35th European Modeling & Simulation Symposium
20th International Multidisciplinary Modeling & Simulation Multiconference

2724-0029 © 2023 The Authors.
doi: 10.46354/i3m.2023.emss.042

A proposal based on Petri nets and Artificial Neural Networks for modeling Biotechnological process

Joselito Medina-Marin[1,*], Maria Guadalupe Serna-Diaz[2], Norberto Hernandez-Romero[1], Juan Carlos Seck-Tuoh-Mora[1] and Irving Barragan-Vite[1]

[1]Área Académica de Ingeniería, Universidad Autónoma del Estado de Hidalgo, Carr. Pachuca-Tulancingo Km. 4.5, Col. Carboneras, Mineral de la Reforma, Hidalgo, 42184, México
[2]Área Académica de Química, Universidad Autónoma del Estado de Hidalgo, Carr. Pachuca-Tulancingo Km. 4.5, Col. Carboneras, Mineral de la Reforma, Hidalgo, 42184, México

*Corresponding author. Email address: jmedina@uaeh.edu.mx

Abstract

Biotechnology is an interesting area that addresses global challenges related to human health, agricultural productivity, sustainability, among others. Biotechnology processes require the development of many experiments to study and understand the relationship of the variables participating in a process. On the other hand, this relationship can be modeled by using artificial neural networks. Moreover, the training phase in artificial neural network development can be improved with the use of Petri Net mathematical tools for generating the arc weights with matrix operations. This study proposes the use of graphical and mathematical representation of Petri nets in order to be trained as an Artificial Neural Network. The result is an extended Petri net model, named Neural Petri Net (NPN), which provides basic structures to create Petri net models with learning capabilities. The proposed algorithm is an adaptation of the known backpropagation algorithm, and the incidence matrix and state equation of Petri nets are used to calculate the output value of the trained NPN. The NPN was used to model a biotechnological process to obtain lignocellulosic biomass, showing a good performance in the forecasting.

Keywords: Petri nets, Artificial neural networks, Biotechnological process, State equation

1. Introduction

Characterization of biotechnological processes requires the development of many experiments to understand the relationship among the significative variables of the system. From experimental data, many models have been created to understand the complexity biotechnological systems and control their behavior (Noll & Henkel, 2020)

Advances in technology and computer science have allowed the development of complex modeling in biotechnology (Klyuchko, 2018). With the use of models, dynamic behavior of biotechnological systems can be simulated over the time without invasion in the. systems.

Mathematical methods have been applied for modeling and data processing in biotechnology. In addition, computational tools are used to analyzed data and create models to replicate systems behavior. (Klyuchko, 2017)

These methodologies applied in biotechnological processes modeling includes artificial neural networks (ANN), cluster analysis and image processing. (Niazian & Niedbała, 2020)

On the other side, Petri nets (PNs) have been applied to model manufacturing systems interactions, however, the same methodology was applied to create computational biology models and perform flux balance analysis (Simone et al., 2020).

ANNs training requires an iterative process to find the optimal arc wight values, which is time consuming. And PN mathematical tools can be used to speed up the calculations of arc weghts.

In this paper, the use of a hybrid model based on Petri nets (PN) and Artificial Neural Networks (ANN) is proposed. Both tools have been developed successfully in separate ways and each one provides its own advantages in its application areas. However, there are similarities, such as parallel processing, graphical representation, and mathematical calculations, that should be considered to improve their performance working together.

The reminder of this paper is organized as follows. In Section 2 are presented the works related to PN theory application to model biotechnological processes and some hybrid models of PNs and ANNs. Next, PN and ANN basis are described in Section 3. The results and discussion are given in Section 4. Finally, the conclusions are presented in Section 5.

2. State of the art

Biotechnological processes modeling has been done from many perspectives. PN theory is not the exception, and there are some works published related to the use of PNs in biotechnological modeling.

In 2021, Amstein *et al*. proposed a PN model for molecular reactions to find pathways in cellular response.

In another research work, PNs were applied to model several critical processes involved in ontogenesis. The authors combined PNs in a hierarchical way (nets-within-nets) to represent and simulate the ontogenesis processes. (Bardini et al., 2021)

The work presented by Kardynska et al. in 2023, describes the use of PNs for searching molecular targets for drugs. Moreover, PN analysis tools were used to identify the key elements in the regulatory network.

