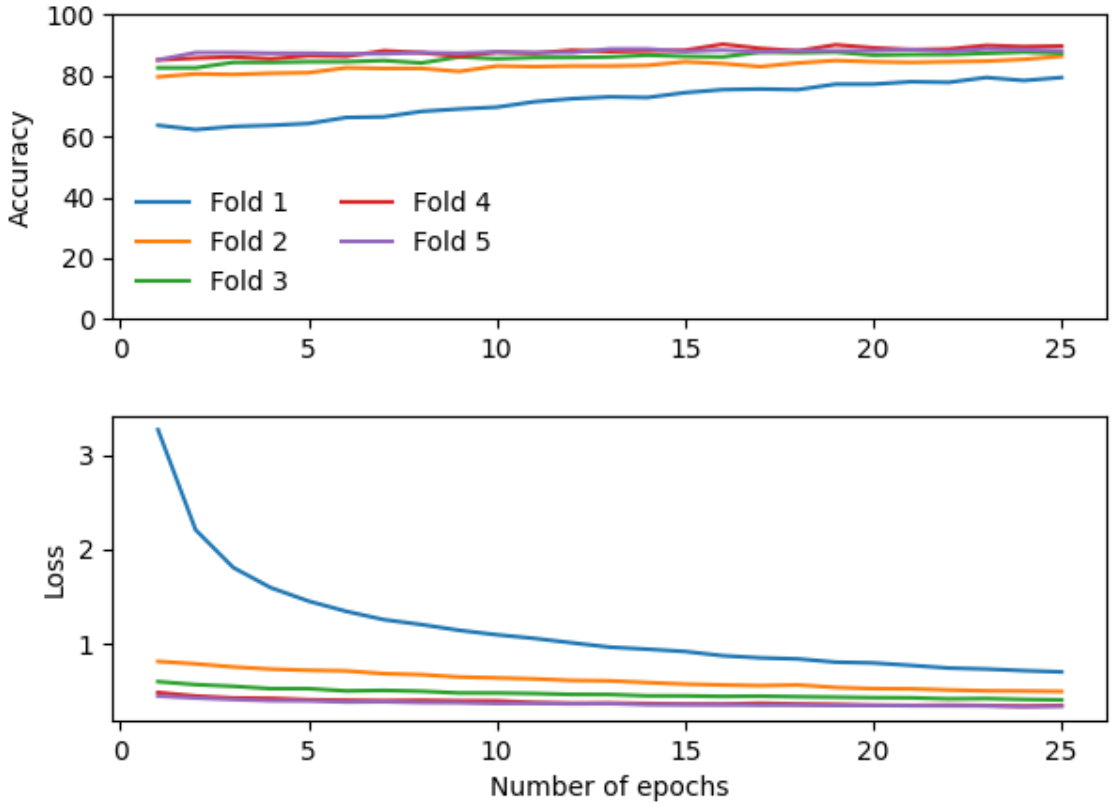


Figure 25: Accuracy and loss of the model including one hidden layer and 1200 nodes on the training set

Table 8: Accuracy of the model including one hidden layer and 1200 nodes tested with the available test set

Fold	Accuracy of the test set (%)
1	61.90
2	77.77
3	77.77
4	72.22
5	77.6
Average accuracy	73.45

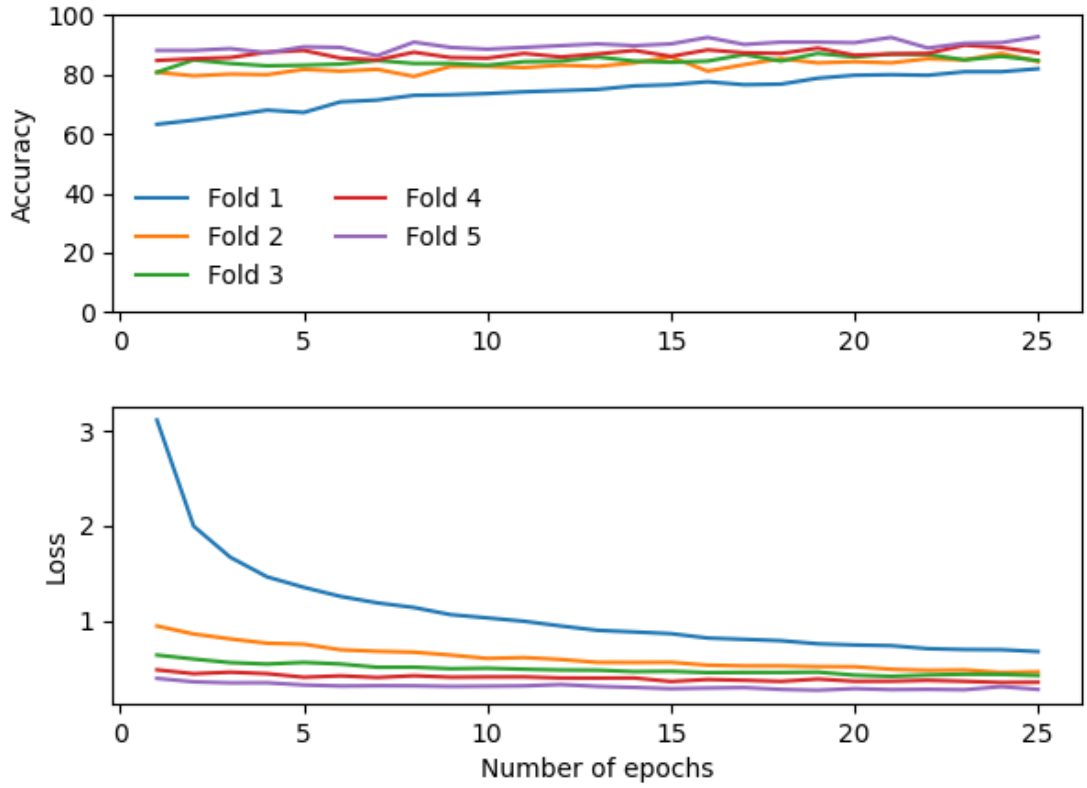


Figure 26: Accuracy and loss of the model including two hidden layers and 600 nodes in each layer on the training set

