Applying Hopfield Neural Networks for Artificial Intelligence problems

Term project
CS 551 (Artificial Intelligence)

Gorodnichy D.O.

December 6, 1995

Abstract

Artificial Intelligence (AI) is known to be rich with problems, where finding the solution by conventional search methods is computationally intensive. The time required is often exponential to the number of variables. Principly different way of searching for the decision is to build a dynamic self-organizing system, where the stable states correspond to the desirable solutions. In the paper the possibility of using such nondeterministic methods for resolving Artificial Intelligence problems is studied. It is shown why Hopfield Neural Networks (HN) are so suitable for the role of such a dynamical system. The following approaches of applying Hopfield Neural Networks for solving AI problems are discussed: the Energy approach, the Probability approach, the Graph approach. The last approach is based on using a Hopfield clique Network, introduced by Jagota in 1990. This network is known by the property that its stable states are exactly the maximal clique of the underlying graph.

The purpose of the paper is to give a clear idea of how HN can be applied for different AI problems. All the the described approaches are illustrated with concrete examples. The main theorems, needed for understanding the mechanisms, are stated and the ideas of their proofs are presented. The areas, where these approaches are applicable and where not, are shown.

1 Introduction

1.1 Two approaches of searching for a solution

Artificial Intelligence (AI) is known to be rich with problems, where finding the solution by conventional algorithmic methods is computationally intensive. Since these methods look over all possible variable instantiations (which is usually accomplished by a tree-search technique), the time required for the search is exponential in the number of variables. Much research has been done in optimizing and finding heuristics to improve the performance of the tree search. Some improved algorithms were proposed and theoretical evaluation were done to analyze their effectiveness (see [Kondr95]). But these tree-search algorithms while showing a good performance on certain NP-hard problems still cannot escape from behaving badly on some others – NP-hard problems. This is because they can not break away from the circle of conventional *"looking over"*.

Principly different way of searching for the decision is to build a dynamic self-organizing system, where the stable states correspond to the desirable solutions. The task of the researcher is then to assign somehow these stable states and leave the problem of finding (i.e. converging to) these states, starting from some initial state, to the system. In this case the researcher may not know, how (by what particular steps) the system converges to the solution. This alternative approach might yield: 1) increase in speed; 2)when it is difficult to find an exact solution it would give approximate one (which is due to the ability to converge in desirable direction). Other positive feature might be that such system may be very suitable for parallel implementation. That's why it is very important to study and develop this approach.

1.2 Why Hopfield Networks?

Consider a dynamic system of N units (*neurons*), the evolution of which in time is determined by the *Update Rule*:

$$Y_i(t+1) = F(S_i(t)), \quad S_i(t) = \sum_{j=1}^{N} W_{ij} Y_j(t), \quad Y_i \in \{-1; +1\}, \quad i = 1..N, \tag{1}$$

where Y_i is a *state* of neuron, and $F(x)$ is a "hard" threshold function for binary neurons and a "smooth" threshold function for analog ones[1].

It's easy to show that:

Theorem. *If $W_{ij} = W_{ji}$, then the energy of such a system defined as*

$$E(t) \equiv -\vec{Y}(t) \cdot \vec{S}(t) = -\sum_{i=1}^{N} Y_i S_i = -\sum_{i=1}^{N}\sum_{j=1}^{N} Y_i Y_j W_{ij} \tag{2}$$

is an ever decreasing function of time.

◯ The proof (in simplified form) consists in checking that $\frac{dE}{dt} < 0$:

$$\frac{dE}{dt} = \frac{\partial E}{\partial Y_1}\frac{\partial Y_1}{\partial t} \cdots \frac{\partial E}{\partial Y_N}\frac{\partial Y_N}{\partial t} =^{(2)} -\sum_{i=1}^{N}\sum_{j=1}^{N} Y_i \frac{\partial Y_i}{\partial t} W_{ij} - \sum_{i=1}^{N}\sum_{j=1}^{N} \frac{\partial Y_i}{\partial t} Y_j W_{ij} =^{(W_{ij}=W_{ji})}$$

$$-2\sum_{i=1}^{N} \frac{\partial Y_i}{\partial t} \sum_{j=1}^{N} Y_j W_{ij} = -2\sum_{i=1}^{N} \frac{\partial Y_i}{\partial t} S_i < 0,$$

as $Y_i(t+1)S_i(t) > 0$ by (1). ◯

Provided that $E(t) \geq -\sum_{i=1}^{N}\sum_{j=1}^{N} W_{ij} = const$, this means that **no matter what the initial state of the system, it will for sure finally converge to some stable state**. If these stable states are those we'd like to find, then we say that the system exhibits representational power and error-correction ability.

