High Energy Hadronic Collisions Using Neural Network and Genetic Programming Techniques

Moaaz A. Moussa

***Abstract*—Artificial Intelligence (AI) techniques of artificial neural networks (ANN) and evolutionary computation of genetic programming (GP) have recently been used to design and implement more effective models. The artificial neural network (ANN) model has been used to study the charged particles multiplicity distributions for antiproton-neutron ($p^- - n$) and proton-neutron ($p - n$) collisions at different lab momenta. The neural network model performance was also tested at non-trained space (predicted) and matched them effectively. The trained NN shows a good fitting with the available experimental data. The NN simulation results prove a solid existence in modeling hadronic collisions. Genetic Programming (GP) model is a flexible and powerful technique that can be used for solving the same problem. In this paper, genetic programming (GP) has been used to discover a function that calculates the charged particles multiplicity distribution of created pions for the same interactions at high energies. The predicted distributions from the GP-based model are compared with the available experimental data. The discovered function of GP model has proven an excellent matching with the corresponding experimental data.**

***Index Terms*—Artificial intelligence technique, genetic programming, hadronic collisions, machine learning (ML), multiplicity distribution, neural network, pion production.**

I. Introduction

The theories and ideas concerning multiparticle production go back to the late of 1930's with a significant interlude at Fermi's statistical theory of particle production [1]. One of the basic interactions in high-energy physics (HEP) is the antiproton-neutron ($p^- - n$) and proton-neutron ($p - n$) interactions particularly above the pion production threshold (1 GeV approx.). Extremely high energy collisions are required to get the fundamental particles close enough to study and understand the interactions between them [2]–[7]. Different models are provided for the hadron structure [8]–[11], such as the three-fireball model [12], fragmentation model [13]–[15] quark models [16]–[18], and many others.

The application of artificial intelligence (or the machine learning) such as genetic programming (GP) and neural network (NN) has a strong presence in the high energy physics [19]–[23]. The effort to understand the interactions of fundamental particles requires complex data analysis for which machine learning (ML) algorithms are essential. Machine learning (ML) algorithms are becoming more useful as alternate approaches to conventional techniques [24].

Parallel to the theoretical approach based on different views, development in the artificial intelligence (AI) and evolutionary computation field have given the neural networks and genetic programming a strong presence in high-energy physics [25]-[27]. Neural networks are composed of simple interconnected computational elements operating in parallel. These artificial neural networks (ANNs) are trained, so that a particular input leads to a specific target output.

The complicated behavior of many interactions due to the nonlinear relationship between the interaction parameters and the output often becomes more complicated. In this sense, ML techniques such as artificial neural network [28], genetic algorithm [29] and genetic programming [30] can be used as alternative tool for the simulation of these interactions [18]–[22], [31]–[36].

The motivation of using a GP approach is its ability to evolve a model based entirely on prior data without the need of making underlying assumptions. Another motivation for applying such machine learning approach (e.g. GP) is simply the lack of knowledge (in most cases) about the mathematical dependence of the quantity of interest on the relevant measured variables [37].

In the present work, we illustrate the NN and GP techniques to model the multiplicity distribution of charged pions for different beams at different high energies in hadronic collisions. The history of studies of these interactions is therefore very long and extremely interesting from both the experimental and theoretical view point [38, 39]. Making use of the capability of the artificial intelligence and the evolutionary computation, the present work uses the NN and GP to model the charged particles multiplicity distribution for ($p^- - n$) and ($p - n$) interactions at different lab momenta. Also, GP has been used to discover a function that calculates the multiplicity distribution for different beams. The rest of the paper is organized as follows; The NN model is described in Sections II, III. Section VI gives a review to the basics of the GP technique. Section V explains how genetic programing is used on modeling the hadron-hadron collisions. The results and discussions of both models are explained in Section VI.

II. Artificial Neural Networks (ANNs)

An ANN is made up of a number of simple and highly interconnected computational elements. There are many types of ANNs, but all of them have three things in common:

Manuscript received September 24, 2012; revised January 14, 2013. This work was supported and financed by the Buraydah Colleges under a Grant from Dr. Abdullah Bin Saleh El-Shetaiwy.