Table 9: Accuracy of the model including two hidden layers and 600 nodes tested with the available test set

Fold	Accuracy of the test set (%)
1	68.25
2	66.66
3	78.57
4	85.71
5	73.60
Average accuracy	74.56

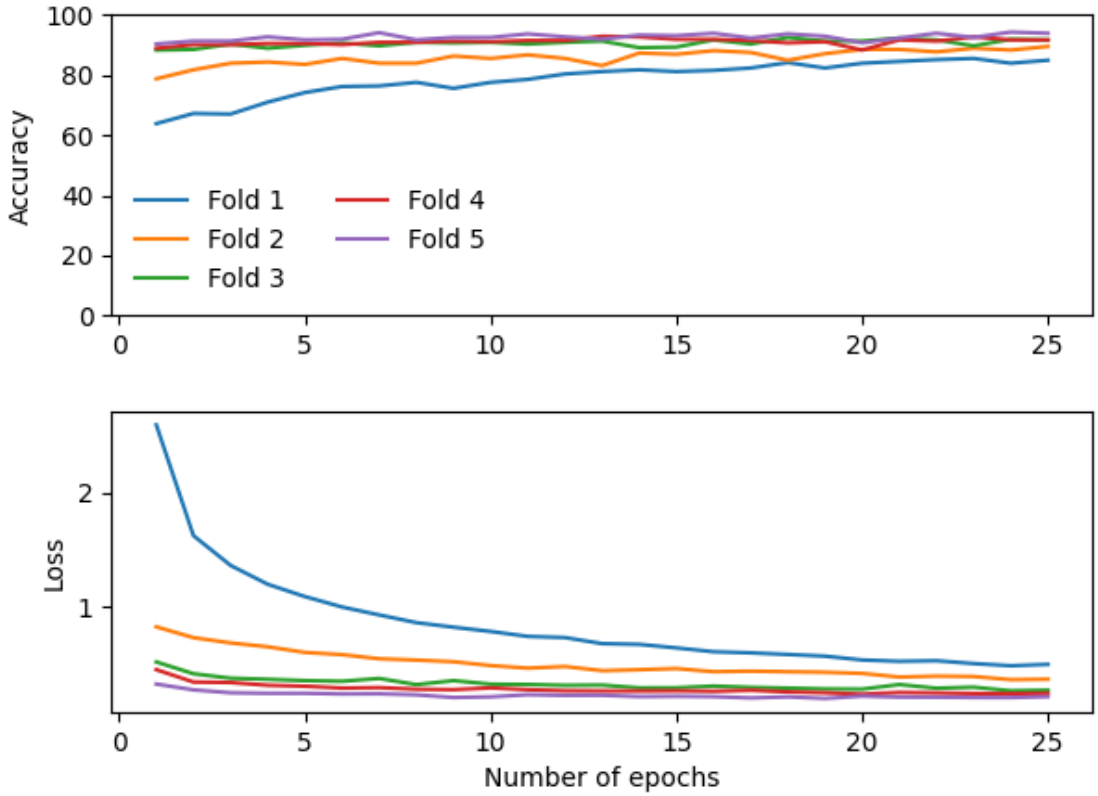


Figure 27: Accuracy and loss of the model including two hidden layers and 1200 nodes in each layer on the training set

Table 10: Accuracy of the model including two hidden layers and 1200 nodes tested with the available test set

Fold	Accuracy of the test set (%)
1	69.84
2	73.01
3	73.81
4	75.40
5	82.40
Average accuracy	74.90

The runtimes were measured over the entire training and testing process to assess the performance in providing a real-time solution and selecting the model with the highest accuracy and shortest runtime. Table 11 shows the runtimes of different models with one and two hidden layers and different numbers of nodes.

Table 11: The run time (s) of the proposed model for different numbers of hidden layers and processing nodes

Number of hidden layers	Number of processing elements		
	60	600	1200
	(s)	(s)	(s)
One	11.75	19.95	42.68
Two	15	72.98	317.87

Finally, the trade-off between the accuracy, runtime, number of hidden layers, and the number of processing nodes is illustrated by plotting them against each other. This comparison is noteworthy because, at some points, the number of hidden layers or processing elements increases while the accuracy decreases. Figure 28a shows that for a fixed number of processing elements (1200 nodes), by increasing the number of hidden layers, the accuracy first increases and then decreases. The model with three hidden layers has an accuracy of 75.36%, which was the highest accuracy. By increasing the number of hidden layers to four and five, the accuracy was kept decreasing. Figure 28b and Figure 28c demonstrate that by increasing the number of hidden layers or the processing nodes, the runtime increases, which results in delays in receiving the outputs.

Likewise, Figure 28d, which shows the model's accuracy against the runtime, indicates that the accuracy was first increasing to a certain point, and after that, it starts to decrease. In this case, the number of nodes was kept constant, and the number of hidden layers was increased. In order to select the most productive model that provides high accuracy within the minimum runtime, such analysis that was presented here is necessary. In this case, the model with two hidden layers and 1,200 nodes provided an accuracy of 74.9%, which is slightly lower than the model with three hidden layers, while the runtime for the model with two hidden layers is 317 seconds and the runtime for the model with three hidden layers is 476 seconds. In other words, the model with two hidden layers can provide almost the same accuracy within shorter runtime.

