DEUTSCHES ELEKTRONEN – SYNCHROTRON DESY

DESY 90-048
May 1990

Multi-State Spin Glasses, Neural Networks and Generalized Subgraph Polynomials

D. Graudenz
II. Institut für Theoretische Physik, Universität Hamburg

ISSN 0418-9833

NOTKESTRASSE 85 · 2 HAMBURG 52

DESY 90-048
May 1990

ISSN 0418-9833

Multi - State Spin Glasses, Neural Networks and Generalized Subgraph Polynomials

Dirk Graudenz[*]

II. Institut für Theoretische Physik, Universität Hamburg

D-2000 Hamburg 50, Luruper Chaussee 149

FRG

May 1990

Abstract

For each finite graph G we define a polynomial $f_G(p,q)$ depending on the number of edges and number of components of all subgraphs of G. We derive a formula for a generating function Z of these subgraph polynomials. Z turns out to be the canonical partition function of a spin glass with p states and an S_p symmetry. For a specific choice of the couplings we derive a generalization of the Hopfield model that can be viewed as a neural network with more than 2 states per neuron. Finally the system is investigated numerically and some results for the associative properties of this new network type are presented.

*) Supported by Studienstiftung des deutschen Volkes

1. Introduction

Neural networks (for an overview, see [1]) are presently discussed as possible candidates for associative memories that could be capable of reconstructing the whole information from a part of a pattern. The Hopfield model [2] has received much interest because of its similarity to the Ising spin model. Using a simple learning rule that goes back to Hebb [3], it is possible to store information in the network and retrieve it afterwards [4].

The Hopfield model essentially uses neurons with two states ± 1. In this article we are going to discuss a generalization of this model. We allow for more than two states per neuron. Every neuron can be in one of p states. These p states are equivalent in the sense that the system has an S_p symmetry. States that differ only in a permutation of the p symbols have the same energy and therefore have a similar dynamical behaviour.

The outline of this paper is as follows. In the second section we consider a graph theoretical problem. We assign polynomials in two variables to finite graphs. The polynomials are connected to a counting problem in graph theory: How many subgraphs of a given graph with a fixed number of edges and components exist? We construct a generating function Z for these subgraph polynomials. In the third section we recognize that Z is essentially the canonical partition function of a spin glass, where the coupling constants of the spin glass are the parameters of the generating function. The subgraph polynomials can be obtained by differentiation of Z with respect to the parameters, and these derivatives of Z are connected to the correlation functions of the spin glass, such that a high-temperature expansion can be expressed in terms of subgraph polynomials. Then in section 4 we turn to a non-perturbative application of this physical system. By generalizing Hebb's learning rule, we arrive at a model of a neural network that allows storage and retrieval of patterns with more than two symbols. We present some results of a numerical investigation of such a network. We show that indeed patterns that are stored can be retrieved. We finally demonstrate that in a certain sense our network can show superposition of patterns.

2. Generalized Subgraph Polynomials

In this section we introduce the subgraph polynomials and a generalization of them. Then we discuss the generating function of these polynomials. We will see that the generalized subgraph polynomials of subgraphs of a graph may be obtained by differentiating its generalized subgraph polynomial with respect to certain parameters that label the edges of the graph.

Definition: An *undirected graph* G is a pair (V_G, ε_G), where V_G is the set of vertices of G, and

$$\varepsilon_G : \mathcal{B}_G \longrightarrow \mathbb{N} \tag{2.1}$$

is the map that determines the multiplicities of the edges. Here

$$\mathcal{B}_G := \left\{ \{a,b\} \;\middle|\; a,b \in V_G \right\} \tag{2.2}$$

is the set of unordered pairs of vertices. If

$$V_G = \underline{n} := \left\{ 1, 2, \ldots, n \right\}, \tag{2.3}$$

then we write $\mathcal{B}_G = \mathcal{B}_n$.

A *subgraph* H of G, $H \leq G$, is a graph (V_H, ε_H) with $V_H = V_G$ and $\varepsilon_H \leq \varepsilon_G$, that is, $\varepsilon_H(k) \leq \varepsilon_G(k)$ for all $k \in \mathcal{B}_G = \mathcal{B}_H$.