For the first time this property was shown by Hopfield in 1982 ([Hopf82]). Since then such a system is referred to as *Hopfield Neural Network* (HN).

More generally in HN a neuron is defined by equation $Y_i = F(S_i)$, where $S_i = (\mathbf{W}Y)_i - B_i$ ($\mathbf{W}$-weight matrix, B_i - *bias* of a neuron). Also neuron can not only be discrete, but analog (i.e. $Y_i \in [-1, +1]$), then instead of $F_{hard}(x)$ we should use "smooth" threshold function (e.g. $f(x) = 2(\frac{1}{1+e^{-\lambda x}} - 1)$). Moreover, the update rule can be synchronous or asynchronous (updating only one neuron at a time). Neurons can also be clamped sometime, i.e. forced to be in the same state during the evolution of the network. Thus there are different models of HN: discrete and analog (or continuous), synchronous and asynchronous, with clamped neurons (see [Cough95] for more elaborate definitions). But as can be seen from the reasoning in the proof of the theorem above, all these models possess the property of *"minimizing energy"*. It is this property that is considered as the *main property of Hopfield Networks* (with the only difference that the energy function might look a little different for different types of neurons).

1.3 Main objectives

So as we can see HN seems to be suitable for the role of a dynamical system described in Section 1.1[2]. But for serious application of HN in AI, we have to find an efficient way of representing a variety of knowledge that AI systems deal with. The importance of finding the ways of knowledge representation in terms of neurons is apparent now, and it is this question that I will mostly study in this paper. I will summarize known approaches and show the areas where they are applicable and where not. All approaches will be illustrated by examples.

[1] See Nomenclature at the end of the paper for unknown symbols or variables

[2] HN is not the only type of neural network used as a self-organizing dynamical system. Kohonen feature mapping network is other type, which is used for AI problems. For example very good results were obtained by applying this network for Traveling Salesman Problem (see [Angen91]). See also [Hertz91] for overview of different types of neural networks.

Then I will focus my attention on the variant of HN, called Hopfield clique Network (HcN) proposed by Jagota in 1990. HcN is constructed in such a way that its stable states are exactly the maximal clique of the underlying graph. The Maximum Clique problem is known to be NP-hard, and even NP-hard to approximate. That is why studying such a practical algorithms for finding approximate solutions for the maximum clique problem as those provided by HcN appeared to be a challenging problem.

I will show how HcN can be applied to Constraint Satisfaction Problems (CSP), N-Queen puzzle will be a particular example. I will discuss problems occuring when applying HcN, some of the deficiencies of HcN (not mentioned by other authors) will be shown.

2 Overview of Existing Approaches

In this section I will discuss the problem of applying Hopfield neural networks to AI tasks and describe the main approaches.

2.1 Energy Approach

This approach was proposed by Hopfield in 1985 ([Hopf85]) and is considered as first contribution to the above problem. The idea is the following:

Having an optimization problem, we reformulate it as a problem of minimizing the function of unknown variables. Considering this function as an energy function of a HN, and these variables as states of neurons of this network, we can find the weights of

this network. By such a way we construct the HN, stable states of which corresponds to the minima of the energy, and are therefore the solutions.

The typical example of applying this approach is the Traveling Salesman Problem ([Hopf85],[Xu91]):

Δ For N cities we consider $N \cdot N$ neurons V_{xi}, meaning x city is in j place in the tour. E is defined by the formula

$$E = A\sum_{x=1}^{N}\sum_{i=1}^{N}\sum_{j\neq i}^{N} V_{xi}V_{xj} + B\sum_{i=1}^{N}\sum_{x\neq y}^{N}\sum_{x=1}^{N} V_{xi}V_{yi} + C(\sum_{x=1}^{N}\sum_{i=1}^{N} V_{xi} - N)^2 + F\sum_{x=1}^{N}\sum_{x\neq y}^{N}\sum_{i=1}^{N} d_{xy}V_{xi}(V_{yi+1} + V_{yi-1})$$

A, B, C, D are some large constants: A prevents one city to appear at more than one places, B prevents more than one city to appear at the same place, C charges for missing a city in the trip and D charges for great total distance.