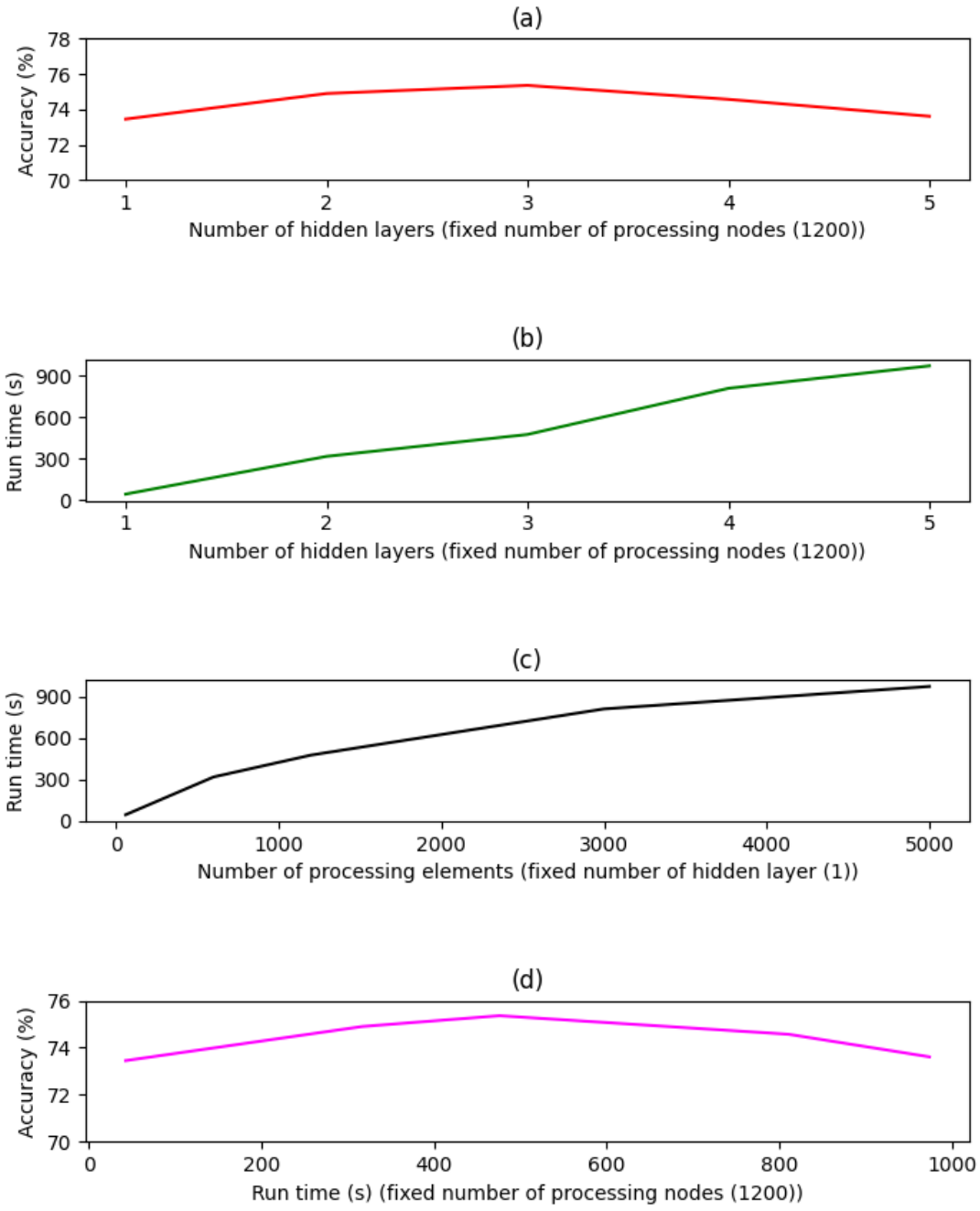


Figure 28: (a) Accuracy against the number of layers; (b) and (c) runtime against the number of hidden layers and the number of processing nodes, and (d) Accuracy against runtime

3.7 Conclusion and Future Work

Crane planning still has some limitations in terms of existing solutions and their efficiency. As part of a more extensive research study on crane planning, this paper proposed a machine learning technique incorporating an artificial neural network to classify and predict the most suitable mobile cranes regarding safety, cost, and scheduling for industrial projects. Various models were created and tested to find the one with higher efficiency; the model provides higher accuracy in a shorter time.

The proposed models were validated using test sets, which were selected randomly for each model. The selected model has two hidden layers in which 1,200 processing nodes for each layer exist. The accuracy of this model was measured as 74.90%, which was a high accuracy compared to other proposed models. The low computation cost of these algorithms enables lift engineers and construction managers to make a decision faster with higher confidence regarding the results. Besides, a connection was made between the neural network model in this research and the application that was proposed by Azami et al.(2021). The purpose of this connection was to use the results of the heuristic approach for training and testing the neural network. Moreover, a distinguishing characteristic of this work lies in the fact that it can be used for any type of cranes, such as a tower or mobile cranes. The model can be adjusted and used with a new training set to train the model for other types of cranes. This model can be a fast and accurate method for industrial projects to select mobile cranes. Moreover, this research can help other academic researchers to develop and implement machine learning techniques for automating construction processes.

Future work for this research is proposed in three main areas: improve the developed model in this research, develop a 3D visualization for the crane selection algorithm, and integration of this algorithm with previously developed algorithms for lift planning such as Automated Crane Planning and Optimization (ACPO) and Crane Motion Planning that has been designed for PCL Industrial Management to automate the process of checking the capacity and clearances of a crane (Lei 2011, 2014; Taghaddos et al. 2018). (1) Model improvement: Even though this paper investigated automated mobile crane selection, other cranes still need to be appropriately studied, such as tower and derrick cranes. In addition, the model can be modified so that it can be used for project cost prediction; (2) 3D visualization: Developing a 3D visualization can help lift engineers better understand the crane's coordinations regarding its surrounding objects. This application can help to improve the clash detection process or increase the safety of the project; (3) future work needs to study the integration of the crane planning applications that have been developed in the past few years by PCL to develop a comprehensive lift planner which can plan for the entire process automatically.