Some papers have been published describing the use of PNs and ANNs. In (Schuster, 2007), the author proposes a technique based on a "tokenized artificial neural network". He applied his technique to a basic perceptron-type network. A PN is trained with the use of colored tokens for classification purposes.

In another research work, the authors developed a system composed of a multilayer perceptron and a PN to control multiple mobile robots. PN model and the multilayer perceptron are collaborating in the system, but each one has particular functions and does not work together. (Pham et al., 2003)

A Dynamic Petri Recurrent Fuzzy Neural Network is proposed in (Wai & Lin, 2012). This PN model was developed for a vision-based mobile robot scenario. In (Tan, 2015), the author proposed a Wavelet Petri Fuzzy Neural Network Control, which is used to control squirrel-cage induction generators for grid-connected wind power applications. Nevertheless, in those research works the mathematical power of PNs is not used in the learning and execution process.

PN theory offers a graphical representation of systems and mathematical operations to analyze their state space. These tools and can be used for modeling ANNs. Thus, in this paper, we propose the implementation of learning algorithms in PN structures by using the graphical properties of PNs and matrix operations.

3. Materials and Methods

The proposal presented in this paper is based on concepts of PNs and ANNs.

3.1. Petri nets concepts

A Petri net is a powerful graphical and mathematical tool for modeling discrete event driven systems (Peterson, 1977; Zhou). The graph is directed, *arc* weighted, and has two types of nodes: *places* (circles) and *transitions* (boxes).

In a PN, places are connected to transitions, and transitions to places, but nodes of the same type are not allowed to be connected. The system state is represented using *tokens*, which are put in the corresponding PN places.

A formal definition of a PN is showed in Table 1. (Murata, 1989)

Table 1. Formal definition of a PN.

A Petri Net is a 5-tuple, $PN=(P, T, F, W, M_0)$ where:
$P = \{p_1, p_2, \dots, p_m\}$ is a finite set of places.
$T = \{t_1, t_2, \dots, t_n\}$ is a finite set of transitions.
$F \subseteq \{P \times T\} \cup \{T \times P\}$ is a set of arcs.
$W = F \to \{1,2,3, \dots\}$ is a weight function.
$M_0 = P \to \{0,1,2, \dots\}$ is the initial marking.
$P \cap T = \emptyset$ and $P \cup T \neq \emptyset$

The token game animation is used to represent the dynamical behavior of the system modeled through PNs. The change in the token position is based on the following transition firing rule: (Murata, 1989)

1. A transition $t \in T$ is enabled if each input place $p \in P$ of t contains at least $w(p,t)$ tokens.
2. An enabled transition t will be fired if the event denoted by t occurs.
3. Once an enabled transition t is fired, $w(p,t)$ tokens are deleted from each input place p of t, and $w(t,p)$ tokens are put on every output place p of t.

In addition, the BAP modeling needs the use of processing time. Thus, an extension of PN model with time in places is applied. A Timed Place Petri Net (TPPN) is a six-tuple $TPPN = \{P, T, F, W, M_0, D\}$, where $D = \{d_1, d_2, \dots, d_m\}$ represents the processing time for every place $p \in P$. (Zhao et al., 2011). In this case, the first transition firing rule for PNs now considers not only the number of tokens in the input places, also the time d_i indicated for place p_i must be reached.

3.2. Incidence Matrix and state equation

In this paper, the matrix equation of PN theory is used to simulate the token game animation in the production line, obtain the total of products processed, and finally calculate the throughput of the system.

A PN with n transitions and m places can be denoted as a matrix of integers $A = [a_{ij}]$ with dimension $n \times m$. The a_{ij} values are calculated with the mathematical expression $a_{ij} = a_{ij}^{+} - a_{ij}^{-}$, where a_{ij}^{+} is the arc weight $w(t_i, p_j)$ and a_{ij}^{-} is the arc weight $w(p_j, t_i)$.