From here using Eqn. 2 we obtain the weights

$$W_{xi,yj} = -A\delta_{xy}(1-\delta_{ij}) - B\delta_{ij}(1-\delta_{xy}) - C - Dd_{xy}(\delta_{ji+1} + \delta_{ji-1})$$

Starting from some initial state $\vec{V}(0)$ (V_{xi} being random small positive values), the network will converge to the state of the energy minima.

It's clear that solutions obtained by such a network are influenced by the A,B,C,D constants, and are only approximate ones. But as shown in [Xu91], by appropriately choosing the form of energy function and these constants **"for random distance matrix this approach outperforms such a well-known heuristic algorithm as the Lin and Kernigan algorithm ([Gold85]) significally (it is 70 % better with confidence level $\alpha = 0.00188$)"**. Δ

Moreover, we did not take into account another advantage of this approach, that is the possibility of exploiting the massive parallelism of neural networks.

The bottleneck of this approach is that sometime the network can be trapped in the local minima of the energy, and then the solution obtained will be far from the real one. But as shown in [Xu91] introducing a generalized Hopfield model eliminates this problem.

Unfortunately, not many problems in AI can be represented as finding a minimum of a function, and this is the main drawback of the energy approach. For example it is impossible to apply a minimizing function for describing structured knowledge representations, also known as schemata.

2.2 Probability Approach: Storing Schemata by Neural Networks

For representing schemata, consider a HN, where each unit corresponds to an element in schemata. It's possible to perceive that if we'd like that in a stable state some schemata elements be both "ON"(fired) (e.g. "oven" and "stove"), then we have to

assign positive value to the weight between them (e.g.$W_{oven,stove} = 1$), and otherwise: when some elements should have different values in a stable state, we should assign negative value to the weight between them.

In 1986 Rumelhart et al [6]proposed approach which is based on this idea. In particular they proposed to calculate probability of co-occurrence of the events, and set then the weights and bias by the formulas

$$W_{ij} = -ln\frac{P(V_i = 0, V_j = 1)P(V_i = 1, V_j = 0)}{P(V_i = 0, V_j = 0)P(V_i = 1, V_j = 1)},$$

$$B_i = -ln\frac{P(V_i = 0)}{P(V_i = 1)}$$

i.e. the more often the two unit-events occur simultaneously, the greater is the weight between them, $W_{ij} = 0$ meaning the events are mutually independent.

Lets demonstrate how this approach works by a particular example:

Δ For the schemata { {"oven","stove"}, {"stove","kitchen","table","chair"} }, which in neuron form can be rewritten as $\{\{V_1 = 1, V_2 = 1, V_3 = 0, V_4 = 0, V_5 = 0\}, \{V_1 = 0, V_2 = 1, V_3 = 1, V_4 = 1, V_5 = 1\}\}$ we obtain the following values for weight

s and biases

$$W_{12} = -ln\frac{\frac{1}{2}\cdot\epsilon}{\frac{1}{2}\cdot 1}, \quad W_{13} = -ln\frac{1\cdot 1}{\epsilon\cdot\epsilon} \quad \cdots$$

$$B_1 = -ln\frac{\frac{1}{2}}{\frac{1}{2}}, \quad B_2 = -ln\frac{1}{1}, \quad \cdots$$

where ϵ is a small positive value.

Then given the initial state {"chair"} (i.e.$\{V_1 = 0, V_2 = 0, V_3 = 0, V_4 = 0, V_5 = 1\}$), the network will converge to the state {"stove","kitchen","table","chair"} Δ

As the experiment showed such a method actually gave stable states corresponding to the schemata stored, but no theoretical proof was obtained, neither any evaluations were done as for the attractivity of these states and the existence of other (so called

spurious) states.

For many problems we cannot assign the probabilities of the event. So, this approach (also called *Probability approach*) is unsuitable for such AI problems as constraint satisfaction and boolean formulae.

Because of this it is not surprising why a new approach, more theoretically grounded, proposed by Jagota in 1990, caused a new wave of interest to the problem.

3 Hopfield clique Network

Before describing a graph based neural network, let first recall the main things we know about graphs and the problems dealing with them, the most interesting of which is the Max-Clique problem.

3.1 The Max-Clique problem

In a graph $G_N(V, E)$ with undirected edges E, a *clique* is a set of vertices such that every pair is connected by an edge. A clique is *maximal* if it is the largest clique. For example in Figure 1, the vertex sets: {2,3,4,5} is maximal clique (of size 4).

Max-Clique is the optimization problem of finding a maximum clique in a given graph. This problem is NP-complete (see [Garey79]). That is, with high probability (i.e. unless $P = NP$) there is no algorithm that, for *every* given graph, can find a maximum clique in time polynomial in the size of the graph.