Appendix II: Neural Network

Training Dataset – Sample Data for Training the Neural Network

Weight (klb)	Height (ft)	Length (ft)	Gross load (klb)	Moment (klbf. ft)	Set Radius (ft)	Label
140.05	20	75	248	11,226	45	19403
180.75	17	51.	284	18,509	65	27225
121.50	61	21	193	10,790	55	22124
121.50	68	13	193	11,094	57	27644
174.76	24	108	308	32,783	106	27262
141.09	23	69	235	33,117	140	27262
137.50	25	78	276	19,427	70	27217
38.15	20	69	96	7,910	81	19403
75.72	24	70	250	11,590	46	27263
141.63	20	75	239	14,795	62	27217
132.18	20	76	237	15,310	64	27217
131.06	17	75	248	11,226	45	27263
140.05	20	75	284	18,509	65	27217
180.75	18	51	193	10,790	56	21691
121.50	62	21	193	11,094	57	21685
121.50	69	13	111	6,383	57	22131
55.85	79	13	596	61,289	103	27450
393.00	13	121	339	20,464	60	27217
232.70	75	15	337	21,191	63	27217
231.10	75	15	250	11,590	46	27263
…(total record 630)						

Python Code for the Neural Network

```
start = time.time()

#Creat the model using a for-loop

input_size = 10

hidden_size = [1200, 1200]

output_size = 58

class MultilayerPerceptron(nn.Module):

    def __init__(self, input_size = 10, hidden_size = [1200, 1200],

output_size = 58, p = 0.2):

        super().__init__()

        layerlist = []

        for i in hidden_size:

            layerlist.append(nn.Linear(input_size,i))

            layerlist.append(nn.ReLU())

            layerlist.append(nn.Dropout(p))

            input_size = i

            #print(layerlist)

        layerlist.append(nn.Linear(hidden_size[-1],output_size))

        self.hidden_size = nn.Sequential(*layerlist)

    def forward(self,X):

        X = self.hidden_size(X)

        return X

model = perceptron(input_size, hidden_size, output_size)

optimizer = optim.Adam(model.parameters(), lr = 0.0001)

lossFunction = nn.CrossEntropyLoss()

#reading the dataset

data = pd.read_excel('kir.xlsx')

data["Labels"] = data["Labels"].astype('category')
```

```
data["Labels"] = data["Labels"].cat.codes
# spliting data label and the data feature
data_label = data['Labels']
data.drop(columns=['Labels'], inplace=True)
data_feature = data.to_numpy()
data_label = data_label.to_numpy()
num_epochs = 25
k_folds = 5
kfold = KFold(n_splits=k_folds, shuffle=True)
# For fold results
results = {}
fig, axs = plt.subplots(2)
for fold, (train_index, test_index) in
enumerate(kfold.split(data_feature)):
    #variables to save the data to plot the results
    x = np.array([])
    y = np.array([])
    z = np.array([])
    # Print
    print(f'FOLD {fold}')
    print('--------------------------------------')
    X_train, X_test = data_feature[train_index],
data_feature[test_index]
    y_train, y_test = data_label[train_index], data_label[test_index]



    scaler_train = StandardScaler().fit(X_train)
    X_train_scaled = scaler_train.transform(X_train)
```

```
    scaler_test = StandardScaler().fit(X_test)
    X_test_scaled = scaler_test.transform(X_test)
    X_train_scaled_tensor = torch.FloatTensor(X_train_scaled)
    y_train_tensor = torch.LongTensor(y_train)
    X_test_scaled_tensor = torch.FloatTensor(X_test_scaled)
    y_test_tensor = torch.LongTensor(y_test)
    train_dataset = TensorDataset(X_train_scaled_tensor,
y_train_tensor)
    trainloader = DataLoader(train_dataset, batch_size= 5,
shuffle=True)
    test_dataset = TensorDataset(X_test_scaled_tensor, y_test_tensor)
    testloader = DataLoader(test_dataset, batch_size= 5, shuffle=True)
    # Run the training loop for defined number of epochs
    for epoch in range(0, num_epochs):
        #Print epoch
        print(f'Starting epoch {epoch+1}')
        # Set current loss value
        current_loss = 0.0
        running_loss = 0.0
        j = 0
        # Iterate over the DataLoader for training data
        for i, data in enumerate(trainloader, 0):
            j += 1
            # Get inputs
            inputs, targets = data
            # Zero the gradients
            optimizer.zero_grad()
            # Perform forward pass
```

```
            outputs = model(inputs)
            # Compute loss
            loss = lossFunction(outputs, targets)
            # Perform backward pass
            loss.backward()
            # Perform optimization
            optimizer.step()
            # Print statistics
            current_loss += loss.item()
            running_loss += loss.item()
            if i % 10 == 9:
            print('Loss after mini-batch %5d: %.3f' % (i + 1,
current_loss / 10))
            current_loss = 0.0
        output = model(X_train_scaled_tensor)
        #print("Epoch{}, Training loss:{}".format(epoch, loss_ /
len(trainloader)))
        true = (y_train_tensor == torch.max(output.data, 1)[1]).sum()
        accuracy = (true.item()/y_train.shape[0])* 100
        print(f'Epoch: {epoch:2}  Average loss: {running_loss/j :5.3f}
accuracy: {accuracy:7.3f}')
        x = np.append(x,epoch+1)
        y = np.append(y,accuracy)
        z = np.append(z, running_loss/j)
    # Process is complete.
    print('Training process has finished. Saving trained model.')
     # Print about testing
    print('Starting testing')
```

```
    # Saving the model
    save_path = f'./model-fold-{fold}.pth'
    torch.save(model.state_dict(), save_path)
    class_correct = list(0. for i in range(58))
    class_total = list(0. for i in range(58))
    # Evaluation for this fold
    correct, total = 0, 0
    with torch.no_grad():
      # Iterate over the test data and generate predictions
        for i, data in enumerate(testloader, 0):
        # Get inputs
            inputs, targets = data
        # Generate outputs
            outputs = model(inputs)
        # Set total and correct
            _, predicted = torch.max(outputs.data, 1)
            total += targets.size(0)
            correct += (predicted == targets).sum().item()
        print('Accuracy for fold %d: %d %%' % (fold, 100.0 * correct /
total))
        output = model(X_test_scaled_tensor)
        __, pre = torch.max(output.data,1)
print(metrics.classification_report(y_test_tensor,
torch.max(output.data, 1)[1]))
        print('--------------------------------')
        results[fold] = 100.0 * (correct / total)
    # Print fold results
    print(f'K-FOLD CROSS VALIDATION RESULTS FOR {k_folds} FOLDS')
```

```
    print('---------------------------------')
    sum = 0.0
    for key, value in results.items():
        print(f'Fold {key+1}: {value :.3f} %')
        sum += value
    print(f'Average: {sum/len(results.items()):.3f} %')
print(f'TOTAL TRAINING TIME : {time.time()-start:.3f}')
```

References

Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A., & Arshad, H. (2018). State-of-the-art in artificial neural network applications: A survey. *Heliyon*, *4*(11), e00938. https://doi.org/10.1016/j.heliyon.2018.e00938

