ISSN 2278-3091
Volume 9, No.4, July – August 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse77942020.pdf
https://doi.org/10.30534/ijatcse/2020/77942020

Applying Hopfield Neural Networks To Solve CSP Problems

Anatolii Balanda[1], Mykola Pohoretskyi[2], Diana Serhieieva[3], Mikhail Hribov[4], Zoriana Toporetska[5]
[1]The Special Department Military Diplomatic Academy named after Yevheniy Bereznyak, Kyiv, Ukraine
[2]Department of Justice, Taras Shevchenko National University of Kyiv, Kyiv, Ukraine
[3]Department of Justice, Taras Shevchenko National University of Kyiv, Kyiv, Ukraine
[4]Department of operational investigative activities, National Academy of Internal Affairs of Kyiv, Kyiv, Ukraine
[5]Department of Justice, Taras Shevchenko National University of Kyiv, Kyiv, Ukraine

ABSTRACT

The article reviews methods based on the Hopfield neural network for solving CSP and FCSP problems. The first attempt to apply this type of neural network to solving the CSP problem was made by Hopfield himself, after which a number of modifications of the original algorithm took place. That is, all the methods presented in the article are modifications of each other and have developed consistently. Some characteristics of Hopfield network-based methods in comparison with other (non-neural network-based) algorithms and CSP solutions are also given.
In the field of artificial intelligence, there is a class of combinatorial problems called CSP problems (Constraint satisfaction Problems). They are a powerful tool for solving practical problems that can be designed for many variables that are bound together by constraints.

Key words : CSP, constraint satisfaction problem, neural network, Hopfield neural network.

1. INTRODUCTION

At the same time, the classic CSP model is quite primitive in order to be the prototype of most real-life AC applications. Sometimes it is difficult to formulate constraints in real problems in a rigid form, that is, in such a way that it is possible to determine unambiguously whether this restriction is sufficient or not. Another way in which such problems can occur is if it is acceptable to partially satisfy the restrictions. Moreover, in some situations, it is not possible to fully satisfy all the constraints of the task [1-4].

All these factors caused the appearance of a variety of CSP tasks, which became known as FCSP (Fuzzy Constraint Satisfaction Problem). In contrast to CSP tasks, FCSP tasks operate with fuzzy logic and flexible constraints, which allows for incomplete satisfaction of constraints in the task [5].

Today, there is a large Arsenal of methods for solving CSP problems [1], while the number of algorithms for solving FCSP problems is quite small. Basically, FCSP is solved by raising it to the CSP of a task or a number of CSP tasks and then solving them. However, there are other methods, in particular, algorithms for solving FCSP problems based on neural networks. Traditionally, neural networks are used to solve problems of prediction, classification and pattern recognition, but history shows that there have also been numerous attempts to apply them to solving combinatorial optimization problems. Many of these attempts were successful, and the effectiveness of the constructed methods was comparable to alternative approaches [10]. However, despite this, it is generally considered that neural networks "are not very successful when applied to solve optimization problems and they do not go into any comparison with the best metaeuristics", such as the annealing method, taboo search, and genetic algorithms [15]. This article is intended to rehabilitate the authority of neural networks, at least in the field of CSP problem solving.

Historically, only neural networks with training without a teacher have been used to solve CSP tasks. They are best suited for this purpose, since they do not require generating training examples [6-9]. This can easily be explained by a simple example: if we assume that the input of a neural network under control is a solution to the CSP problem, then a "good" training example for the network would have to be a valid CSP solution, hence we would already have denouements, which is obviously absurd. However, this does not mean that neural networks with teacher training are not applicable to CSP tasks, just that the CSP task must be correctly designed for a neural network.

The most popular neural network for solving CSP problems was the Hopfield network, which will be discussed in the article.

2. MATERIALS AND METHODS

2.1 CSP issue

A Constraint Satisfaction Problem is a problem consisting of N variables, a set of domains $A_i(i = 1...n)$ these variables and the set of constraints imposed on these variables $C(P_1)...C(P_r)$, where $C(P_j)$ is the set of values of variables associated with these constraints [3].
For example, such a problem is the graph coloring problem, where $x_{1 \text{ is the variable}}...X_n$are the vertices of the graph, and the value of the variable $X_i$is the color in which the vertex is colored. The constraints have the form $C(X_i, X_j) = \{(a, b)$: a, b

– different colors} for each pair (X_i, X_j)acent number of graph vertices.