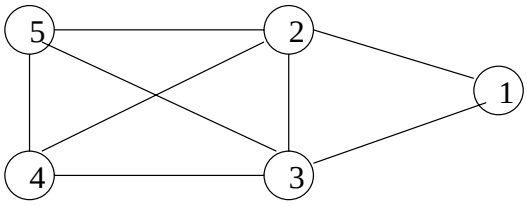


Figure 1: Graph $G_N(V,E)$, N=5, {2,3,4,5}—the maximal clique of the graph

Theoretical results [Cres91] also show that the Max-Clique problem is not only NP-hard but also NP-hard to approximate within a polynomial factor. More formally, unless $P = NP$, there is a fixed $\epsilon > 0$ such that no algorithm can find, for *every* n-vertex graph, a clique of size $f(n)$ such that $\frac{f^*(n)}{f(n)} < n^\epsilon$, in time at most polynomial in n (where $f^*(n)$ is the size of its largest clique)

Several problems can be efficiently reduced to the Max-Clique problem, and thus solved by algorithms for the Max-Clique problem. For example, there are reductions of satisfiability of Boolean Formulae to the Max-Clique (see [Cres91]). Many other problems are in principle efficiently reducible to Max-Clique because of its *NP*-completeness. Some reductions also preserve approximability. In such cases, reduced problems can also be approximated, by approximating Max-Clique. Max-Sat, the optimization problem of finding the largest number of simultaneously satisfiable clauses, is one such problem ([Cres91]).

Other problems which can also be modeled as Max-Clique Problem are Constraint Satisfaction Problems (e.g. N-queens, graph coloring).

In the next chapter we will see how the Max-Clique problem can be solved with the aid of HcN.

3.2 HcN as a way to solve the Max-Clique problem

In 1990 Jagota introduced a variant of HN, called Hopfield style Network (after 1992 referred to as Hopfield clique Network) built in such a way that its stable states are exactly the maximal cliques of the underlying graph ([Jag92]).

Definition. *A discrete asynchronous Hopfield Network of N units ($Y_i \in \{0,1\}$), with weights $W_{ij} \in \{-K,1\}$,$(K>0)$, $W_{ii}=0$, and bias $B_i \in (0,1)$ is called Hopfield clique Network (HcN).*

HcN is characterized by its *underlying* graph, i.e. by graph $G_N = (V,E)$ whose vertices are the units and (undirected) edges are the positive weights ($W_{ij} = W_{ji} = 1$). That is, $\{V_i, V_j\} \in E(G_N)$ if and only if $W_{ij} = 1$ ((see.Fig 2)

The following *learning rule*, can be used to make any given graph G the graph underlying the HcN:

The initial weight state is: for all $i \neq j$, $W_{ij}(0) = -K$. Any set of vertices $V' \subseteq V$ is stored in HcN on step m as follows. For all $i \neq j$, if $S_i, S_j \in V'$ than change W_{ij} as follows $W_{ij}(m+1) = W_{ij}(m)$

The dynamics of the network is governed by the **Steepest Descent Rule**: In every cycle, exactly one unit i—the one whose switch would maximally decrease the energy defined by Eqn.2 (i.e. $\Delta E_i(t+1) = -Y_i(t+1)S_i(t+1) = max_{i=1..N}\Delta E_i(t+1)$) —is first picked and then switched by Eqn.1.

The name of HcN is attributed to its property stated in the theorem below. The understanding of this theorem is very important for the further investigation, and its proof is rather simple, that's why I thought it is necessary to give it in this paper.

Theorem 1. *If $K > N$, then the stable states of HcN are exactly the maximal cliques of the graph G_N underlying it.*

◯ Lets show this on a particular example (see. Fig. 3). Take $N = -10$ According to the *Update Rule* (Eqn. 1):

1. A unit (vertex) i can switch $ON(1) \to OFF(0), iff$

$$0 > \sum_{all\_other\_vertices} WijS_j = \sum_{adjecent\_with\_i} 1 \cdot S_j + \sum_{not\_adjecent\_with\_i} (-10) \cdot S_j, \quad S_j \in \{0,1\} \quad \Longrightarrow$$

$$Y_i(t+1) = \begin{cases} 1 - Y_i(t), & \text{if} \quad \Delta E_i(t+1) = -Y_i(t+1)S_i(t+1) = max_{i=1..N}\Delta E_i(t+1) \\ Y_i(t), & \text{otherwise} \end{cases}$$