Abudayyeh, O., & Al-Battaineh, H. T. (2003). As-Built Information Model for Bridge Maintenance. *Journal of Computing in Civil Engineering*, *17*(2), 105–112. https://doi.org/10.1061/(asce)0887-3801(2003)17:2(105)

Ahmed, M. O., Khalef, R., Ali, G. G., & El-adaway, I. H. (2021). Evaluating Deterioration of Tunnels Using Computational Machine Learning Algorithms. *Journal of Construction Engineering and Management*, *147*(10), 04021125. https://doi.org/10.1061/(ASCE)CO.1943-7862.0002162

Ahn, J., Ji, S.-H., Ahn, S. J., Park, M., Lee, H.-S., Kwon, N., Lee, E.-B., & Kim, Y. (2020). Performance evaluation of normalization-based CBR models for improving construction cost estimation. *Automation in Construction*, *119*, 103329. https://doi.org/10.1016/j.autcon.2020.103329

AL-Zwainy, F. M. S., & Aidan, I. A.-A. (2017). Forecasting the Cost of Structure of Infrastructure Projects Utilizing Artificial Neural Network Model (Highway Projects as Case Study). *Indian Journal of Science and Technology*, *10*(20), 1–12. https://doi.org/10.17485/ijst/2017/v10i20/108567

Aljarah, I., Faris, H., & Mirjalili, S. (2018). Optimizing connection weights in neural networks using the whale optimization algorithm. *Soft Computing*, *22*(1). https://doi.org/10.1007/s00500-016-2442-1

Almonacid, F., Fernandez, E. F., Mellit, A., & Kalogirou, S. (2017). Review of

techniques based on artificial neural networks for the electrical characterization of concentrator photovoltaic technology. *Renewable and Sustainable Energy Reviews*, *75*, 938–953. https://doi.org/10.1016/j.rser.2016.11.075

Angelini, E., di Tollo, G., & Roli, A. (2008). A neural network approach for credit risk evaluation. *The Quarterly Review of Economics and Finance*, *48*(4), 733–755. https://doi.org/10.1016/j.qref.2007.04.001

Azadi, Sama, & Karimi-Jashni, A. (2016). Verifying the performance of artificial neural network and multiple linear regression in predicting the mean seasonal municipal solid waste generation rate: A case study of Fars province, Iran. *Waste Management*, *48*, 14–23. https://doi.org/10.1016/j.wasman.2015.09.034

Azadi, Samira, & Sepaskhah, A. R. (2012). Annual precipitation forecast for west, southwest, and south provinces of Iran using artificial neural networks. *Theoretical and Applied Climatology*, *109*(1–2), 175–189. https://doi.org/10.1007/s00704-011-0575-9

Azami, R., Lei, Z., Hermann, R., & Zubick, T. (2021). *An Automated Mobile Crane Selection System for Heavy Industrial Construction Projects*. https://csce2021.ca/

Beavers, J. E., Moore, J. R., Rinehart, R., & Schriver, W. R. (2006). Crane-Related Fatalities in the Construction Industry. *Journal of Construction Engineering and Management*, *132*(9), 901–910. https://doi.org/10.1061/(asce)0733-9364(2006)132:9(901)

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press, Inc.

Bowden, G. J., Nixon, J. B., Dandy, G. C., Maier, H. R., & Holmes, M. (2006).

Forecasting chlorine residuals in a water distribution system using a general regression neural network. *Mathematical and Computer Modelling*, *44*(5–6), 469–484. https://doi.org/10.1016/j.mcm.2006.01.006

Chen, C. (2016). *Crane Planning Optimization for Construction*. Master of Science The University of Michigan.

Chen, P.-Y., Wu, Z. Y., & Taciroglu, E. (2021). Classification of Soft-Story Buildings Using Deep Learning with Density Features Extracted from 3D Point Clouds. *Journal of Computing in Civil Engineering*, *35*(3), 04021005. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000968

Cheng, J. C. P., & Wang, M. (2018). Automated detection of sewer pipe defects in closed-circuit television images using deep learning techniques. *Automation in Construction*, *95*, 155–171. https://doi.org/10.1016/j.autcon.2018.08.006

Cooper, C. N. (1987). *CRANES - A Rule-Based Assistant with Graphics for Construction Planning Engineers*. 47–54. https://doi.org/10.4203/ccp.6.2.4

Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., & Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, *542*(7639), 115–118. https://doi.org/10.1038/nature21056

Fan, C.-L. (2020). Defect Risk Assessment Using a Hybrid Machine Learning Method. *Journal of Construction Engineering and Management*, *146*(9), 04020102. https://doi.org/10.1061/(ASCE)CO.1943-7862.0001897

Faris, H., Aljarah, I., Al-Madi, N., & Mirjalili, S. (2016). Optimizing the Learning Process of Feedforward Neural Networks Using Lightning Search Algorithm. *International Journal on Artificial Intelligence Tools*, *25*(06), 1650033.

https://doi.org/10.1142/S0218213016500330

Figueroa, R. L., Zeng-Treitler, Q., Kandula, S., & Ngo, L. H. (2012). Predicting sample size required for classification performance. *BMC Medical Informatics and Decision Making*, *12*(1), 8. https://doi.org/10.1186/1472-6947-12-8

Firat, M., & Gungor, M. (2009). Generalized Regression Neural Networks and Feed Forward Neural Networks for prediction of scour depth around bridge piers. *Advances in Engineering Software*, *40*(8), 731–737. https://doi.org/10.1016/j.advengsoft.2008.12.001

Goodarzizad, P., Mohammadi Golafshani, E., & Arashpour, M. (2021). Predicting the construction labour productivity using artificial neural network and grasshopper optimisation algorithm. *International Journal of Construction Management*, 1–17. https://doi.org/10.1080/15623599.2021.1927363

Gray, C., & Little, J. (1985). A systematic approach to the selection of an appropriate crane for a construction site. *Construction Management and Economics*, *3*(2), 121–144. https://doi.org/10.1080/01446198500000010

Hanna, A. S., & Lotfallah, W. B. (2004). A Fuzzy logic approach to selection of cranes. *Journal of Structural Engineering (Madras)*, *30*(4), 215–224.