For a graph G and a subgraph $H \leq G$ we define the multiplicity m(H,G) by

$$m(H,G) := \prod_{k \in \mathcal{B}_G} \binom{\varepsilon_G(k)}{\varepsilon_H(k)}. \tag{2.4}$$

If G is a graph, then we define $\flat G$ to be the number of components of G and $\sharp G$ to be the number of edges of G, so that

$$\sharp G = \sum_{k \in \mathcal{B}_G} \varepsilon_G(k). \tag{2.5}$$

<u>Example</u>:

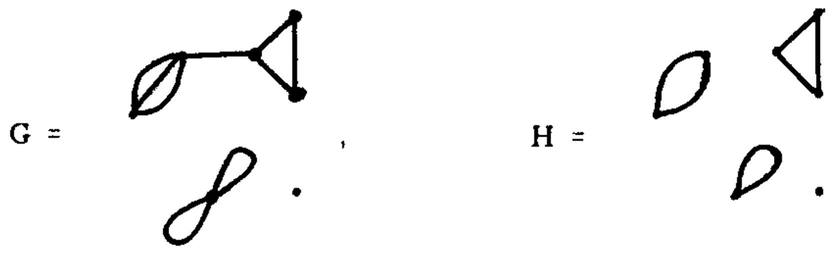

Figure 2.1

The multiplicity of the graphs in Fig. 2.1 is

$$m(H,G) = \binom{3}{2} \cdot \binom{2}{1} = 6, \tag{2.6}$$

and $\flat H = 4$, $\sharp H = 6$.

Definition: If G is a graph, then the *subgraph polynomial* (SGP) $f_G(p,q) \in K(p,q)$ is defined by [5]

$$f_G(p,q) := \sum_{H \le G} m(H,G)\ p^{bH}\ q^{\# H}. \tag{2.7}$$

To be definite, we set $K = \mathbb{C}$ in the sequel. We note that the coefficient of $p^b q^\#$ is equal to the number of subgraphs of a graph with b components and $\#$ edges (subgraphs are counted with multiplicity). Therefore the calculation of $f_G(p,q)$ is equivalent to the solution of a counting problem. Now we generalize the subgraph polynomials to obtain an algebraic expression for them, and later will will see that these expressions are the building blocks of a generating function for the SGP's. But first we have to define some notations.

Definition: Let F(A,B) be the set of maps from A to B. For $p \in \mathbb{N}$, an element

$$\alpha \in F(\underline{n}, \underline{p}) =: S \tag{2.8}$$

will be called a state (the reason for this name will become clear later). Since n and p will be fixed, for ease of writing we denote the set of states by S.

Definition: For $\lambda \in F(\mathcal{B}_G, \mathbb{C})$, we define the *generalized subgraph polynomial* (GSGP) of a graph G by

$$f_G(p,q,\lambda) := \sum_{H \le G} m(H,G)\ p^{bH}\ q^{\# H} \prod_{k \in \mathcal{B}_G} \lambda_k^{[\varepsilon_G(k) - \varepsilon_H(k)]} . \tag{2.9}$$

Now we derive an explicit expression for $f_G(p,q,\lambda)$.

<u>Definition</u>: For a state $\alpha \in S$ and $k \in \mathcal{B}_G$, let $\delta(\alpha,k) := \delta_{\alpha(a)\,\alpha(b)}$, where δ_{ij} is the Kronecker-δ and $k = \{a,b\}$.

<u>Theorem</u>: For a graph $G = (\underline{n},\ \varepsilon_G)$ define

$$h_G(p,q,\lambda) := \sum_{\alpha \in F(\underline{n},\underline{p})} \ \prod_{k \in \mathcal{B}_n} \left(q\,\delta(\alpha,k) + \lambda_k \right)^{\varepsilon_G(k)}. \tag{2.10}$$

Then $f_G(p,q,\lambda) = h_G(p,q,\lambda)$.