The CSP model is built on rigid logic, which means that restrictions can only take the values 0 (the restriction does not fully satisfy) or 1 (the restriction is fully satisfied). In other words, in order to get a valid solution to such a problem, all its limitations must be fully satisfied [11].

2.2 FCSP task

Fuzzy Constraint Satisfaction Problem is a type of CSP problem. It appeared at a time when the potential of CSP tasks was fully realized, and it became obvious that CSP is a poor model for designing real problems on it, where it is often impossible to do with strict conditions and complete satisfaction of all constraints. In such a situation, came to the aid of fuzzy logic Zadeh [21], which, unlike conventional hard logic, limit yourself to two constants to determine the falsity or truth of the assertion, that is, 0 (false) and 1 (true), and proposes to use for this purpose, the whole interval [0, 1]. It also offers a mathematical structure such as fuzzy sets, which do not consist of a rigidly defined set of elements, but instead consist of elements that belong to them (sets) to a certain extent.

Applying the concept of fuzzy logic to CSP problems allowed us to solve, for example, incompatible problems where all constraints cannot be fully satisfied simultaneously. The worst example is problems with a large number of constraints (over constrained), for which it is also usually difficult or impossible to find a solution using hard logic [12-15].

The first attempt to formalize FCSP is probably made by Dubois [5] and Ruttkay [16]. All subsequent attempts were very similar to the first, so let's consider the FCSP model of problems described in one of the following articles [7].

Let be the space of all possible solutions to the problem U, which consists of vectors of the form ($x_1 \ldots x_k$), where k is the number of variables in the problem, and x acquires all the other values from its domains. Enter the "membership function degree" function») A: $U \rightarrow [0, 1]$, which will determine the degree of belonging of each vector ($x_1 \ldots x_k$) up to the set of solutions, or more precisely, the degree to which this solution satisfies all the problems discussed. In normal CSP issues with strict restrictions, this function looks *like* A: $U \rightarrow \{0, 1\}$.

Then any possible solution that belongs to the "alpha slice" will be considered a solution of the FCSP problem»:

$$A\,|_{\Box} = \{u\ U\ A\ u \in |() \geq \Box\}.$$

Moreover, the method of setting the threshold α is either part of the algorithm for solving the problem, or it is manually selected during the solution.

2.3 Hopfield neural networks

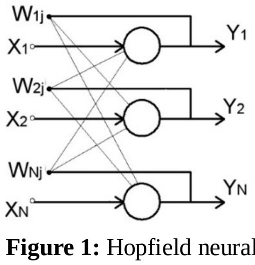

Figure 1: Hopfield neural network

The Hopfield network (figure 1) is a neural network where all neurons are connected to each other. Each neuron receives input from all other neurons and, in turn, sends its signal to all other neurons in the network. The matrix of connections of such a network is symmetrical, and the elements of the main diagonal must be zeros. These zeros mean that the neuron's signal will not be transmitted to the input itself. This condition and the symmetry condition should ensure the stability of the system. In [4], it is stated that a network with gate connections is stable only if the two conditions mentioned above are met. Stability means that the system stabilizes after a certain number of steps and after reaching a certain state, it will remain unchanged in the future. Otherwise, it may fluctuate constantly between two different States, unable to reach equilibrium [16].

The function Φ in formula (1) can be represented in various variants.

The interaction of neurons in a network is described by a function that is also called network energy (2). its Main property is that at each iteration of network operation, the energy decreases, and not necessarily monotonously. In this case, the neural network itself becomes a minimizer of the energy function. That is, after a certain update cycle, the network should reach the local minimum and then go to a stable state.

$$E\,x(\ ) = -2 \sum\sum_{i= =1\,j\,1} x\ x\ w_{ij} - \sum_{i=1} x_i \Box_i\,. \quad (1)$$

It should be noted that all this comes true only if the matrix of network weights is symmetric, then the energy function is a Lyapunov function.

2.4 Hopfield Tank Network

In 1985 John Hopfield teamed up with Dev Tank to collaborate on adapting his neural network model to solve optimization problems [10].

The main idea was to present the problem as a function of energy, provided that the better the solution, the less energy. Then we could construct a Hopfield network for this function and try to minimize this function with it [11].