Moaaz A. Moussa is with the Buraydah Colleges, Al-Qassim, Buraydah, King Abdulazziz Road, East Qassim University, P.O.Box 31717, Kingdom of Saudi Arabia (e-mail: moaaz2030@yahoo.com).

individual neurons (processing elements), connections (topology), and a learning algorithm. The processing element calculates the neuron transfer function of the summation of weighted inputs. A simple neuron structure is shown in the Fig. 1. The neuron transfer function, f is typically step or sigmoid function that produces a scalar output (n) as in Eq. (1).

$$n = f\sum_i w_i I_i + b \tag{1}$$

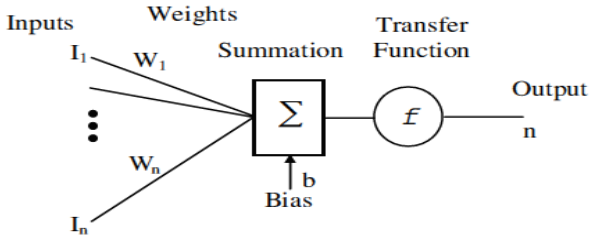


Fig. 1. Neuron model.

where, I_i , w_i , b are the i th input, the i th weight and b the bias respectively.

A network consists of one or more layers of neurons. A layer of neurons is a number of parallel neurons. These layers are configured in a highly interconnected topology.

III. Training of the H-H-Ann

Neural network can be trained to perform a particular function by adjusting the values of the connections (weights) between elements. Training in simple is to make a particular input leads to a specific target output. The weights are adjusted, based on a comparison of the output and the target, until the network output matches the target. Typically many such input/target pairs are used, in this supervised learning, to train a network.

The proposed ANNs in this paper was trained using the Levenberg–Marquardt optimization technique. This optimization technique is more powerful and flexible than the conventional gradient descent techniques [40]-[44]. The Levenberg–Marquardt updates the network weights using the following rule,

$$\Delta W = (J^T J + \mu I)^{-1} J^T e$$

where, J is the Jacobian matrix of derivatives of each error with respect to each weight, μ is a scalar, changed adaptively by the algorithm and e is an error vector.

The only requirement for this method is a considerably large memory for large problems. The initial training weights were also chosen using the Nguyen–Widrow random generator in order to speed up the training process [40]-[44].

IV. Genetic Programming Overview

Genetic programming is an extension to Genetic Algorithms (GA). GA is an optimization and search technique based on the principles of genetics and natural selection. A GA allows a population composed of many individuals (chromosome) to evolve under specified selection rules to a state that maximizes the "fitness" (i.e. minimizes the cost function). The GP is similar to genetic algorithms but unlike the latter its solution is a computer program or an equation as against a set of numbers in the GA. A good explanation of various concepts related to GP can be found in Koza (1992) [30], [45].

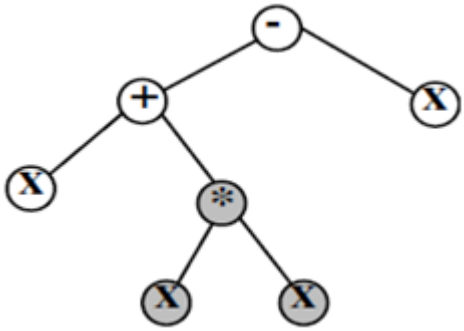


Fig. 2. Tree representation of the equation $(x + x^2 - x)$.

In GP a random population of individuals (equations or computer programs) is created, the fitness of individuals is evaluated and then the 'parents' are selected out of these individuals. The parents are then made to yield 'offspring's' by following the process of reproduction, mutation and crossover. The creation of offspring's continues (in an iterative manner) until a specified number of offspring's in a generation are produced and further until another specified number of generations are created. The resulting offspring at the end of all this process is the solution of the problem. The GP thus transforms one population of individuals into another one in an iterative manner by following the natural genetic operations like reproduction, mutation and crossover. Each chromosome (individual) contributes with its own genetic information to the building of new ones (offspring) adapted to the environment with higher chances of surviving. This is the basis of genetic algorithms and programming [46]. The representation of a solution for the problem provided by the GP algorithm is a tree (Fig. 2).