<u>Proof</u>: We expand the binomial in the product and obtain

$$\left(q\,\delta(\alpha,k) + \lambda_k \right)^{\varepsilon_G(k)} = \sum_{i=0}^{\varepsilon_G(k)} \binom{\varepsilon_G(k)}{i} q^i \lambda_k^{\varepsilon_G(k)-i}\,\delta(\alpha,k)^i. \tag{2.11}$$

If we now multiply these terms for all $k \in \mathcal{B}_n$, we see that each summand can be identified with a subgraph H of G, namely, we set $\varepsilon_H(k) := i$, if the i^{th} term in the expansion of the binomial is a factor in the product. Therefore we obtain

$$\prod_{k \in \mathcal{B}_n} \left(q\,\delta(\alpha,k) + \lambda_k \right)^{\varepsilon_G(k)}$$

$$= \sum_{H \leq G} m(H,G) \prod_{k \in \mathcal{B}_n} q^{\varepsilon_H(k)}\,\lambda^{\varepsilon_G(k)-\varepsilon_H(k)}\,\delta(\alpha,k)^{\varepsilon_H(k)}. \tag{2.12}$$

It remains to perform the sum over the states α. The subgraphs H of G split into components. For each of these compents C, the $\delta(\alpha,k)$'s restrict the sum of all α_i for the vertices i that belong to C to one value, so that the sum over the states contributes a factor of p for each component. Finally we arrive at

$$h_G(p,q,\lambda) := \sum_{H\leq G} m(H,G)\, p^{bH}\, q^{\#H} \prod_{k\in\mathcal{B}_n} \lambda_k^{[\varepsilon_G(k)-\varepsilon_H(k)]} , \tag{2.13}$$

which proves $f_G(p,q,\lambda) = h_G(p,q,\lambda)$.

∎

$h_G(p,q,\lambda)$ is defined for $p\in\mathbb{N}$. But since there is a unique polynomial in p for $p\in\mathbb{C}$ that coincides with $h_G(p,q,\lambda)$ if $p\in\mathbb{N}$, we see that the calculation of h_G suffices to determine f_G for all $p\in\mathbb{C}$.

<u>Definition</u>: We normalize the generalized subgraph polynomials by

$$g_G(p,q,\lambda) := \Big[\prod_{k\in\mathcal{B}_G} \frac{1}{\varepsilon_G(k)!} \Big] h_G(p,q,\lambda). \tag{2.14}$$

<u>Definition</u>: If $k\in\mathcal{B}_G$ is an edge of G, that is, $\varepsilon_G(k)>0$, then we define the graph G-k by (V_G,ε'), where $\varepsilon'(l) = \varepsilon_G(l) - \delta_{kl}$. The graph $G+k = (V_G, \varepsilon'')$ is defined by $\varepsilon''(l) = \varepsilon_G(l) + \delta_{kl}$.

Now we regard the λ_k as parameters of the polynomial $g_G(p,q,\lambda)$.

Theorem: If k is an edge of G, then

$$g_{G-k}(p,q,\lambda) = \frac{\partial}{\partial\lambda_k} g_G(p,q,\lambda). \tag{2.15}$$

Proof: This is obvious from the definition of $h_G(p,q,\lambda)$ and the normalization for $g_G(p,q,\lambda)$.

■

Therefore the GSGP for a graph G can be calculated by multiple differentiation with respect to the λ_k of the GSGP of any graph G' with $G \leq G'$. If $\mathbf{1}$ is the constant map $\mathbf{1}_k = 1$, then we see that $f_G(p,q) = f_G(p,q,\mathbf{1})$.

By introducing new parameters $\mu \in F(\mathcal{B}_n, \mathbb{C})$ we can construct the generating function $Z_n(p,q,\lambda,\mu)$ for the GSGP's.