Thus, if the optimization problem looks like this: minimize $f(x)$ provided that it is imposed *with r* constraints of the form $C(x) = b$, then the energy function for such a problem can be described by the formula (2):

$$E\,x() = f\,x() - \sum \Box_i P\ x_i\,(), \quad (2)$$

where x is the vector of input variables of the problem;

$f(x)$ – the target function; r – the number of constraints; $a_1 \ldots a_r$ – coefficients reflecting the" relative importance " of satisfying each of the constraints;

P_i – so-called "penalties", which are directly related to the level of satisfaction of the corresponding constraints of the problem and are equal to zero only if the constraint is completely assaulted. Otherwise, they are proportional to the satisfaction of the constraint. The higher the level of pleasure – the lower the coefficient. It is allowed that restrictions in the issue may represent equalities or inequalities [17-20].

Since the energy function must be at least quadratic, the constraints-equality in it are presented in this form:

$$P\ x_i\,() = -(Cb_{ii})^2. \quad (3)$$

This expression will only be nullif the constraint is completely satisfied, otherwise its value will be proportional to the degree of dissatisfaction.
Regarding the representation of constraints-inequalities, they should look like this:

$$P x_i ()=\Omega - (C_i\ b_i).\ (4)$$

In this formula, the omega function can be represented, for example, by the sigmoid function.
It should be noted that the penalty approach was originally developed to solve combinatorial optimization problems. We consider it in a narrower context, namely in thecontext of solving constraint optimization problems, or, in other words, for constraint satisfaction problems with flexible constraints, where the goal is to find a solution where the overall level of constraint violation is optimal (minimum). In suchdachas, there is no target function as such, that is, the amount of fines in it is the target function. That is, the first term in formula (3), in fact, does not exist.
The multipliers that stand in the derived derivatives *of* $x_j$will be the values of the corresponding weight coefficients w_{ij}, and the constant terms are the values *of* θ_i.
Most of the questions in the Hopfield – Tank method are about the selection of coefficients a_i in the formula $E(x)$ and initialization of the network state. An unsuccessful selection of these parameters can significantly affect the further course of events. For example, by initializing neurons with values that are a possible solution or close to a solution, we can thereby contribute to the fact that the network immediately falls into the local minimum. It is recommended to use values of ~0.5 for neuronal initialization, or use the approach [12]. As for the choice of coefficients a_i, the authors of the method did not give any recommendations on this, which later became the cause of extensive discussions. This will be discussed in more detail later [18].
The new method was successfully applied by the authors to solve some optimization problems, including, in particular, the problem of a traveling salesman. The results obtained were so inspiring that in 1988, a new study was published. Wilson and Pavlev wanted to test them, confirm them, and improve them. It turned out that Hopfield and Tank's results were valid only for relatively small tasks (for the traveling salesman's task, they were able to repeat the successful results only for circles of different cities, no more than 10), but it is much more difficult to successfully select coefficients for 900 neurons [10]. Subsequently, many methods were proposed to improve the network developed by Hopfield, but most of them actually did not significantly increase the effectiveness of the method, that is, the improvements were generally insignificant. However, some of them deserve attention.
Modification of the Wilson and Pavlev Hopfield Tank network
As already mentioned, the cornerstone of the Hopfield – Tank method is the selection of coefficients for the energy formula. Wilson proposed a modification of this formula, where all these coefficients are replaced by one, because the probability of correctly choosing one coefficient is much greater than guessing with the values *of n* coefficients. Therefore, in the new interpretation, the energy function has the form [10]:

$$E\,x()=f\,x()+\frac{\gamma}{2}D\,x(),\ (5)$$

where instead of the coefficients a_i, there is a single coefficient γ, and $D(x)$ should be understood as a deviation of the vector x (i.e., the solution) from the plane of constraints defined for the problem. Now, if we set a sufficiently large value for γ, we guarantee that we will find a solution.
In more detail, this is the formulation of the energy function and the method itself is described in articles [2] and [10].
Lagrange multiplier method for Hopfield-Tank network
It is impossible not to notice that the energy function behind its appearance closely resembles the Lagrange function. If a_i we substitute Lagrange factors for the coefficients ai, we get [14]:

$$E\,x(\lambda)=f\,x(\ )+\sum \lambda_i P\,x_i ().\ (6)$$

The advantage of this formulation of the energy function is that the coefficients for penalties P_i no longer need to be selected at random. They can participate in a neural network as its elements along with other neurons, which are task variables.
It follows that the success of the project in the first case will be related to the choice of a hybrid model of state regulation, and in the second case - the most effective model will be the state monopoly.