The state equation is helpful to determine the marking in a PN through the transition firings. This equation is as shown in eq. 1.

$$M_k = M_{k-1} \times A^T U_k, \qquad k = 1,2,\dots \tag{1}$$

where U_k is a $n \times 1$ column vector of zeros and a unique one. The position of the one value indicates the transition t_j that will be fired. A^T is the transpose of the incidence matrix A. M_{k-1} is the marking before t_j is fired. And M_k is the marking reached after the firing of t_j.

3.3. Artificial Neural Networks

An Artificial Neural Networks (ANN) is part of computer intelligence, it contains a set of units named neurons, which are connected to perform classification tasks, pattern recognition, function fitting, among other applications. Moreover, ANNs have the ability to learn and have been applied to solve problems in a wide range of areas (van Gerven & Bohte, 2017; Chen et al., 2019).

An ANN is composed of a set of inputs, synaptic weights, a propagation rule, and an activation function. It can be applied to create models with incomplete information, is fault tolerant and its memory is distributed, and parallelism can be used for its processing. (Wen et al., 2020)

The features of ANNs are the following: (Aggarwal, 2018)

1. Each input node i is related to an input variable x_i.
2. Arc connections between nodes i and j have a weight $w_{ij} \in \Re$.
3. A threshold θ_i is defined for each node i.
4. The new state for each node i is defined by an activation function $f_i(x_j, w_{ij}, \theta_i)$, which depends on the states of nodes j (x_j), the weights of arcs connection (w_{ij}), and the threshold θ_i.

The graphical representation of an ANN is shown in Figure 1. It is composed of three types of layers: input, hidden and output layers. Every node in the input layer is relate to only one input state variable. Hidden layers can be composed of one or more layers. Finally, the output layer contains the response variables of the system.

The information travels through every ANN layer, which is updated in a progressive processing. The function applied to each neuron with n inputs and m outputs is expressed in eq. 2.

$$y_i(t) = f\left(\sum_{j=1}^{n} w_{ij} x_j - \theta_i\right), \forall i, 1 \le i \le m \tag{2}$$

The input layer sends the information received by each neuron to the next hidden layer, and do not compute anything.

During the training phase, arc weights (w_{ij}) are updated in order to obtain similar values to the target used as input data.

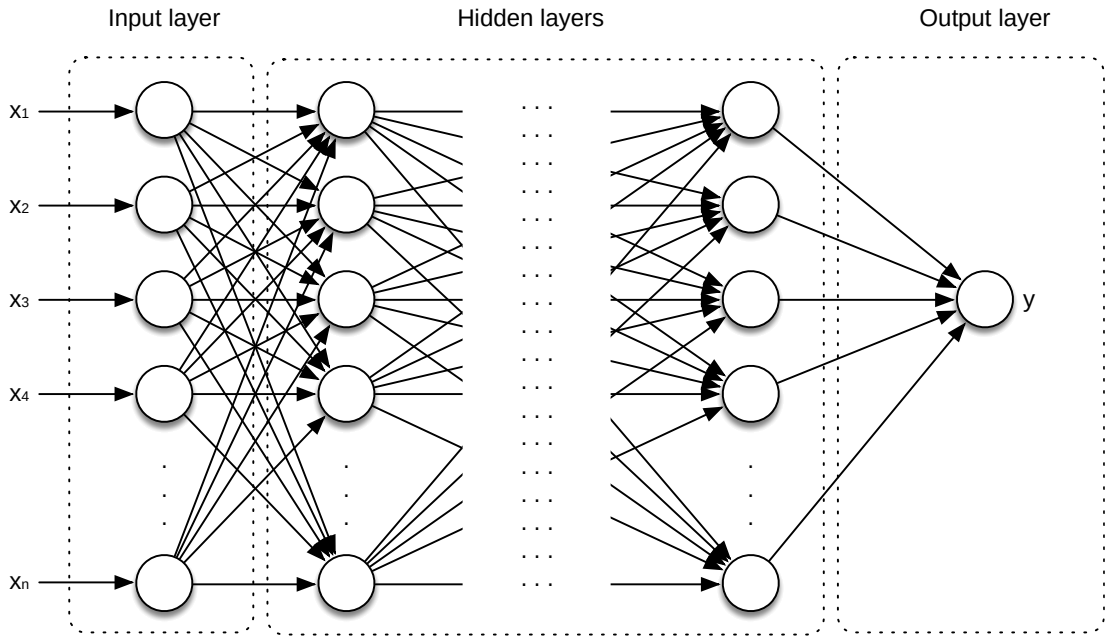

Figure 1. Structure of an ANN with one input layer, the hidden layers, and one output layer.