Defintion: Denote by $\mathcal{G}_n$ the set of all graphs G whose vertex set is $\underline{n}$. For $p, q \in \mathbb{C}$ and $\lambda, \mu \in F(\mathcal{B}_n, \mathbb{C})$ define

$$Z_n(p,q,\lambda,\mu) := \sum_{G \in \mathcal{G}_n} \frac{\mu^{\varepsilon_G}}{\varepsilon_G!} f_G(p,q,\lambda). \tag{2.16}$$

Here we used the shorthand notation

$$\mu^{\varepsilon_G} := \prod_{k \in \mathcal{B}_G} \mu_k^{\varepsilon_G(k)}, \quad \varepsilon_G! := \prod_{k \in \mathcal{B}_G} \varepsilon_G(k)!. \tag{2.17}$$

For a multiindex $\nu \in F(\mathcal{B}_n, \mathbb{N})$ and a map $\varkappa \in F(\mathcal{B}_n, \mathbb{C})$ we define

$$\partial_\varkappa^\nu := \prod_{k \in \mathcal{B}_G} \left(\frac{\partial}{\partial \varkappa_k} \right)^{\nu_k} \quad \text{and} \quad |\nu| := \sum_{k \in \mathcal{B}_G} \nu(k). \tag{2.18}$$

The generating function $Z_n(p,q,\lambda,\mu)$ has the property that

$$f_G(p,q,\lambda) = \partial_\mu^{\varepsilon_G}\Big|_{\mu=0} Z_n(p,q,\lambda,\mu). \tag{2.19}$$

By setting λ equal to $\mathbb{1}$ we obtain

$$f_G(p,q) = \partial_\mu^{\varepsilon_G}\Big|_{\mu=0} Z_n(p,q,\mathbb{1},\mu). \tag{2.20}$$

<u>Theorem</u>: For $p \in \mathbb{N}$, we have

$$Z_n(p,q,\lambda,\mu) = \sum_{\alpha \in F(\underline{n},\underline{p})} \exp\left[A_n(\alpha,q,\lambda,\mu) \right], \tag{2.21}$$

where

$$A_n(\alpha,q,\lambda,\mu) := \sum_{k \in \mathcal{B}_n} \mu_k \big(q\,\delta(\alpha,k) + \lambda_k \big). \tag{2.22}$$

Proof: By inserting the definition of $h_G(p,q,\lambda)$ into the definition of $Z_n(p,q,\lambda,\mu)$ we obtain

$$Z_n(p,q,\lambda,\mu) := \sum_{G\in\mathcal{G}_n} \frac{\mu^{\varepsilon_G}}{\varepsilon_G!} \sum_{\alpha\in F(\underline{n},\underline{p})} \prod_{k\in\mathcal{B}_n} \left(q\,\delta(\alpha,k) + \lambda_k \right)^{\varepsilon_G(k)}$$

$$= \sum_{\alpha\in F(\underline{n},\underline{p})} \sum_{\varepsilon\in F(\mathcal{B}_n,\mathbb{N})} \prod_{k\in\mathcal{B}_n} \frac{\mu_k^{\varepsilon(k)}}{\varepsilon(k)!} \left(q\,\delta(\alpha,k) + \lambda_k \right)^{\varepsilon(k)}$$

$$= \sum_{\alpha\in F(\underline{n},\underline{p})} \exp\Big[\sum_{k\in\mathcal{B}_n} \mu_k \big(q\,\delta(\alpha,k) + \lambda_k \big) \Big]. \tag{2.23}$$

■

Therefore we succeeded in constructing a generating function of the subgraph polynomials. We note that it is a finite sum of exponentials whose argument has a simple structure.

3. Multi - State Spin Glasses

In this section we define a physical system whose canonical partition function coincides with the generating function $Z_n(p,q,\lambda,\mu)$ for the GSGP's. If a system is described by some parameters λ, μ, then the canonical partition function at inverse temperature $\beta = 1/(kT)$ (k is the Boltzmann constant) is given by

$$\underline{Z}(\beta,\lambda,\mu) = \sum_{\alpha\in S} \exp\Big[-\beta\, H(\alpha,\lambda,\mu) \Big]. \tag{3.1}$$

Here we assumed that the set of possible states S of the system is discrete and the Hamilton function (the energy) is H. We will assume that our system consists of n components, each of which can be in p different states. Therefore the state of the system is given by an $\alpha \in F(\underline{n},\underline{p})$. The Hamilton function H then describes the interactions between the components. We assume that the interaction energy arising from the components i and j is

$$H_k(\alpha) = -\Big[\rho_k\,\delta(\alpha,k) + \sigma_k\big(1-\delta(\alpha,k)\big)\Big] \;;\quad k := \{i,j\}. \tag{3.2}$$