V. Genetic Programming Technique

Genetic programing is a technique that mimics natural evolution and improvement of life through reproduction to find a computer program that solves a particular task. It is inspired by the Darwinian principle "the most fit chromosome duals have the greatest chance of surviving and passing into the next generation" [47].

Genetic programing searches the space of computer programs, or the space of functional forms specified by compositions of functions from a function set acting on terminals from the terminal set. The chromosome represents the model of the problem solution using trees. A tree is a model representation that contains nodes and leaves. Nodes are mathematical operators from the specified function set. Leaves are terminals from the specified terminal set [46]. Table I shows some typical functions and terminals used in GP.

TABLE I: TYPICAL FUNCTIONS AND TERMINALS USED IN GP.

Functions	Terminals
Mathematical	Sin, cos, exp, log
Variables	X, y
Looping	For, Repeat
Boolean	AND, OR, NOT
Random constant	Random
Conditional	IF, THEN-ELSE
0-arity functions	Rand, go-left
Arithmetic	+, -, *, /
Constant values	3, 0.45

Trees are manipulated through the basic genetic operators: crossover (sexual recombination operation), mutation (asexual operation), and reproduction.

Crossover (Sexual Recombination) Operation: In the crossover or sexual recombination operation, two parental programs are probabilistically selected from the population based on fitness. The two parents participating in crossover are usually of different sizes and shapes. A crossover point is randomly chosen in the first parent and a crossover point is randomly chosen in the second parent. Then the sub-tree rooted at the crossover point of the first, or receiving, parent is deleted and replaced by the sub-tree from the second, or contributing, parent. Crossover is the predominant operation in genetic programing (and genetic algorithm) work and is performed with a high probability.

Mutation Operation: In the mutation operation, a single parental program is probabilistically selected from the population based on fitness. A mutation point is randomly chosen, the sub-tree rooted at that point is deleted, and a new sub-tree is grown there using the same random growth process that was used to generate the initial population. This asexual mutation operation is typically performed sparingly (with a low probability of, say, 1% during each generation of the run).

Reproduction Operation: The reproduction operation copies a single chromosome, probabilistically.

In order to apply the genetic programing technique to a problem, one must first perform the preparatory steps and the executional steps [48]. The preparatory steps are the problem-specific and domain-specific steps that are performed by the human user prior to launching a run of the problem-solving method. The executional steps are automatically executed during a run of the problem-solving method.

The five major preparatory steps for the basic version of genetic programming require a human user to specify:

1) the set of terminals,
2) the set of primitive functions,
3) the fitness measure,
4) certain GP parameters (see Table II) for controlling the run, and
5) a termination criterion and method for designating the result of the run.

The fitness function defines the quality of chromosome as a solution to the problem. The dataset is divided into two parts: one is for training and the second for validation. The training dataset is used to obtain the model and the validation dataset is used to measure the accuracy of the model with data that was not used in training. The fitness function evaluates how accurate the mathematical model.