Hasan, M. S. (2013). *Decision Support System for Crane Selection and Location Optimization on Construction Sites*. PhD Thesis University of Alberta (Canada).

Haykin, S. (2019). Neural Networks: A Comprehensive Foundation. In *Encyclopedia of Bioinformatics and Computational Biology* (Vols. 1–3, p. 840). Prentice Hall PTR. https://linkinghub.elsevier.com/retrieve/pii/B9780128096338203397

Hazarika, N., Chen, J. Z., Tsoi, A. C., & Sergejew, A. (1997). Classification of EEG

signals using the wavelet transform. *Signal Processing*, *59*(1), 61–72. https://doi.org/10.1016/S0165-1684(97)00038-8

Heidari, A. A., Faris, H., Aljarah, I., & Mirjalili, S. (2019). An efficient hybrid multilayer perceptron neural network with grasshopper optimization. *Soft Computing*, *23*(17), 7941–7958. https://doi.org/10.1007/s00500-018-3424-2

Hermann, U. H., Hasan, S., Al-Hussein, M., & Bouferguene, A. (2011). Innovative System for Off-the-Ground Rotation of Long Objects Using Mobile Cranes. *Journal of Construction Engineering and Management*, *137*(7), 478–485. https://doi.org/10.1061/(asce)co.1943-7862.0000309

Ho, Y., & Wookey, S. (2020). The Real-World-Weight Cross-Entropy Loss Function: Modeling the Costs of Mislabeling. *IEEE Access*, *8*, 4806–4813. https://doi.org/10.1109/ACCESS.2019.2962617

Hornaday, W. C., Haas, C. T., O'Connor, J. T., & Wen, J. (1993). Computer-Aided Planning for Heavy Lifts. *Journal of Construction Engineering and Management*, *119*(3), 498–515. https://doi.org/10.1061/(ASCE)0733-9364(1993)119:3(498)

Hykin, S. (1999). *Neural Networks: a comprehensive foundation*. Printice-hall.

Ilbeigi, M., Ghomeishi, M., & Dehghanbanadaki, A. (2020). Prediction and optimization of energy consumption in an office building using artificial neural network and a genetic algorithm. *Sustainable Cities and Society*, *61*, 102325. https://doi.org/10.1016/j.scs.2020.102325

Jain, A. K., Jianchang Mao, & Mohiuddin, K. M. (1996). Artificial neural networks: a tutorial. *Computer*, *29*(3), 31–44. https://doi.org/10.1109/2.485891

Kocyigit, Y., Alkan, A., & Erol, H. (2008). Classification of EEG Recordings by Using

Fast Independent Component Analysis and Artificial Neural Network. *Journal of Medical Systems*, *32*(1), 17–20. https://doi.org/10.1007/s10916-007-9102-z

Lei, Z. (2011). *A Robotic Approach to the Analysis of Obstacle Avoidance in Crane Lift*. Master of science thesis University of Alberta.

Lei, Z. (2014). *Automated Simulation Model for Crane Motion Planning in Heavy Industrial Projects* (Vol. 1) [PhD Thesis University of Alberta]. https://doi.org/10.1017/CBO9781107415324.004

Leśniak, A., & Juszczyk, M. (2018). Prediction of site overhead costs with the use of artificial neural network based model. *Archives of Civil and Mechanical Engineering*, *18*(3), 973–982. https://doi.org/10.1016/j.acme.2018.01.014

Li, L., Doroslovacki, M., & Loew, M. H. (2020). Approximating the Gradient of Cross-Entropy Loss Function. *IEEE Access*, *8*, 111626–111635. https://doi.org/10.1109/ACCESS.2020.3001531

Mohammadhassani, M., Nezamabadi-pour, H., Jumaat, M. Z., Jameel, M., & Arumugam, A. M. S. (2013). Application of artificial neural networks (ANNs) and linear regressions (LR) to predict the deflection of concrete deep beams. *Computers & Concrete*, *11*(3), 237–252. https://doi.org/10.12989/cac.2013.11.3.237

Mohebali, B., Tahmassebi, A., Meyer-Baese, A., & Gandomi, A. H. (2020). Probabilistic neural networks. In *Handbook of Probabilistic Models* (pp. 347–367). Elsevier. https://doi.org/10.1016/B978-0-12-816514-0.00014-X

Montazer, G. A., & Giveki, D. (2015). An improved radial basis function neural network for object image retrieval. *Neurocomputing*, *168*, 221–233. https://doi.org/10.1016/j.neucom.2015.05.104

Moon, S., & Munira Chowdhury, A. (2021). Utilization of Prior Information in Neural Network Training for Improving 28-Day Concrete Strength Prediction. *Journal of Construction Engineering and Management*, *147*(5), 04021028. https://doi.org/10.1061/(ASCE)CO.1943-7862.0002047

Moselhi, O., Alkass, S., & Al-Hussein, M. (2004). Innovative 3D-modelling for selecting and locating mobile cranes. *Engineering, Construction and Architectural Management*, *11*(5), 373–380. https://doi.org/10.1108/09699980410558575

Mund, A. T. (1999). *Intellicrances – A Neural Network-Based Crane Selection System* [Wester Michigan University]. https://scholarworks.wmich.edu/masters_theses/4851/?utm_source=scholarworks.wmich.edu%2Fmasters_theses%2F4851&utm_medium=PDF&utm_campaign=PDFCoverPages

Naik, M. G., & Radhika, V. S. B. (2015). Time and Cost Analysis for Highway Road Construction Project Using Artificial Neural Networks. *Journal of Construction Engineering and Project Management*, *5*(1), 26–31. https://doi.org/10.6106/JCEPM.2015.5.1.026

Networks., E. M. for T. N., & Trajan Software Ltd.: Co. Durham, U. K. (1999). *Trajan Software Ltd. (1999)*.