This means that the interaction energy is $-\rho_k$, if the state of component i is equal to the state of component j, and $-\sigma_k$, if the states of the two components are not equal. The Hamilton function of the system is then given by the sum

$$H(\alpha) = \sum_{k\in\mathcal{B}_n} H_k(\alpha) \tag{3.3}$$

and the partition function is

$$\underline{Z}(\beta,\rho,\sigma) = \sum_{\alpha\in S} \exp\Big[\beta \sum_{k\in\mathcal{B}_n}\big(\rho_k\,\delta(\alpha,k) + \sigma_k\,(1-\delta(\alpha,k))\big)\Big]. \tag{3.4}$$

It is easy to see that

$$Z_n(p,q,\lambda,\mu) = \underline{Z}(\beta, \frac{\mu(\lambda+q)}{\beta}, \frac{\mu\lambda}{\beta}). \tag{3.5}$$

Therefore, Z_n corresponds to the partition function of a certain physical system that turns out to be a spin glass that is generalized to the case of more than two states of every component. We note that our system has an S_p symmetry: Given a state $\alpha \in S$ and a permutation $\pi \in S_p$, then the state $\pi \circ \alpha$ has the same energy as the state α. The set of states S can be divided into equivalence classes: $S' := S/S_p$, where $\alpha \sim \pi \circ \alpha$, $\pi \in S_p$. It is obviously desirable that the observables respect this symmetry. The observables we decide to discuss are *correlations* $\langle \nu, \nu' \rangle$.

<u>Definition</u>: If $\nu, \nu' \in F(\mathcal{B}_n, \{0, 1\})$, let

$$\delta^{\nu}(\alpha) := \prod_{k \in \mathcal{B}_n} \left(\delta(\alpha,k) \right)^{\nu(k)}, \tag{3.6}$$

$$(1-\delta)^{\nu'}(\alpha) := \prod_{k \in \mathcal{B}_n} \left(1 - \delta(\alpha,k) \right)^{\nu'(k)}, \quad \text{and} \tag{3.7}$$

$$\langle \nu, \nu' \rangle := \sum_{\alpha \in S} \delta^{\nu}(\alpha) \cdot (1-\delta)^{\nu'}(\alpha) \cdot \exp\Big[\beta \sum_{k \in \mathcal{B}_n} \Big(\rho_k \, \delta(\alpha,k) + \sigma_k \left(1 - \delta(\alpha,k) \right) \Big) \Big]. \tag{3.8}$$

We note that our correlations are not normalized, so in general $\langle 0,0 \rangle \neq 1$.

It is easy to see that

$$\langle \nu, \nu' \rangle = \frac{1}{\beta^{|\nu|+|\nu'|}} \, \partial_{\rho}^{\nu} \, \partial_{\sigma}^{\nu'} \, \underline{Z}(\beta,\rho,\sigma). \tag{3.9}$$

λ, μ, ρ and σ are related by

$$\rho = \frac{\mu(\lambda+q)}{\beta}, \quad \sigma = \frac{\mu\lambda}{\beta}. \tag{3.10}$$

It follows that

$$\mu_k \frac{\partial}{\partial \mu_k} = \rho_k \frac{\partial}{\partial \rho_k} + \sigma_k \frac{\partial}{\partial \sigma_k}\text{, or} \tag{3.11}$$

$$\frac{\partial}{\partial \ln \mu_k} = \frac{\partial}{\partial \ln \rho_k} + \frac{\partial}{\partial \ln \sigma_k}. \tag{3.12}$$

A perturbative evaluation of the generating function $Z_n(p,q,\lambda,\mu)$ with respect to the coupling constants λ and μ can be expressed in terms of the GSGP's. This means that by (3.12) a high-temperature expansion of the correlation functions of a spin glass is essentially a sum over the f_G's, which are the values of the "Feynman graphs" G. We will not pursue perturbation theory further, but now turn to a numerical investigation of the system when it is interpreted as a neural network.