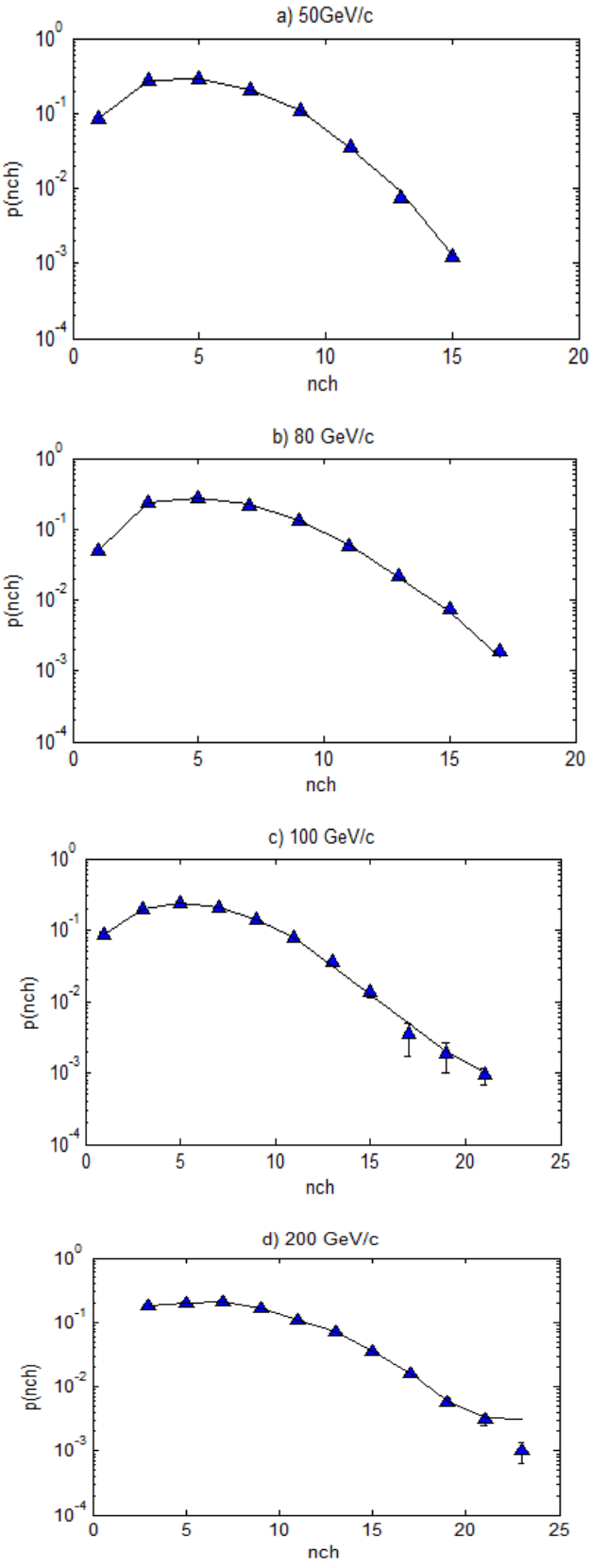


Fig. 3. Comparison between the experimental and simulated multiplicity distribution of pions $P(n_{ch})$ for $p^- - n$ collisions at a) 50, b) 80 GeV/c and for $p-n$ collisions at, c) 100, d) 200 GeV/c: (—) NN model, (▲) experimental data.

TABLE II: DEFINITION OF THE GP PARAMETERS.

GP Parameters	Definitions
Populations size	The number of chromosomes in a population generation size; The number of iterations of the main selection/operation loop
Maximum tree size	The maximum depth of an expression tree (this is necessary since crossover tend to increase the average size of a population, and this inadvertently increase the run time of each generation).
Mutation rate	How often mutation occur
Crossover rate	How often crossover occur
Reproduction rate	How often reproduction occur

VI. RESULTS AND DISCUSSION

The NN and GP models are implemented using the experimental data to simulate the multiplicity distributions of charged pions $P(n_{ch})$ at $P_L = 50,80$ GeV/c for $p^- - n$ and $P_L = 100,200,400$ GeV/c for $p-n$ collisions. The results of these calculations are represented in Fig. 3, 4, 5, and 6 along with the experimental data [49]-[52] which show good agreement with the corresponding experimental data.

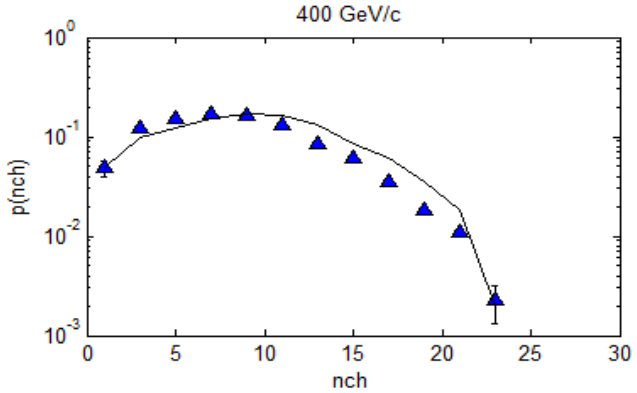


Fig. 4. Comparison between the experimental and predicted multiplicity distribution of pions $P(n_{ch})$ for $p-n$ collisions at 400 GeV/c: (—) NN model, (▲) experimental data