$Y_i \in \{0;1\}$
$W_{ij} \in \{-10, 1\}$
◯ – 1 ("ON")
◯ – 0 ("OFF")

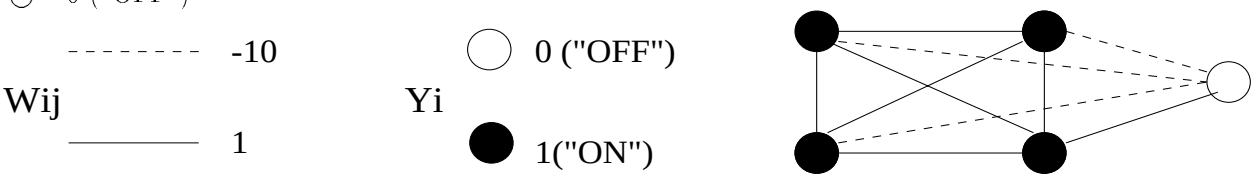


Figure 2: Hopfield clique Network: its underlying graph $G_N(V,E)$, N=5

$\exists$ unit k not adjacent to this i such that $Y_i = ON(1)$. This means, for example, that the state in Fig.3 (a) will not be stable. So we obtain

Condition to be a clique. *If $\neg\exists$ a unit i: switches $ON \to OFF$, then all ON units (which we will designate as Y_{ON}) form a clique.*

But this condition allows a state shown on Fig.3 (b): $Y_{ON} = \{3,4,5\}$.

2. A unit (vertex) i can switch $OFF(0) \to ON(1), iff$

$$0 < \sum_{all other\_vertices} W_{ij}S_j = \sum_{adjecent\_with\_i} 1 \cdot S_j + \sum_{not\_adjecent\_with\_i} (-10) \cdot S_j, \quad S_j \in \{0,1\} \quad \Longrightarrow$$

$\exists$ at least one unit k adjacent to this i such that $Y_i = ON(1)$. This will not allow the state depicted on Fig.3 (b) to be stable. So we have

Maximality Condition. *If $\neg\exists$ unit i: switches $OFF \to ON$ and Y_{ON} is a clique, then Y_{ON} is a maximal clique.*

The small bias b_i prevents the values at the right side of equations above to be equal to zero. ◯

In [Gros92] and in [Jag92] the new type of HcN dynamics called **Steepest Ascent Dynamics** was proposed. It is very similar to the Steepest Descent dynamics, except for that the unit i is first picked and then switched, if its switch would maximally

increase the energy.

We have another theorem for such a dynamics([Jag.2]). Its proof is similar to that of Theorem 1.

Theorem 2. *For Steepest Ascent HcN, if $K > N$ then the stable states of HcN are exactly the maximal independent set of the G_N underlying it, (where Independent Set (IS) is defined as a complement to the clique graph).*

In [Jag94] Jagota showed the scope of knowledge representation problems, which are suitable for HcN. They are 1) constraint satisfaction, 2) retrieval of stored schemata from incomplete information, 3) multi-relations, 4) boolean formulae, 5) sets and graphs.

Lets demonstrate how schemata is stored by HcN (example of Sect. 2.2)

Δ Considering each (of two) tuples of schemata as a clique of the graph underlying the HcN, we obtain the HcN of size N shown in Fig.2. Note that although this representation is very similar to that of the *Probability approach* (only the weights differs), it is very different because we can be sure (by Theorem 1) that our schemata is actually stored. But this is not true for the *Probability approach*. Δ

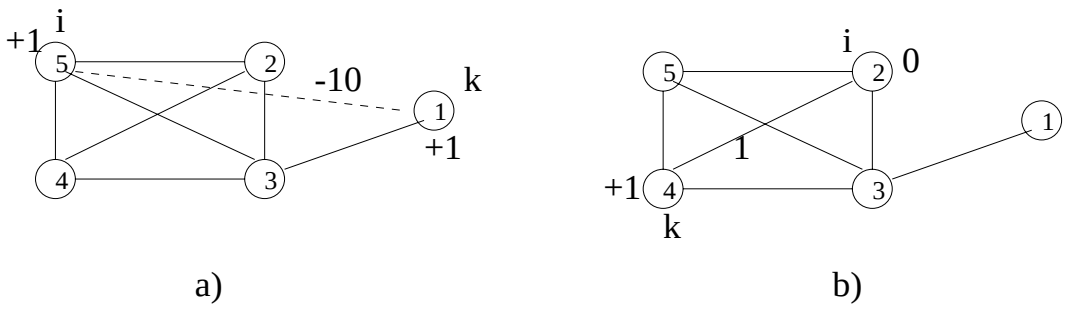


Figure 3: Unstable states of HcN: (a) by **Clique Condition**, (b) by **Maximality Condition**

4 HcN and Constraint Satisfaction Problems

The Constraint Satisfaction Problem is an important problem in Artificial Intelligence, with applications in several domains such as computer vision; natural language processing (see [Mac92],[Dec92]). We also draw our attention to this problem since it is one of the most representative problems in AI.