4. A Generalization of the Hopfield Model

A very popular model of a neural network is the Hopfield model [2]. It consists of n neurons $S_i \in \{-1,1\}$, $i=1, \ldots, n$. The energy of the state S is given by

$$H(S) = -\frac{1}{2} \sum_{i,j} J_{ij} S_i S_j, \tag{4.1}$$

where the J_{ij} are the coupling constants. Given a state S at time t, the state S' at time t+1 is determined in the following way. Choose a random neuron k. Calculate

$$m := \sum_{j=1}^{n} J_{kj} S_j, \tag{4.2}$$

and set $S'_k := \operatorname{sgn} m$, with all other neurons unaffected. One can show that this rule enforces $H(S') \le H(S)$, and equality holds only if the state doesn't change. So there are no cycles possible, and the system finally will settle down in a stable state with unit probability.

In the simplest case, the couplings J_{ij} are given by the Hebb rule [3]. Assume there are M patterns $\xi_{\mu i}$, $\mu = 1, \ldots, M$ that are to be stored in the network. The J_{ij} are then given by

$$J_{ij} := \frac{1}{n} \sum_{\mu=1}^{M} \xi_{i\mu}\, \xi_{j\mu}, \text{ if } i \neq j, \text{ and } J_{ii} := 0. \tag{4.3}$$

This means that the coupling between neuron i and neuron j is large if there is a positive correlation in the patterns with respect to the neurons i and j. One hopes that such a network may serve as an associative memory. An arbitrary pattern S is chosen, and the rule for the time evolution is applied until the system does not change its state any longer. Intuitively, the system will reach the nearest stable state that is similar to the original input pattern S. For random patterns, the Hebb rule is a good choice, but for regular patterns, there are better rules that involve the Moore-Penrose pseudo inverse of a matrix [6].

We generalize the model in the following way. We assume that our model neurons $\alpha_1, \ldots, \alpha_n$ can be in p possible states, so that a state of the system is given by an $\alpha \in F(\underline{n}, \underline{p})$. The energy is defined by

$$H(\alpha) := -\sum_{i,j} J_{ij}\, \delta_{\alpha_i \alpha_j}. \tag{4.4}$$

We generalize the Hebb rule by

$$J_{ij} := \frac{1}{n} \sum_{\mu=1}^{M} \delta_{\xi_{\mu i} \xi_{\mu j}} \text{ if } i \neq j, \text{ and } J_{ii} := 0, \tag{4.5}$$

where $\xi_{1i}, \ldots, \xi_{Mi}$ are M input patterns, $\xi_{\mu i} \in \underline{p}$. The update rule should take into account the correlation of all other neurons with respect to the chosen neuron k. So given a state α, we choose a random neuron k. We calculate the numbers

$$m_\beta := \sum_{j=1}^{n} J_{kj} \, \delta_{\beta,\alpha_j} \tag{4.6}$$

for $\beta \in \underline{p}$. Then we set α'_k to that β with the highest m_β and leave all other neurons unaffected. This rule means that the randomly chosen neuron k is pushed into the state that has, according to the couplings, the highest correlation. One can show that H never increases, so with unit probability, the system will eventually reach a stable state.

Finally we describe some computer experiments to show that this model has some interesting features. We used a 10*10 lattice and investigated systems with two and three states.

In the first experiment we stored the four patterns of fig. 4.1 in the network. We used two regular patterns that will be used later and two random patterns that we added to see the influence of additional information that is stored.

```
* * * . . . . 2 2 2      * * * * * * * * * *
* * * . . . . 2 2 2      * * * * * * * * * *
* * * . . . . 2 2 2      * * * * * * * * * *
* * * . . . . 2 2 2      . . . . . . . . . .
* * * . . . . 2 2 2      . . . . . . . . . .
* * * . . . . 2 2 2      . . . . . . . . . .
* * * . . . . 2 2 2      . . . . . . . . . .
* * * . . . . 2 2 2      2 2 2 2 2 2 2 2 2 2
* * * . . . . 2 2 2      2 2 2 2 2 2 2 2 2 2
* * * . . . . 2 2 2      2 2 2 2 2 2 2 2 2 2

2 . 2 . * * * . * .      * . * . . * . . . .
2 2 * * 2 2 2 2 2 2      . 2 . 2 2 * . . * .
. . 2 2 * * 2 2 . .      * 2 * . 2 2 * * 2 2
. . * . * * . * * 2      . * . . * 2 . . . *
* . * . * * 2 2 . 2      2 * . . * . * * * *
* * . . * 2 * * 2 .      2 2 * . 2 . . . * .
. 2 * 2 2 2 . * 2 *      . 2 . 2 * * 2 2 * 2
2 2 2 2 . * 2 * * 2      2 2 2 . . * . * * 2
2 . . . 2 . * 2 * 2      * . 2 2 2 . . * 2 .
. 2 * . . * * * 2 2      . * 2 . * . 2 2 2 2
```