3.4. The biotechnological process

The data used in this modeling were gathered from lab experiments for obtaining lignocellulosic biomass of barley straw. The input variables considered are the sulfite concentration (1%, 5%, and 10%), particle size (mesh number 8, 12 and 20), and reaction time (30, 60, and 90 minutes). 27 experiments were made 3 times, and the data were used for training the ANN based on PNs.

4. Results and Discussion

In this paper, an extended Petri net model enhanced with learning capabilities is introduced, which is called Neural Petri Network (NPN). The learning process is based on the backpropagation algorithm, adapted to be applied to a PN structure.

4.1. Extended Petri Net model

The basic elements used to model a neuron are shown in Figure 2. The neuron depicted in Figure 2(a) is used to represent input values to the NPN. Place pa holds a token with input value $x_i \in \mathbb{R}$, this token is consumed by t_j and it is replicated to the output places of t_j.

Figure 2(b) denotes the neuron model used in the hidden layers. Each neuron model receives the result of the dot product of vector X (containing the token values) and W (denoting the weights for each arc) $\sum_{i=1}^{n} x_i w_i - \theta_i$, the result is stored in place p_b as a token. This token is consumed by transition t_k, where the token value less the bias (θ_i) is evaluated in the activation function $f_1(\sum_{i=1}^{n} x_i w_i - \theta_i)$, the result is stored in a token and it is replicated to the output places of t_k.

Place p_c of the output neuron shown in Figure 2(c) has the same behavior as p_b. It sends the resulting token to transition t_l, where activation function f_2 performs a similar evaluation as f_1, and the result is sent to p_d. as the output value y of NPN.

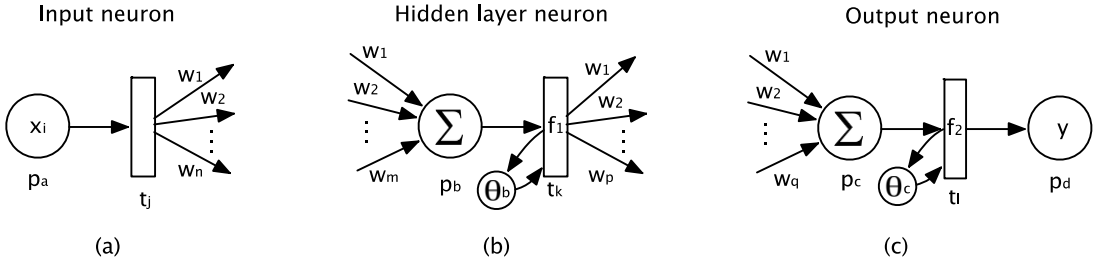

Figure 2. PN structures to represent neurons in an NPN.

An NPN has a vector X of input values in the input layer, one or more hidden layers, and the output layer indicating the result of the NPN execution (Figure 3).