4.1 Constraint Satisfaction Problem in terms of graphs

Definition. *A Constraint Satisfaction Problem (CSP) is a set of N variables $X_1, ..., X_N$ which take values from the sets $D_1, ..., D_N$ respectively, under given binary compatibility constraints. For all variable pairs X_i, X_j, $C(X_i, X_j) \subseteq D_i \times D_j$ is a binary constraint set and specifies the pairs (d_i, d_j) such that $X_i = d_i$ is* compatible *with $X_j = d_j$.*

A solution of a CSP is an N-tuple of values to the N variables so that all pairs of value assignments are compatible.

Lets reformulate the CSP problem as the Max-Clique problem on a graph. For an N variable CSP, define an N-partite graph G, partition i representing variable X_i. Partition i contains $|D_i|$ vertices, one for each possible value of X_i and named by the value. We make $C(X_i, X_j)$ the set of edges between vertices in partition i and partition j. Every compatibility constraint of the CSP is represented as an edge of the graph. The fact that the vertex-set of a partition has no edges enforces the constraint that a variable may be assigned at most one value. Let a k-solution denote a set of pair-wise compatible assignments to k variables of a CSP. We say that the size of a solution is k. It is clear that there is a one-to-one correspondence between k-solutions of a CSP and k-cliques of its reformulated graph. The set of N-cliques is exactly the set of the CSP's solutions of size N.

It is useful to view the CSP problem as the following optimization problem: find the largest size k-solution. In the case when an N-solution exists, the objective reduces to the one to fully solve the CSP. Otherwise, sufficiently large k-solutions suffice. Even when an N-solution exists, it may be computationally difficult to find one and one might relax the objective to find a sufficiently large solution.

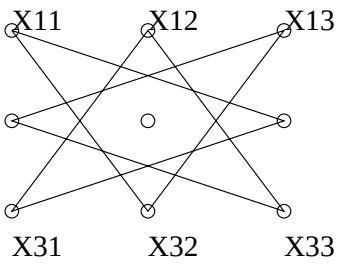
Such a representation of the problem was used by Jagota ([Jag92, Jag91]) and Grossberg ([Gross]), and below it is shown how they proposed to use HcN for the N-Queen problem.

4.2 N-Queen problem

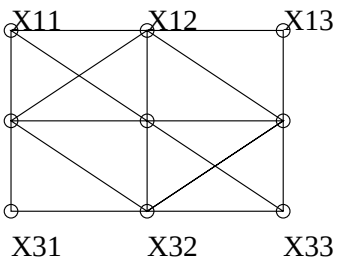
Consider an HcN with $N \cdot N$ units (here we will refer to them as variables). The variable X_i having value j denotes that there is a queen on column i and row j. The constraints that no more than one queen can be in the same row or the diagonals can be captured by the binary constraints $C(X_i, X_j)$. The constraints that no more than one queen can be on the same column are already captured by our representation. Figure 4 shows clique and Independent Set graph (IS) presentations for the 3-Queen problem. As can be noticed the size of the training set for the IS presentation is $N + N + ((N-2)2+1)2 = 6(N-1)$ (see Fig.4) that is much less than for clique presentation (where you have to train net

work with each edge of the clique graph.