Using the input-output arrangement, different networkconfigurations were tried to achieve good mean squared error (MSE) and good performance for the network. It consists of an input layer (P_{Lab}, n_{ch}), one hidden layers of 10 neurons, respectively, and an output layer consisting of one neuron $P(n_{ch})$. The transfer function where chosen to be a tan sigmoid function for the hidden layer and a pure line function for the output layer, the trained NN model shows almost exact fitting. It is worth mentioning that the NN training data did not include the experimental data at $P_L = 400$ GeV/c. This means that the NN model not only simulated the trained (Fig. 3) observations but also predicted the multiplicity distribution of charged pions for untrained observations as shown in Fig. 4. Then, the ANN technique is able to exactly model for multiplicity distribution at lab momenta for different beams in hadrons collisions.

The GP model was constructed with training sets and the accuracy was verified by the test sets. In order to generate the GP model, we have implemented the GP steps (Fitness evaluation, reproduction, crossover and mutation) that were mentioned in Section V. Table III lists the values of the control parameters and the set of function genes that are used in modeling the multiplicity distribution. Our discovered function is generated using the obtained control GP parameters as follows,

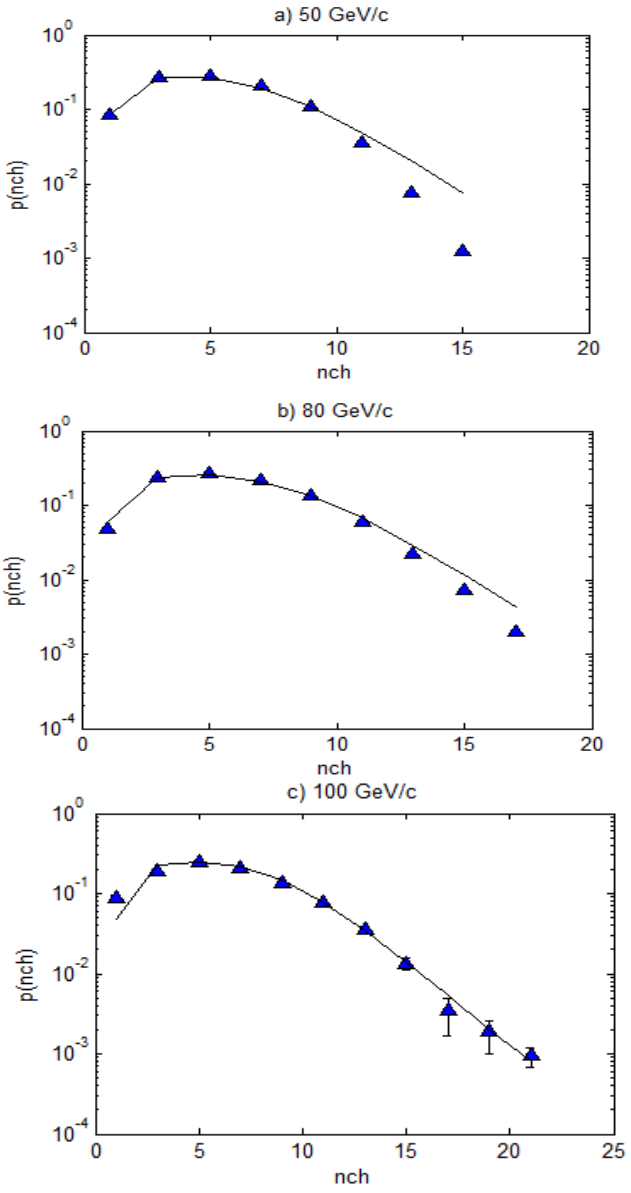
$$P(n_{ch}) = (\text{sqrt}(\log((X + (5/Y))))/((\text{sqrt}(\exp(X))/((Y/7) + (3/2)) + ((\text{sqrt}(Y)/(1+2)) + ((2/5) + (2/3))))),$$

where the actual parameters are,

X = number of charged pions (n_{ch}), Y = lab momentum (P_L). After simplification and putting the corresponding values, the final form of the discovered equation becomes,

$$P(n_{ch}) = (\text{sqrt}(\log((n_{ch} + (5/P_L))))/((\text{sqrt}(\exp(n_{ch}))/((P_L/7) + (3/2))) + ((\text{sqrt}(P_L)/(1+2)) + ((2/5) + (2/3)))))$$

This discovered function has been used to predict the multiplicity distribution of pions for antiproton-neutron $p^- - n$ and proton-neutron $p-n$ interactions.