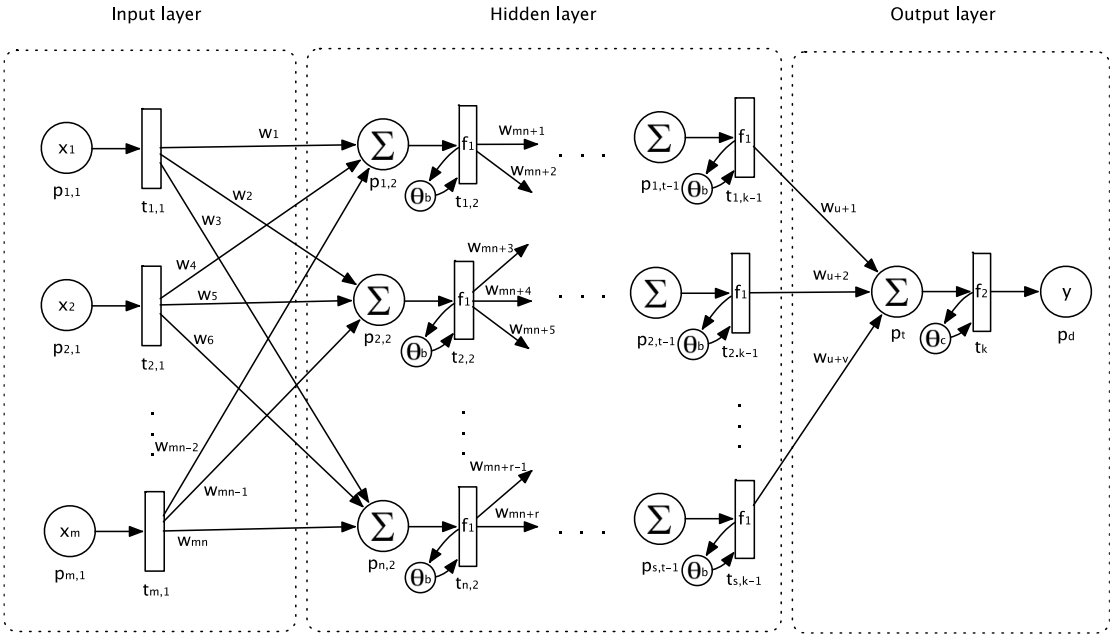

Figure 3. NPN structure with input, hidden and output layers.

Most of the connections are located where transitions connect to their corresponding output places. These connections can be represented as submatrices of the whole NPN incidence matrix. The first submatrix IM_f of $m \times n$ elements contain the weights from the input layer to the first hidden layer (Eq. 3)

$$IM_f = \begin{array}{c} \\ t_{1,1} \\ t_{2,1} \\ \vdots \\ t_{m.2} \end{array} \begin{array}{c} \begin{array}{cccc} p_{1,2} & p_{2,2} & \cdots & p_{n,2} \end{array} \\ \begin{bmatrix} w_1 & w_2 & \dots & w_3 \\ w_4 & w_5 & \dots & w_6 \\ \vdots & \vdots & \vdots & \vdots \\ w_{mn-2} & w_{mn-1} & \dots & w_{mn} \end{bmatrix} \end{array} \tag{3}$$

And the last submatrix IM_l is a column vector of $s \times 1$ elements, and it contains the weights from the last hidden layer to the output layer (Eq. 4).

$$IM_l = \begin{array}{r} \\ t_{1,k-1} \\ t_{2,k-1} \\ \vdots \\ t_{s.k-1} \end{array} \begin{array}{c} \begin{array}{c} p_t \end{array} \\ \begin{bmatrix} w_{u+1} \\ w_{u+2} \\ \vdots \\ w_{u+v} \end{bmatrix} \end{array} \tag{4}$$

These matrices are used to train the NPN with a set of input data. The next section describes the learning process.

4.2. Learning algorithm

Backpropagation (BP) is the algorithm selected for training the NPN because it has been used widely in ANN applications (Sacramento et al., 2018). The steps of the BP algorithm adapted to NPN are shown below. It takes a set of patterns μ used as inputs for the training.

1. Set randomly the weights of all arcs connecting from transitions to places of different layers and the initial *bias*.
2. For each pattern μ_i of the learning set
   a. Execute the NPN with pattern μ_i.
   b. Obtain the associated error signals.
   c. Calculate the partial increase for weights and bias due to μ_i.
3. Calculate the total increase.
4. Update weights in submatrices and *bias*.
5. Calculate the total error with Mean Square Error (MSE). Go to step 2 if the error is unsatisfactory; otherwise, the algorithm finishes.

In order to obtain the result of the NPN execution, the next subsection describes the matrix operations required.

4.3. Execution of NPN

The execution of the NPN produces the evaluation of all the input values across the hidden layers and the output layer. For this purpose, the state equation and

incidence matrix concepts of PN theory are used.

In this case, the vector containing the bias values is used as the previous marking. The matrix IM is the sub incidence matrix composed of arcs connecting from transitions to the next layer places. And as the firing vector we use M_{k-1}, which contains the initial tokens of X, or the token value that arrives at each place in the following calculations. Finally, the result is evaluated with the activation function f_1 or f_2, depending on the type of layer. Therefore, the adapted state equation is shown in equation 5.

$$M_k = f_{1|2}(-\theta + IM^T \times M_{k-1}) \tag{5}$$

This equation is utilized to calculate the token values for places in NPN.