Figure 4.1: Patterns with three states that were stored in the network

Then we used the pattern of fig. 4.2 (a) as an input and updated the network asynchronously 10 times. The system settled down into the final state of fig. 4.2 (b). In the next experiment we used the pattern of fig. 4.3 (a) as an input. Again, after ten total updates, a stored pattern was recognized (fig. 4.3 (b)).

```
* * . . 2 2 2 2 * 2        * * * . . . . 2 2 2
* 2 * . . . . 2 2 2        * * * . . . . 2 2 2
* . * . . * . 2 . 2        * * * . . . . 2 2 2
* * * . . . . 2 2 2        * * * . . . . 2 2 2
* . * . * . . . . .        * * * . . . . 2 2 2
2 2 * . . . . 2 * 2        * * * . . . . 2 2 2
* * . . . * * 2 2 2        * * * . . . . 2 2 2
* . * . . . . 2 * 2        * * * . . . . 2 2 2
* * * . . . . 2 2 2        * * * . . . . 2 2 2
* 2 2 * * . . 2 . .        * * * . . . . 2 2 2

        (a)                        (b)
```

Figure 4.2: (a) input pattern, (b) retrieved pattern.

```
2 2 2 . 2 2 2 * * .        . . . 2 2 2 2 * * *
. . . . 2 * 2 * 2 *        . . . 2 2 2 2 * * *
. . * . 2 2 2 * * *        . . . 2 2 2 2 * * *
. . . . . . 2 * * 2        . . . 2 2 2 2 * * *
. * . . 2 2 2 * * *        . . . 2 2 2 2 * * *
. . . . 2 2 2 * * *        . . . 2 2 2 2 * * *
. . . . 2 * * . . .        . . . 2 2 2 2 * * *
. * * . 2 2 2 * * *        . . . 2 2 2 2 * * *
. . . . 2 2 2 * . *        . . . 2 2 2 2 * * *
2 . * * 2 2 2 * * *        . . . 2 2 2 2 * * *

        (a)                        (b)
```

Figure 4.3

We note, however, that we have permuted the symbols in the input pattern, so that the output pattern shows this permutation, too. Since the system has an S_p symmetry, all input patterns that are permutations of each other are equivalent. We see that our system can recognize patterns that are built up with more than just two symbols, as is usually discussed. Therefore it makes sense to use neurons with more than two states. Of course, in our simple system, there is no ordering relation that specifies an order of the symbols. This means that it is only important if two symbols are equal or not, it is not possible to assign weights to the symbols.

In another experiment we wanted to see what happens if patterns with two symbols are stored in a network, but the dynamics is then applied such as if there were three symbols available. This means that after storing the information we enlarge the phase space of the system. We stored the three patterns of fig. 4.4.

```
. . . * * * * * . *     * * . . . . . . * *     . . . * * * * . . .
* * . * . * . * * .     * * * . . . . * * *     . . . * * * * . . .
. * . . * * * . . .     . * * * . . * * * .     . . . * * * * . . .
. * . . * . . * . .     . . * * * * * * . .     . . . * * * * . . .
. . * . . . * * * *     . . . * * * * . . .     . . . . . . . . . .
. * * * . . . * . *     . . . * * * * . . .     . . . . . . . . . .
. . . * . . . . * *     . . * * * * * * . .     . . . . . . . . . .
. . . * . . . * * *     . * * * . . * * * .     . . . . . . . . . .
. * . . * . * . * *     * * * . . . . * * *     . . . . . . . . . .
. . . . * * . . * *     * * . . . . . . * *     . . . . . . . . . .
```