O'Connor, J. T., O'Brien, W. J., & Choi, J. O. (2014). Critical Success Factors and Enablers for Optimum and Maximum Industrial Modularization. *Journal of Construction Engineering and Management*, *140*(6), 04014012. https://doi.org/10.1061/(asce)co.1943-7862.0000842

Oğulata, S. N., Şahin, C., & Erol, R. (2009). Neural Network-Based Computer-Aided

Diagnosis in Classification of Primary Generalized Epilepsy by EEG Signals. *Journal of Medical Systems*, *33*(2), 107–112. https://doi.org/10.1007/s10916-008-9170-8

Ojha, V. K., Abraham, A., & Snášel, V. (2017). Metaheuristic design of feedforward neural networks: A review of two decades of research. *Engineering Applications of Artificial Intelligence*, *60*, 97–116. https://doi.org/10.1016/j.engappai.2017.01.013

Ozturk, A. U., & Turan, M. E. (2012). Prediction of effects of microstructural phases using generalized regression neural network. *Construction and Building Materials*, *29*, 279–283. https://doi.org/10.1016/j.conbuildmat.2011.10.015

Parzen, E. (1962). On Estimation of a Probability Density Function and Mode. *The Annals of Mathematical Statistics*, *33*(3), 1065–1076. https://doi.org/10.1214/aoms/1177704472

Puig-Arnavat, M., Hernández, J. A., Bruno, J. C., & Coronas, A. (2013). Artificial neural network models for biomass gasification in fluidized bed gasifiers. *Biomass and Bioenergy*, *49*, 279–289. https://doi.org/10.1016/j.biombioe.2012.12.012

Rafiei, M. H., & Adeli, H. (2018). Novel Machine-Learning Model for Estimating Construction Costs Considering Economic Variables and Indexes. *Journal of Construction Engineering and Management*, *144*(12), 04018106. https://doi.org/10.1061/(ASCE)CO.1943-7862.0001570

Rácz, A., Bajusz, D., & Héberger, K. (2021). Effect of Dataset Size and Train/Test Split Ratios in QSAR/QSPR Multiclass Classification. Molecules, 26(4), 1111. https://doi.org/10.3390/molecules26041111

Raynar, K. A., & Smith, G. R. (1993). Intelligent Positioning of Mobile Cranes for Steel

Erection. *Computer-Aided Civil and Infrastructure Engineering*, *8*(1), 67–74. https://doi.org/10.1111/j.1467-8667.1993.tb00193.x

Rivara, F. ., & Alexander, B. . (2007). *Occupational Injuries in Clinical, Occupational, and Environmental Medicine*.

Roysson, S., Sitompul, T. A., & Lindell, R. (2021). *Using Artificial Neural Network to Provide Realistic Lifting Capacity in the Mobile Crane Simulation* (pp. 448–462). https://doi.org/10.1007/978-3-030-80568-5_37

Sawhney, A., & Mund, A. (2002). Adaptive Probabilistic Neural Network-based Crane Type Selection System. *Journal of Construction Engineering and Management*, *128*(3), 265–273. https://doi.org/10.1061/(ASCE)0733-9364(2002)128:3(265)

Shapira, A., & Glascock, J. D. (1996). Culture of Using Mobile Cranes for Building Construction. *Journal of Construction Engineering and Management*, *122*(4), 298–307. https://doi.org/10.1061/(ASCE)0733-9364(1996)122:4(298)

Shapira, A., & Schexnayder, C. J. (1999). Selection of mobile cranes for building construction projects. *Construction Management and Economics*, *17*(4), 519–527. https://doi.org/10.1080/014461999371439

Sharma, S., Sharma, S., & Athaiya, A. (2020). Activation Functions in Neural Networks. *International Journal of Engineering Applied Sciences and Technology*, *04*(12), 310–316. https://doi.org/10.33564/IJEAST.2020.v04i12.054

Shehadeh, A., Alshboul, O., Al Mamlook, R. E., & Hamedat, O. (2021). Machine learning models for predicting the residual value of heavy construction equipment: An evaluation of modified decision tree, LightGBM, and XGBoost regression. *Automation in Construction*, *129*, 103827.

https://doi.org/10.1016/j.autcon.2021.103827

Specht, D. (1991). A general regression neural network. *IEEE Transactions on Neural Networks*, *2*, 568–576.

Subasi, A. (2007). EEG signal classification using wavelet feature extraction and a mixture of expert model. *Expert Systems with Applications*, *32*(4), 1084–1093. https://doi.org/10.1016/j.eswa.2006.02.005

Subasi, A., & Erçelebi, E. (2005). Classification of EEG signals using neural network and logistic regression. *Computer Methods and Programs in Biomedicine*, *78*(2), 87–99. https://doi.org/10.1016/j.cmpb.2004.10.009

Taghaddos, H., Hermann, U., & Abbasi, A. B. (2018). Automated Crane Planning and Optimization for modular construction. *Automation in Construction*, *95*(July), 219–232. https://doi.org/10.1016/j.autcon.2018.07.009

Tijanić, K., Car-Pušić, D., & Šperac, M. (2020). Cost estimation in road construction using artificial neural network. *Neural Computing and Applications*, *32*(13), 9343–9355. https://doi.org/10.1007/s00521-019-04443-y

Übeyli, E. D. (2009). Combined neural network model employing wavelet coefficients for EEG signals classification. *Digital Signal Processing*, *19*(2), 297–308. https://doi.org/10.1016/j.dsp.2008.07.004

Wang, D., Wang, X., Ren, B., Wang, J., Zeng, T., Kang, D., & Wang, G. (2022). Vision-Based Productivity Analysis of Cable Crane Transportation Using Augmented Reality–Based Synthetic Image. *Journal of Computing in Civil Engineering*, *36*(1). https://doi.org/10.1061/(ASCE)CP.1943-5487.0000994

Wing Lun Chiu, A. (2002). Hybrid Neural Networks Using Artificial Neural Networks

for the A nalysis and Control of Biological Neural Networks. In *Dissertation: Vol. Master the* (Issue Structural Biology and Molecular Biophysics, University of Pennsylvania, PA, USA.).