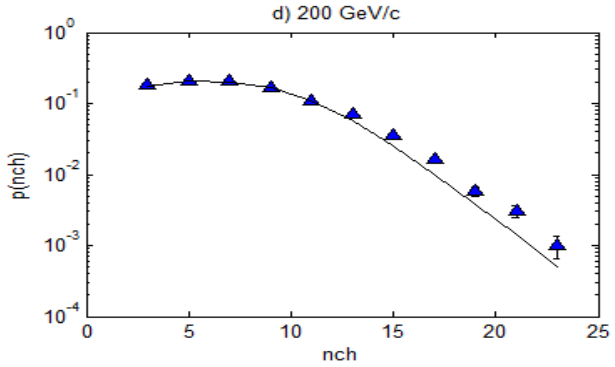


Fig. 5. Comparison between the experimental and simulated multiplicity distribution of pions $P(n_{ch})$ for $p^- - n$ and $p - n$ collisions at a) 50, b) 80, c) 100, d) 200 GeV/c: (—) GP model, (▲) experimental data

TABLE III: LISTS THE VALUES OF THE CONTROL PARAMETERSUSED IN MULTIPLICITY DISTRIBUTION

GP Parameters	Values
Generations	1000
Populations	40000
Function set	*, /, -, +, log, sqrt, sin, cos
Terminal Set	{constant, X,Y}
Fitness function	SSE
Selection method	Elites, rank and roulette
Mutation rate	0.01
Crossover rate	0.9

Simulation results based on GP model, for modeling the multiplicity distribution of pions for antiproton-neutron ($p^- - n$) and proton-neutron ($p - n$) interactions at $P_L = 50,80$ GeV/c for $p^- - n$ and $P_L = 100,200$ GeV/c for $p - n$ (the training cases) are given in Fig. 5 (a), (b), (c), (d) respectively. While Fig. 6 describes the predicted results of $P_L = 400$ GeV/c for $p - n$ interaction, we notice that the curves (for training cases and prediction case) obtained by the trained GP model show a best fitting to the experimental data in the five cases. Then, the GP model is able to exactly model for multiplicity distribution at different lab momenta for different beams in h-h collisions. If the large dataset is used in training, the best GP model is obtained.

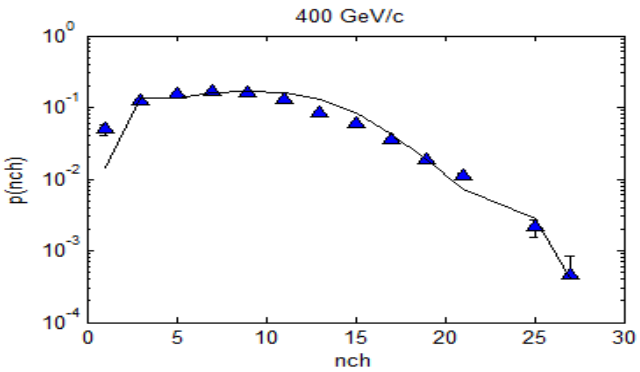


Fig. 6. Comparison between the experimental and predicted multiplicity distribution of pions $P(n_{ch})$ for $p - n$ collision at 400 GeV/c: (—) GP model, (▲) experimental data

ACKNOWLEDGMENT

Moaaz A. Moussa is grateful to all Buraydah Colleges Staff Members. Especially, physics Lab. colleagues and deeply gratitude to Dr. Abdullah Bin Saleh El-Shetaiwy for his continuous advice and encouragement. This work was supported and financed by a grant from Buraydah Colleges.