2.5 GENET and Fuzzy GENET

The GENET algorithm was introduced to the world by Wang i Tsang in 1991 [6] and was a logical extension of GDS networks. As in GDS, here neurons are divided into clusters. If the cluster is a separate task variable, then neurons within the cluster represent possible values of the variable. All neurons can take the values 1 or 0, or "on" or "off" respectively. In each cluster, at any given time, only one neuron can be in the "on" state, or have a value of 1, respectively. in Fact, this means that the neuron in the cluster that is in the "on" state is the value that at that time is assigned to the variable that this cluster represents.
Neurons from different clusters are connected only by the variables they represent: if the corresponding variables are bound by a constraint in the task, then the neurons are also bound. Moreover, in these clusters, only those neurons have connections among themselves that represent conflicting values of variables. For example, if you set a task like in figure 2, then the GENET network will look like in figure 3. We recommend initializing the network weights -1.
What is significant in this method is that, unlike many other methods, the GENET developers conducted experimental studies comparing the effectiveness of THIS method with the efficiency of conventional methods for solving CSP problems. They compared it with two local search algorithms: GSAT [19]-on a graph coloring problem, and min – conflicts hill climbing (MCHC)-on random-generated CSP problems [19].

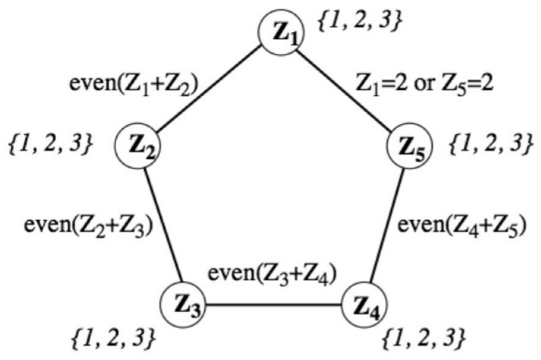

Figure 2: Example of a CSP issue

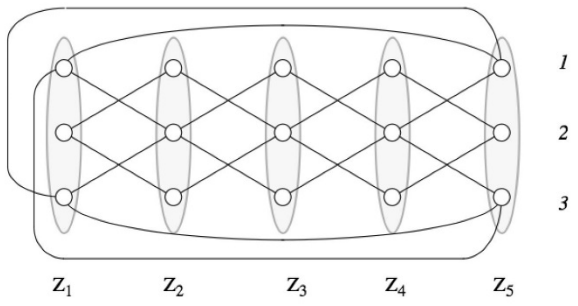

Figure 3: Example of a GENET network for a problem in figure 2

Figure 2 shows us the example of a CSP issue.
As a result of the first comparison, it was found that both algorithms met all the test tasks. Below are the results in figures [6].
The second experiment, which tested the effectiveness of GENET and some of its modifications GENET2, as well as the MCHC algorithm and its modification MCHC2, showed that GENET was again the winner, and by a striking margin. The test tasks were CSPS with Adachi with 50 variables, each of which had a domain of 10 values. Restrictions for these issues were generated randomly. The graph (Fig. 3) shows the success rate of each algorithm, which is measured as a percentage of the total number of tasks solved [6].

2.6 Fuzzy GENET

Logical continuation GENET became its version for CSP issues with flexible restrictions, i.e. for Fuzzy CSP (FCSP). This is how the algorithm called Fuzzy GENET was developed [20].
The fuzzy GENET algorithm differs from the basic GENET algorithm in that each connection between neurons in a neural network is described by a two-level satisfaction level of the constraint associated with these variables (all constraints in the problem are binary), and a weight factor $W_{i\,jy\,z}$. The weighting factors are initialized as $W_{i\,jy\,z} = □_{i\,jy\,z} - 1$. InCE, the rest is basically the same as in GENET.
The authors of the paper on Fuzzy GENET also conducted a number of experiments to compare the effectiveness of GENET and Fuzzy GENET on conventional CSP tasks. Both algorithms showed similar results, which means that Fuzzy GENET is universal and in General is a worthy replacement for its predecessor.