Wu, D., Lin, Y., Wang, X., Wang, X., & Gao, S. (2011). Algorithm of crane selection for heavy lifts. *Journal of Computing in Civil Engineering*, *25*(1), 57–65. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000065

Zhu, J., Zhong, J., Ma, T., Huang, X., Zhang, W., & Zhou, Y. (2022). Pavement distress detection using convolutional neural networks with images captured via UAV. *Automation in Construction*, *133*, 103991. https://doi.org/10.1016/j.autcon.2021.103991

Chapter 4: Summary and Conclusions

Mobile cranes have been used frequently in industrial projects to install heavy modular components onsite. Cost-saving and schedule-shortening are subject to appropriate crane planning, which includes: 1) crane type selection; 2) crane model selection; 3) crane potential location; 4) lifting sequence; and 5) feasibility of lifts. Many factors (e.g., lifting capacity, clearance) can affect the feasibility of lifts. The current manual-based process for crane model selection is time-consuming, and errors might occur when more criteria need to be considered. In order to increase the efficiency of the planning process and avoid errors, two different algorithms have been proposed. A heuristic algorithm was first developed that simultaneously considers the project duration, the monthly rate of the cranes, the vertical and horizontal clearance, the crane capacity, the super-lift existence, and the safety factor. This algorithm utilizes a scoring technique to rank the results based on each mobile crane configuration; the closer the crane configuration is to the optimum solution, the higher its score is. Finally, the algorithm finds all the possible crane configurations for lifting the modules and ranks them based on their scores. Thus, the lift engineers would select the most suitable mobile crane based on its score.

Furthermore, a neural network algorithm has been developed, which works based on historical data, to predict the crane configurations for future projects. This algorithm has been trained and tested by the historical data, and the accuracy of the results on the test set was 75.36%. Multilayer Perceptron Neural Network was implemented with multiple hidden layers. Having run the model with different hidden layers and processing

nodes, the model with three hidden layers and 1200 processing nodes provided the highest accuracy. However, the tradeoff between the runtime and the accuracy demonstrated that the model with two hidden layers and the same number of processing nodes could achieve slightly lower accuracy, 74.90%, in a shorter runtime. A comparison between the heuristic and neural network results showed that the runtime of the heuristic approach was 10 minutes long. In contrast, the neural network could solve the same problem in slightly less than a minute. In addition, the results of the heuristic approach were more effective than the neural network since the database that was used to train the NN model was not as comprehensive as it should be. One way to improve the results of the neural network model is to train it with a more extensive database so that the model can predict optimal solutions.

Finally, a connection between the two algorithms has been made through the interface so that the user has access to both the algorithms. This connection provides the user with the option to use either method or both. In addition to this option, the user could use the heuristic method to generate training data for the ANN method. All the features mentioned above were implemented to increase the system's productivity and make it more user-friendly. Both of the algorithms developed in this research can be used in industrial projects to automate crane selection, which decreases the cost and duration of the projects. Furthermore, selecting the right crane can increase the safety of the workers and engineers on the job site.

From the theoretical point of view, developing and connecting two different algorithms, which consider the most critical parameters for crane selection simultaneously, has been done in this research for the first time. Since implementing and

utilizing machine learning techniques in the construction industry is relatively new, this research can be a helpful source for other researchers to learn how to take advantage of the power of machine learning techniques to automate construction processes and increase the productivity of the construction industry.

4.1 Contributions

A system that automatically selects crane configurations was developed through a cooperative initiative between the University of New Brunswick and PCL Industrial Management Inc. The purpose of the proposed algorithm was to automate mobile crane selection for industrial construction projects.

The proposed system, Crane Configurator, comprises two approaches: heuristic and neural network. The heuristic algorithm, which considers the modules' features, duration of the projects, the cranes' monthly rate, the cranes' capacity, the safety criteria, and the super-lift capacity simultaneously, enables the user to find a list of crane configurations for each construction project. This algorithm ranks the results based on the range of the criteria defined by the user, resulting in a list of the suitable mobile crane for the project.

The neural network algorithm was developed using machine learning techniques and trained by historical data to predict mobile crane configurations for industrial construction projects. In other words, the model was provided with a set of modules and the cranes that were used for lifting in the past, and the model forms a relationship between them in a way that can be generalized to other modules. The model has included a set of layers (i.e., the input, hidden, and output layers). The input layer receives the

features of the modules and pass them to the hidden layer. In the meantime, the features are multiplied by the associated weight of each connection and summed. A function called “Activation Function” decides whether the node in the hidden layer should be activated or not. The final results are then passed to the output layer, indicating the probability of lifting the module with each crane configuration. The crane configuration with the highest probability is considered to be the best option for lifting the module.

Finally, the two algorithms were connected; therefore, lift engineers can use both of them simultaneously and compare the results. In addition, the heuristic algorithm can generate the training dataset for the neural network algorithm, and the trained model can be used to predict the mobile cranes for industrial projects.

The contributions presented in this thesis (listed below) address the first and second stages of the Crane Configurator.

1. Develop two different methods for selecting the crane configurations, which can be used in parallel to validate the results or in series.
2. Develop a heuristic algorithm for mobile crane configuration selection in industrial projects that considers the duration of projects, the monthly rate of the cranes, and the super-lift existence simultaneously for the first time.
3. The neural network developed in this research can be utilized for selecting any type of crane provided the required dataset for training and testing is available, although this research was specifically related to mobile cranes.
4. In comparison to the manual method, both of the developed tools are capable of providing results in significantly shorter time. The neural network model, on the other hand, is the faster of the two methods.

5. The developed application can be improved by other researchers and implemented in other areas in construction industry such as cost prediction, scheduling, and production prediction.
6. Implementing this algorithm into the PCL's crane management system would make it fully automated since selecting the crane location, sequence of lifting the modules, and the path planning for the cranes have been done previously.

Curriculum Vitae

Candidate's full name: Ramtin Azami

Universities attended (with dates and degrees obtained):

University of New Brunswick 2020-2021 Master of Science

University of Tarbiat Modares 2014 – 2017 Master of Science

Tabriz university 2010 – 2014 Bachelor of Science

Publications:

Azami R, Lei Z, Hermann R, and Zubick T. 2021. "An Automated Mobile Crane Selection System For Heavy Industrial Construction Projects." CSCE 2021 Annual Conference of the Canadian Society for Civil Engineering, proceeding of construction conference

Conference Presentations: Canadian Society for Civil Engineering Annual Conference May 26-29, 2021