REFERENCES

[1] E. Fermi, *Prog. Theor. Phys*., vol. 5, pp. 568, 1950.
[2] ALICE Collaboration *et al*., "ALICE: Physics Performance Report, Volume II," *J. Phys. G: Nucl. Part. Phys*., doi:10.1088/0954-3899/32/10/001, 2006.
[3] F. Carminati *et al*., *J. Phys. G: Nucl. Part. Phys*., vol. 30, pp. 1517, 2004.
[4] D. G. d'Enterria*et al*., *J. Phys. G: Nucl. Part. Phys.*, vol. 34, pp. 2307, 2007.
[5] G. L. Bayatian*et al*., *J. Phys. G: Nucl. Part. Phys*., vol. 34, pp. 995, 2007.
[6] G. Aad *et al*., arXiv:0901.0512v4 [hep-ex]
[7] A. A. Alves *et al*., *J. Instrum*. 3, S08005, 2008.
[8] M. Tantawy, M. El Mashad, and M. Y. Elbakry, *Indian J. Phys*. Pt-A 72, vol. 110, 1998.
[9] E. Fermi, *Prog. Theor. Phys*. 5, vol. 570, 1951.
[10] E. Fermi, *Phys. Rev*. 81, vol. 683, 1950.
[11] J. Ranft, *Phys. Lett*. B 31, vol. 529, 1970.
[12] C. Xu, C. Wei-Qin, and M. Ta-chung, *Phys. Rev*. D 33, vol. 1287, 1986.
[13] Y. Nambu, *Sci. Am*. 235, 45, 1976.
[14] M. Gyulassy, *Prog. Part. Nucl. Phys*., vol. 15, 403, 1985.
[15] L. S. Kisslinger,*Nucl. Phys*. A 446, vol. 479, 1985.
[16] M. Jacob, R. Slansky, *Phys. Rev. D* 5, vol. 1847, 1972.
[17] R. Hwa, *Phys. Rev. D* 1, vol. 1790, 1970.
[18] R. Hwa, *Phys. Rev. Lett*., vol. 26, pp. 1143, 1971.
[19] L. Teodorescu and D. Sherwood, *Comput. Phys. Commun*., vol. 178, pp. 409, 2008
[20] L. Teodorescu, *IEEE T. Nucl. Sci*., vol. 53, pp. 2221, 2006.
[21] M. A. Moussa, M. Y. El-Bakry, A. Radi, E.-S. A. El-Dahshan, and M. Tantawy, "Artificial Neural Network Modeling in Hadrons Collisions," *International Journal of Scientific and Engineering Research*, vol. 3, issue 8, August, 2012.
[22] M. Y. El-Bakry, M. A. Moussa, A. Radi, E.-S. A. El-Dahshan, and M. Tantawy, "Genetic Programming Model for Hadronic Collisions," *International Journal of Scientific and Engineering Research*, vol. 3, issue 8, August, 2012.
[23] S. Whiteson and D. Whiteson, *Eng. Appl. Artif. Intel*., vol. 22, pp. 1203, 2009.
[24] S. Haykin, *Neural Networks: A Comprehensive Foundation*, (IEEE Press and Macmillan College Publishing Company, New York: NY, 1994.
[25] A. K. Hamid, *Can. J. Phys*., vol. 7, no. 76, 1998.
[26] P. Bhat, "Using neural networks to identify jets in hadron–hadron collisions," in *Proc. The 1990 Summer Study on HEP*, 1990.
[27] R. P. Lippman, *IEEE Acoust Speech Signal Process Mag*, 1987.
[28] J. H. Holland, *Adaptation in Natural and Artificial Systems*, Ann Arbor: University of Michigan Press, 1975.
[29] J. R. Koza, *Genetic Programming: On the Programming of Computers by means of Natural Selection*, The MIT Press, Cambridge, MA, 1992.
[30] K. Cranmer and R. S. Bowman, *Comput. Phys. Commun*, vol. 167, pp. 165, 2005.
[31] M. Y. El-Bakry and A. Radi, *Int. J. Mod. Phys. C*, vol. 18, pp. 329, 2007.
[32] M. Y. El-Bakry, K. A. El-Metwally, *Chaos Soliton. Fract*., vol. 16, pp. 279, 2003.
[33] K. A. El-Metwally, T. I. Haweel, and M. Y. El-Bakry, *Int. J. Mod. Phys. C*, vol. 11, pp. 619, 2000.
[34] E. El-dahshan, A. Radi, and M. Y. El-Bakry, *Int. J. Mod. Phys. C*, vol. 19, pp. 1787, 2008.
[35] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Algorithms*, Berlin: Springer, 2003.
[36] H. Etemadi, A. A. A. Rostamy, and H. F. Dehkordi, *Expert Syst. Appl*., vol. 36, pp. 3199, 2009.
[37] E. El-Dahshan, A. Radi, M. Y. El-Bakry, and M. El Mashad, "Artificial neural network in heavy ion collision," in *Proc. 6th Nuclear and Particles Physics Conference*, Luxor, Egypt, November, 2007, pp. 17-21.
[38] M. Y. El-Bakry, *Chaos, Solitons and Fractals*, vol. 18, pp. 995, 2003.