3. CONCLUSION

John Hopfield was a pioneer in trying to adapt his neural network to solve CSP problems. Despite the failure of the first attempts and the criticism that his published results were not confirmed when they were verified by other researchers, Hopfield's work laid the Foundation for further research on this topic. The Hopfield-Tank network clearly highlighted all the problematic areas in the methodology of applying the Hopfield network to solving CSP problems and thus stimulated the active development of research aimed at improving these shortcomings.
The first major improvement of the network was the introduction of a new energy formula by Wilson and Pavlev [10], where all coefficients were replaced by a single γ. This greatly simplified the initialization of the neural network, which needs to select "magic" values for a large number of input parameters (constraint coefficients, neuron weights), because the initial configuration significantly affected how the network would develop further, whether it would not fall into a local minimum, or become infinitely oscillatory between two States. A follower of Wilson and Pavlev was Lach [14], who saw in the Hopfield network energy function a similarity with the Lagrange function. He was the first to compare the results of his experimental studies not only with their predecessors, but also with completely different algorithms for solving the same problem, and the results of the comparison spoke in favor of Lach.
The next step in improving the Hopfield network was its hybridization with a metaheuristic algorithm – the annealing method. Thus, the Boltzmann machine, the Cauchy machine, and a hybrid of these two methods appeared [11]. In General, according to the author of the article [10], the hybridization of neural networks with metaheuristic algorithms is one of the two promising directions for the development of methods based on neural METAS in the field of CSP problem solving. The second promising direction is the search for alternative approaches in the field of neural networks, i.e. the use of other types of networks to solve CSP, or the development of other models for designing CSP tasks based on neural networks of EGE.
The first really serious descendant of the Hopfield network was the GDS network, which was designed to solve the problem of scheduling the Hubble telescope schedule [8]. And its logical extensions – Genet [6] and Fuzzy GENET [20] algorithms-became the first attempt to apply the Hopfield network principle to solving the FCSP problem.

REFERENCES

1. Agrawal, D., Trandel, G., **Dynamics of policy adoption with state dependence.** Regional Science and Urban Economics, 79, 2019, p.103471. https://doi.org/10.1016/j.regsciurbeco.2019.103471
2. Cheng, Y., Ma, H., **China Sports Lottery System Structure Legal Regulation Improvement**. DEStech Transactions on Social Science, Education and Human Science, 2018.
3. Dorohykh V. M., **Administrative and legal regulation of gambling business in Ukraine**. Extended abstract of candidate's thesis, 2004.

4. Gataker T., **The Nature and Uses of Lotteries**. Review by: Peter Stone History of Political Thought Vol. 34, No. 1, 2013, pp. 172-175.
5. Getmantsev D.O. (2008) **Hazart game in Ukraine and abroad]**. SEEM Print
6. Huang, Y., Wen, Q., **Auction-lottery hybrid mechanisms**: structural model and empirical analysis. International Economic Review, 60(1), 2018, pp.355-385.
7. Kovtun E. V., **Gambling in Russia: law regulation**. Yuridicheskiy centr Press, 2005.
8. Lyskov M. O., **Public administration of lottery sphere**. Doctor’s thesis, 2017.
9. Osyka S. G., **State regulation of global gambling industry**. K.I.S., 2011.
10. Parra, Carlos, **Capital Mobility and Regulation Frictions**: Evidence from U.S. Lottery Winners, 2018.
11. Petrychko N. O., **Illegal gambling: criminal-law and criminological research**. Candidate’s thesis, 2010.
12. Pinyaga R. O., **Investigation of crimes related to gambling]**. Extended abstract of candidate’s thesis, 2015.
13. Pohoretskyi M. A., Toporetska Z. M., **Gambling: history and legal regulation**. Dakor, 2014.
14. Pohoretskyi M. A., Toporetska Z. M., **Investigation of gambling business: procedural and forensic principles**. Alerta, 2015.
15. Pohoretskyi M. A., Vakulyk A. O., Serhieieva D. B., **Investigation of Economic Crimes**. Dakor, 2015.
16. Rugh Blair, Banking and Lotteries. **This Week's TriComply Newsletter Article**, 2012.
17. Sevostyanov R. A., **Problems of criminal liability for the organization and maintenance of illegal gambling**. Extended abstract of candidate’s thesis, 2009.
18. Vysotska V. V., **Conduction of gambling business: criminal and legal analysis of the crime**. Extended abstract of candidate’s thesis, 2016.
19. Bhardwaj, B. K. (2019). **A critically review of data mining segment: A new perspective**. International Journal of Advanced Trends in Computer Science and Engineering, 8(6), 2984–2987. https://doi.org/10.30534/ijatcse/2019/50862019
20. Bhat, A. H., & Balachandra Achar, H. V. (2019). **Dual parametric stabilization of interference and throughput in wireless sensor network-optical communication**. International Journal of Advanced Trends in Computer Science and Engineering, 8(5), 1937–1945. https://doi.org/10.30534/ijatcse/2019/18852019
21. Bindumadhavi, G., Kumar, V. V, & Sasidhar, K. (2019). **A new frame work for content based image retrieval based on rule based motifs on full texton images**. International Journal of Advanced Trends in Computer Science and Engineering, 8(4), 1083–1098. https://doi.org/10.30534/ijatcse/2019/15842019