This explains why Jagota used steepest ascent dynamics of HcN for this problem [Jag91]. So, the HcN is trained on IS graph, and then given the initial state "all units ON" according to the Theorem 2 HcN



(a)



(b)

$\{X_{11}, X_{12}, X_{13}\}$ $\{X_{21}, X_{22}, X_{23}\}$ $\{X_{31}, X_{32}, X_{33}\}$ — N sets
$\{X_{11}, X_{21}, X_{31}\}$ $\{X_{12}, X_{22}, X_{32}\}$ $\{X_{13}, X_{23}, X_{33}\}$ — N sets
$\{X_{11}, X_{22}, X_{33}\}$ $\{X_{12}, X_{23}\}$ $\{X_{21}, X_{32}\}$ — $(N-2)\cdot 2-1$ sets
$\{X_{31}, X_{22}, X_{13}\}$ $\{X_{21}, X_{12}\}$ $\{X_{23}, X_{32}\}$ — $(N-2)\cdot 2-1$ sets

(c)

Figure 4: N-Queen Problem, N=3: (a) Clique graph Representation, (b) Independent Set graph (IS) Representation, (c) Training set for IS Representation (not allowed relations)

converges to maximal Independent Set. If this IS is of size N, then the solution is found. If the size of IS is less than N then HcN found k-solution of the problem.

As it is shown in [Jag91] the results obtained by such an approach can not compete with that of some others ([Sosic90]). But it is also shown that "**if instead of finding a legal solution, the problem is to find as large a partial solution as possible but very rapidly, then the algorithm is very successful at that**".

4.3 HcN and Spurious states

Let's first illustrate what is a spurious state in a HcN and how they appear:

Δ Refer again to the graph at Fig. 1. This graph might represent the following schemata: { {"1", "2"}, {"2", "3", "4"}, {"4", "5", "2"} }. But when converging to the maximal clique we will obtain not a stored set, but {"2", "3", "4", "5"}, and this will be a spurious state. Δ

From the theory of HN ([Flor89]) it is known that "*The problem: given a* $\mathbf{W}$*, how many* $\vec{Y}$ *such that* $\vec{Y} = sign(\mathbf{W}\vec{Y})$*, is #P-complete* ".

Provided that the number of stable states is equal to the number of states a HN is trained on (which is known) plus the number of other spurious states, we obtain that we can not even estimate (in polynomial time) the number of spurious states, and this number does affect the results obtaining by applying the HcN approach to a particular problem. This means that **given an AI problem, after reducing it to the Max-Clique problem and obtaining a solution as a stable state of HcN, we will not be able to even say (in polynomial time) is the solution obtained is desirable or spurious.**

This fact is not stated by Jagota([Jag91,92,94]), and it seems to be the main deficiency of the HcN approach. It is worth mentioning that he and other researchers are aware of the spurious states problem and moreover they propose some ways to handle this problem: 1) [Jag91] proposes to apply conventional backtracking through all the stable states, while 2) [Ueber93] introduces more complicated version of HcN, where spurious states are escaped by means of increasing the size of the network.

5 Conclusions and directions for further research

This paper was not intended to provide experimental results, nor to make any comprehensive comparison among methods, but rather to give a clear idea of another, principly different from conventional search algorithms, approach of solving AI problems - th

e approach of applying Hopfield Neural Networks.

I summarized the following approaches : the Energy approach, the Probability approach, the Graph (or HcN) approach, and showed the areas where they are applicable and where not. Every approach was explained with a concrete example to make their mechanisms understandable. I stated the main theorems, needed to understand the problem. The ideas of the proofs was provided.

I described the recently introduced Hopfield clique Network. I showed how this network can be applied to solve (approximately) various NP-complete problems in AI. In particular I demonstrated the way Constraint Satisfaction Problems can be solved by the HcN, N-Queen puzzle being an example.

I also showed the deficiency of the HcN approach. The HcN approach is defined as reducing an AI problem to Max-Clique Problem and then finding stable states of the HcN, which will be exactly maximal cliques of the underlying graph.

From this paper it should become clear that the self-organizing system approach of solving the problems should also be developed along with conventional search methods. In this aspect this paper could be considered as the basic material for further investigations. Below is the list of questions that still need to be explored.

- First there's a need for the experiments that would apply HcN for particular problems. Since HcN was introduced comparatively recently, few such an experiments were carried out (only those mentioned in [Jag91, Gros93]). But in order to compare the performance of HcN approach with others this is significant. Here are some questions which could be answered by empirical study:
  - The number of iterations in HcN *vs* number of visited nodes in a search algorithms for the same problem.
  - What is the cost(time) of backtracking through all the stable states of HcN in comparison with that of converging to these states.
- There are also theoretical questions dealing with spurious stables:
  - Is it possible to avoid these local minima by means of adjusting weights, not by increasing the size of the network? (e.g. we have a freedom of choosing K in $W_{ij} \in \{-K; 1\}$)
  - What corresponds in the original problem (which was reduced to Max-Clique problem) to these spurious states? Can this information be useful? (e.g. in HN theory there are known cases which profit from this [Hertz91).