For instance, to calculate the token values for places $p_{1,2}$, $p_{2,2}$, ..., $p_{n,2}$ in Figure 3, we use equation 5 as follows:

$$M_1 = f_1\left(-\theta + \begin{bmatrix} w_1 & w_2 & \cdots & w_3 \\ w_4 & w_5 & \cdots & w_6 \\ \vdots & \vdots & \vdots & \vdots \\ w_{mn-2} & w_{mn-1} & \cdots & w_{mn} \end{bmatrix}^T \times \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}\right) \tag{6}$$

And so on, until we calculate the token value for place p_d (Figure 3) denoting the evaluation of the whole NPN.

4.4. Results

In order to observe the NPN applicability in biotechnological processes, a data set of lab experiments were collected. The experiment consists of a full factorial composite design to produce lignosulfonates, considering sulfite concentration (1, 5, and 10%), particle size (mesh number 8, 12, and 20), and reaction time (30, 60, and 90 minutes). 27 experiments were carried out three times each one, and the lignocellulosic biomass obtained was recorded.

The NPN can be created with the constructors showed in figure 2. A NPN for modeling this process containing 3 neurons in the input layer (sulfite concentration, particle size, and reaction time), one hidden layer composed of 3 neurons, and one neuron in the output layer (amount of lignocellulosic biomass) is shown in figure 4.

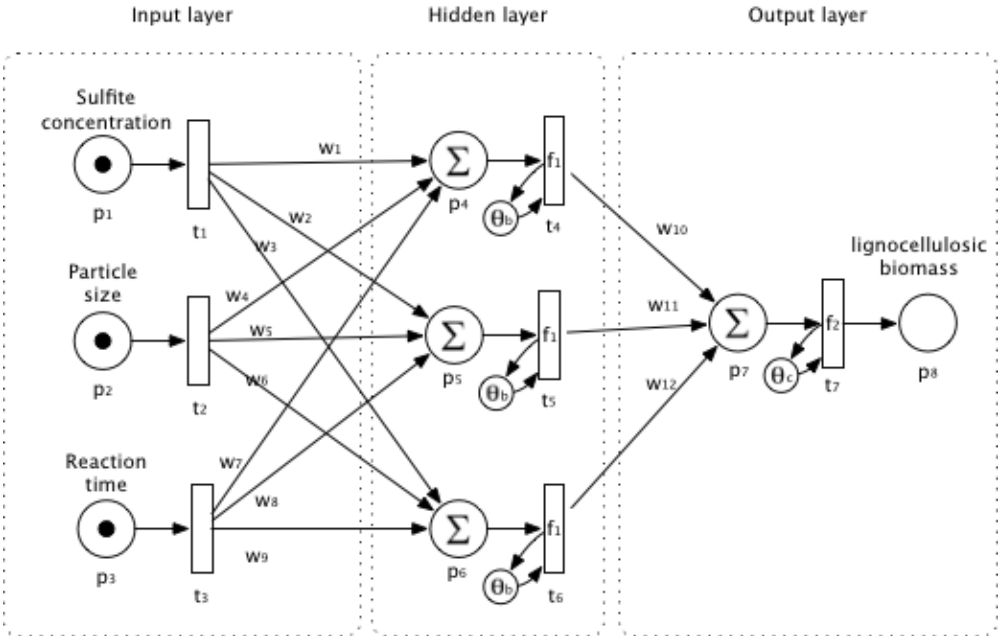

Figure 4. NPN structure for data collected in a biotechnological process with 3 input variables and one response.

The goodness-of-fit between the experimental and predicted values obtained from the NPN has a good correlation coefficient (R^2 = 0.91).

The NPN has a good performance due to predictions of lignocellulosic biomass were closer to experimental values.

5. Conclusions

ANNs and PNs present interesting similarities that can be considered to improve the learning capabilities of ANNs. There are papers where the authors combine PNs together with ANNs, but they do not apply the mathematical strength of PNs in the ANN learning process.

In this work, an Extended PN model with learning capabilities is proposed. This model, named Neural Petri Net (NPN), provides basic PN structures to represent neurons for the input layer, hidden layers, and the output layer. Furthermore, this model includes a learning algorithm based on the well-known backpropagation algorithm, and an execution algorithm to evaluate input values with the trained NPN.