Figure 4.4

We used the pattern of fig. 4.5 (a) as an input and applied the dynamical rules for two states. The result after five complete updates is shown in fig. 4.5 (b). The stored pattern is recognized, as is expected from the Hopfield model.

```
* . . . * * * . . *     * * . . . . . . * *
. * . * * . * . * *     * * * . . . . * * *
. . * * * * * . * .     . * * * . . * * * .
. . . * . * * * . .     . . * * * * * * . .
. . . . * * . . . .     . . . * * * * . . .
. . * * * * * * . .     . . . * * * * . . .
. . * * . . * * * .     . . * * * * * * . .
. * * . . . . * * .     . * * * . . * * * .
* * * . . . . * * *     * * * . . . . * * *
* * . . . . . . * *     * * . . . . . . * *

        (a)                     (b)
```

Figure 4.5

Then we stored the pattern of fig. 4.6 (a) in the network, but now we applied the dynamical rules as if three possible states were available. The final state after five updates is shown in fig. 4.6 (b). Since the input pattern is so close to a pattern that is stored in the network, nothing new happens.

```
, , , , * * * , , ,      , , , * * * * , , ,
, , , * * , * , , ,      , , , * * * * , , ,
, , , * * * * , , ,      , , , * * * * , , ,
, , , * , * * , , ,      , , , * * * * , , ,
, , , , , , , , , ,      , , , , , , , , , ,
, * , , , , , , , ,      , , , , , , , , , ,
, , , * , , * , , ,      , , , , , , , , , ,
, , , , , , , , , ,      , , , , , , , , , ,
, , , , , , , , * ,      , , , , , , , , , ,
, , , , , , , , , *      , , , , , , , , , ,

        (a)                      (b)
```

Figure 4.6

Now we used a pattern that is close to two input patterns as an input (fig. 4.7 (a)).

```
* , , , * * * , , *      * * , 2 2 2 2 , * *
, * , * * , * , * *      * * * 2 2 2 2 * * *
, , * * * * * , * ,      , * * * 2 2 2 * * ,
, , , * , * * * , ,      , , * * 2 * * * , ,
, , , , * * , , , ,      , , , * * * * , , ,
, , * * * * * * , ,      , , , * * * * , , ,
, , * * , , * * * ,      , , * * * * * * , ,
, * * , , , , * * ,      , * * * , , * * * ,
* * * , , , , * * *      * * * , , , , * * *
* * , , , , , , * *      * * , , , , , , * *

        (a)                      (b)
```

Figure 4.7

After five complete updates, the system found a stable state that looks like that in fig. 4.7 (b). We see that the final state looks as if it consists of a cross and a small square, both which are learned patterns of the network. *The system uses the enlarged phase space to occupy a state that is close to two input states. The resulting patterns are superpositions of two input patterns.*

5. Summary and Conclusions

We started with a graph theoretical problem and constructed the generating function for a class of polynomials that arise in graph theory. This generating function is the partition function of a spin glass with p states and an S_p symmetry. The subgraph polynomials are related to correlations in this statistical mechanical system. We give a practical application of this model. A specific choice of the couplings is motivated by the Hebb rule and results in a generalization of the Hopfield model. Finally we present some numerical experiments and show that complex patterns with more than two symbols can be stored and retrieved.

6. Acknowledgement

I gratefully acknowledge discussions with S. Bornholdt and G. Münster and would like to thank G. Kramer for reading the manuscript.

7. References

[1] E. Domany, Conference on the Mathematical Aspects of Non Equilibrium Physics, 1987, ITP Santa Barbara

[2] J.J. Hopfield, Proceedings of the National Academy of Sciences 79 (1982) 2554

[3] D.O. Hebb, The Organization of Behaviour (Wiley, New York 1949)

[4] W. Kinzel, J. Phys. B60 (1985) 205

[5] D. Graudenz, DESY preprint 89-139, to be published in J. Math. Phys.

[6] L. Personnaz, I. Guyon, G. Dreyfus, Phys. Rev. A34 (1986) 4217