[39] M. T. Hagan and M. B. Menhaj, *IEEE Trans Neural Networks* vol. 6, pp. 861–867, 1994.
[40] E.-S. El-dahshan and A. Radi, *Cent. Eur. J. Phys*., vol. 9, no. 3, pp. 874-883, 2011.
[41] S. Y. El-Bakry, E.-S. El-dahshan, and M. Y. El-Bakry, *Ind. J. Phys*., vol. 85, no. 9, pp. 1405-1413, 2011.
[42] S. Y. El-Bakry, E.-S. El-dahshan, S. Al-Awfi, and M. Y. El-Bakry, *ILNouvo Cimento*, vol. 125B, no. 10, 2010.
[43] E.-S. El-dahshan, A. Radi, and M. Y. El-Bakry, *Int. J. Mod. Phys. C*, vol. 20, no. 11, pp. 1817-825, 2009.
[44] M. Wolter, *Phys. Part. Nucl*., vol. 38, pp. 255, 2007.
[45] J. C. Werner. High Purity Neutral Pions Selection Using Genetic Programing Discriminate Function. University of Manchester Internal Report for Research Project. [Online]. Available: http://www.hep.manchester. ac.uk/u/jamwer/.
[46] T. Malisiewicz, *Genetic programing and non-linear regression*, vol. 2 December, 2003.
[47] S. Sette and L. Boullart, *Eng. Appl. Artificial Intelligence*, vol. 14, pp. 727, 2001.
[48] J. E. A. Lys, C. T. Murphy, and M. Binkley, *Phys. Rev. D*, vol. 16, pp. 3127-3136, 1977.
[49] T. Dombeck, L. G. Hyman, *et al.*, *Phys. Rev. D*, vol. 18, pp. 86-91, 1978.
[50] S. Dado, S. J. Barish, A. Engler, and R. W. Kraemer, *Phys. Rev., D*, vol. 20, issue 7, 1979,
[51] D. K. Bhattacharjee, *Phys. Rev., D*, vol. 41, pp. 9, 1990.
[52] Fermilab Proposal No. 422 Scientific Spokesman: A. Fridman Centre De Recherches Nucleaires de Strasbourg Groupe des Chambres a Bulles a Hydrogene, France, 1975.

Moaaz A. Moussa was born in Tara Niece El-Bahr Village, El-Mansura City, El-Dakahliya Governorate, Egypt, on August 5, 1978. He held BSc in Physics and Chemistry Department, from the Faculty of Education, Al-Azhar University, Cairo, Egypt. His MSc degree was from the Physics Department, Faculty of Education, Ain Shams University, Cairo, Egypt. Currently, he is pursuing his study in PhD level at Ain Shams University in the same field which his study focuses on applications of Artificial Intelligence Technique (AI) and evolutionary computation in High Energy Physics (HEP). Presently, he is a Senior Lecturer at Buraydah Colleges, Al-Qassim, Buraydah, King Abdulazziz Road, East Qassim University, P.O.Box 31717, Kingdom of Saudi Arabia.