The incidence matrix and state equation are used in the training and execution of the NPN, which means that the obtained model is a fusion of the features of PNs and ANNs. In addition, the performance of the NPN for predicting values of a biotechnological process was suitable. Nevertheless, this methodology is not only applicable to biotechnology, it can also be applied to data of another fields for generating models with input and response variables.

The main limitation of this proposal is inherited from PNs. There are needed two times the number of nodes in an ANN, thus it requires to handle bigger structures to represent the model.

As further work, the NPN will be applied to model another biotechnological processes such as conidiospores production and microplastics detection. In engineering applications, this proposal will be used to model the making of bricks to optimize its performance, in combination with evolutive algorithms.

Funding

This work was supported by the Consejo Nacional de Humanidades, Ciencia y Tecnología (Conahcyt), under Project CONACYT CB-2017-2018-A1-S-43008

References

Aggarwal, C. C. (2018). Neural networks and deep learning. Springer, 10(978), 3.

Amstein, L. K., Ackermann, J., Hannig, J., Đikić, I., Fulda, S., & Koch, I. (2021). Mathematical modeling of the molecular switch of TNFR1-mediated signaling pathways using Petri nets. BioRxiv, 2021-11.

Bardini, R., Benso, A., Politano, G., & Di Carlo, S. (2021). Nets-within-nets for modeling emergent patterns in ontogenetic processes. Computational and Structural Biotechnology Journal, 19, 5701-5721.

Chen, M., Challita, U., Saad, W., Yin, C., & Debbah, M. (2019). Artificial neural networks-based machine learning for wireless networks: A tutorial. IEEE Communications Surveys & Tutorials, 21(4), 3039-3071.

Kardynska, M., Kogut, D., Pacholczyk, M., & Smieja, J. (2023). Mathematical modeling of regulatory networks of intracellular processes–Aims and selected methods. Computational and Structural Biotechnology Journal.

Klyuchko, O. M. (2017). On the mathematical methods in biology and medicine. Biotechnologia Acta, 10(3), 31-40.

Klyuchko, O. M. (2018). Some trends in mathematical modeling for biotechnology. Biotechnologia Acta, 11(1), 39-57.

Niazian, M., & Niedbała, G. (2020). Machine learning for plant breeding and biotechnology. Agriculture, 10(10), 436.

Noll, P., & Henkel, M. (2020). History and evolution of modeling in biotechnology: modeling & simulation, application and hardware performance. Computational and Structural Biotechnology Journal, 18, 3309-3323.

Pham, D. T., & Parhi, D. R. (2003). Navigation of multiple mobile robots using a neural network and a Petri Net model. Robotica, 21(1), 79-93.

Sacramento, J., Ponte Costa, R., Bengio, Y., & Senn, W. (2018). Dendritic cortical microcircuits approximate the backpropagation algorithm. Advances in neural information processing systems, 31.

Schuster, A. J. (2007). A Computing Model Combining Artificial Neural Networks and Petri Nets. Intelligent Computing Everywhere, 61-77.

Simone, P., Laura, F., Gianfranco, B., Luciano, M., Giulia, S., Niccoló, T., ... & Marco, B. (2020). Integrating petri nets and flux balance methods in computational biology models: a methodological and computational practice. Fundamenta Informaticae, 171(1-4), 367-392.

Tan, K. H. (2015). Squirrel-cage induction generator system using wavelet petri fuzzy neural network control for wind power applications. IEEE Transactions on Power Electronics, 31(7), 5242-5254.

van Gerven, M. A. J., & Bohte, S. M. (2017). Artificial neural networks as models of neural information processing.

Wai, R. J., & Lin, Y. W. (2012). Adaptive moving-target tracking control of a vision-based mobile robot via a dynamic petri recurrent fuzzy neural network. IEEE Transactions on Fuzzy Systems, 21(4), 688-701.

Wen, J., Yang, J., Jiang, B., Song, H., & Wang, H. (2020). Big data driven marine environment information forecasting: a time series prediction network. IEEE Transactions on Fuzzy Systems, 29(1), 4-18.