There are many problems that need to be studied and hopefully this paper will encourage further investigation.

Nomenclature

$\bigcirc$ - the beginning of a proof.
$\bigcirc$ - the end of a proof.
Δ - the beginning of an example.
Δ - the end of an example.
: - such that (e.g. $\exists k : Y_k = 1$)
$=^{(X)}$ - equals because of X

$$sign(x) = \begin{cases} x & x > 0 \\ 0 & x = 0 \\ -x & x < 0 \end{cases}$$

$$\delta_{xy} = \begin{cases} 1 & x = y \\ 0 & otherwise \end{cases}$$

References

[Angen91] B.G.Angeniol and J-Y. Le Texier. Self-Organizing Feature Maps and the salesman problem. *Neural Networks 1, 289-293, 1988*

[Cough95] J.P.Coughlin, Neural computation in Hopfield networks and Boltzmann machines. *Newark: University of Delaware Press; London; Cranbury, NJ: Associated University Presses, c1995.*

[Cres91] P. Crescenzi, C. Fiorini, and R. Silvestri. A note on the approximation of the maxclique problem. *Information Processing Letters*, 40(1):281–298, October 1991.

[Dec92] R. Dechter. Constraint networks. In S.C. Shapiro, editor, *Encyclopedia of Artificial Intelligence, Second Edition*, pages 276–285, New York, 1992. John Wiley and Sons.

[Flo89] P. Floreen and P. Orponen. On the computational complexity of analyzing Hopfield nets. *Complex Systems*, 3:577–587, 1989.

[Garey79] M. R. Garey, & D.S. Johnson, Computers and Intractability—A Guide to the Theory of NP-Completeness, *W.H. Freeman and Company, 1979.*

[Gold85] B.L. Golden and W.R.Stewart Empirical analysis of heuristics. *In E.L Lawler et at (eds) The Travelling Salesman Problem (pp. 207-249) Chichester,Ny Wiley, 1985*

[Gross93] T. Grossman and A.Jagota. *On the equivalence of two Hopfield-type networks. Proc. of IEEE on Neural Networks, pp.1063-1068, 1993*

[Hertz91] J. Hertz, A. Krogh, and R.G. Palmer. *Introduction to the Theory of Neural Computation.* Addison-Wesley, 1991.

[Hopf82] J.J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences, USA*, 79, 1982.

[Hopf85] J.J. Hopfield and D.W. Tank 1985."Neural" Computation of Decisions in Optimization Problems// *Biol.Cybernetics,52,141-152*

[Jag91] A. Jagota. Backtracking Dynamics for a Hopfield-Style Network *International Joint Conference on Neural Networks*, pages 156-162, 1991. Seatle, June, IEEE.

[Jag92] A. Jagota. Efficiently approximating Max-Clique in a Hopfield-style network. *International Joint Conference on Neural Networks*, volume 2, pages 248–253, New York, June 1992. Baltimore, June, IEEE.

[Jag94] A.Jagota . Representing Discrete Structures in Hopfield-Style Network *//In Neural networks for knowledge representation and inference, Lawrence Erlbaum Associates, Publishers, 1994.*

[Jag95] A.Jagota . Approximating Maximum Clique with Hopfield Networks, *//IEEE Transactions on Neural networks, Vol.6, No 3, pp 724-735, 1994.*

[Kondr95] G. Kondrak and P. van Beek. A theoretical evaluation of selected backtracking algorithms. *Proceedings of the 14th International Joint Conference on Artificial Intelligence, Montreal, Quebec, 541-547, August, 1995.*

[Mac92] A.K. Mackworth. Constraint satisfaction. In S.C. Shapiro, editor, *Encyclopedia of Artificial Intelligence, Second Edition*, pages 285–290, New York, 1992. John Wiley and Sons.

[Rum86] Rumelhart, David E. *Ch 14, Vol.2, of Parallel distributed processing: explorations in the microstructure of cognition. Computational models of cognition*

[Soisc90] R. Sosic, J. Gu, A Polynomial Time Algorithm for the N-Queens Problem. *SIGART Bulletin, Volume 1, Number 3, October 1990.*

[Ueber93] J.Ueberla, A.Jagota The Hybrid Hopfiled-clique Memory with Perfect Storage *Proc. of IEEE on Neural Networks, pp.895-899, 1993*

[Xu91] Xu,X. and Tsai W. 1991. Effective Neural Algorithms for the Travelling Salesman Problem//*Neural Networks,4,p.193-205, 1991*