Abstract: Problems of combinatorial optimization, beyond their interest in applied research, play a crucial role in fundamental issues of theoretical computer science, for their inherent computational complexity. Here we use them as test bed on which to gauge the many perspectives and problems offered by neural networks.

The realization that optimization problems for quadratic functions of many Boolean variables which are, in a technical sense to be made precise, as difficult as they can be, are conveniently dealt with by neural networks contributes to the interest of such dynamical systems: the parameters controlling their evolution can indeed be assigned in such a way that they have precisely the function to be minimized as a Lyapunov function. The recognition that such an evolution will, in general, stop in a local minimum of this Lyapunov function, as opposed to the global minima one is searching for, motivates the idea of endowing the dynamics of a neural network with a stochastic transition rule leading to a stationary distribution strongly peack after global minima.

Here we discuss several problems related to the dynamics of both deterministic and stochastic networks with an emphasis on the problem of quantitatively assessing their computational capabilities.

1. Computational Complexity

In this section we review a few notions and facts of life pertinent to the theory of computational complexity, without any pretense of rigour or self-containedness. The interested reader is referred to the monographs by Garey and Johnson [Ga79] or by Stockmeyer [St87] for more precise statements.

Solving a problem can mean providing a Yes/No answer (decision problem) or, more generally, evaluating a function (functional problem). In this paper we are interested in a particular class of functional problems, i.e. combinatorial optimization problems.

A combinatorial optimization problem Π (say, the one informally stated as "finding the minimum energy of a two dimensional antiferromagnetic array of Ising spins in a magnetic field") is determined by:
(1) A set $D_{\Pi}$ of instances (for the example alluded to, an instance $I \in D_{\Pi}$ would be specified by an integer $n$, a set $\Lambda \subseteq \mathbb{Z}^2$ with $|\Lambda| = n$, a subset $E$ of $\Lambda \times \Lambda$ specifying which pairs of spins are actually coupled);

(2) For each instance $I \in D_{\Pi}$, a finite set $S_{\Pi}(I)$ of candidate solutions for this instance (in our example $S_{\Pi}(I)$ would be the set of functions $s : i \in \Lambda \rightarrow s_i \in \{-1,1\}$ giving the value of the spin sitting at each site $i \in \Lambda$);

(3) A function $H_{\Pi}$ that assigns to each instance $I \in D_{\Pi}$ and to each candidate solution $s \in S_{\Pi}(I)$ a rational number $H_{\Pi}(I,s)$, called the solution value for $s$ (in our case, say, $H_{\Pi}(I,s) = \sum_{(i,j) \in E} s_i s_j + \sum_{i \in \Lambda} s_i$).

In a minimization problem an optimal solution for an instance $I \in D_{\Pi}$ is a candidate solution $s' \in S_{\Pi}(I)$ such that, for all $s \in S_{\Pi}(I)$, $H_{\Pi}(I,s') \leq H_{\Pi}(I,s)$.

To every optimization problem a decision problem can be associated in a natural way; it can be posed in the following way: given the instance $(I,k)$, where $I \in D_{\Pi}$, and $k$ is a rational number, does there exist $s \in S_{\Pi}(I)$ such that $H_{\Pi}(I,s) \leq k$?

In order to solve a problem with a machine, one must be able to estimate the amount of resources (say time or memory space) which must be spent to obtain the answer.

The fact that a problem is "technically" solvable, namely that there is an algorithm which for each instance provides the answer, does not necessarily mean that it is "practically" solvable: one often faces situations in which any exact solving algorithm requires an amount of resources rising so sharply with the size of the instance as to make it practically unfeasible to search for exact solutions. It may be wise to realize a priori that this is the situation for a given optimization problem and concentrate, instead, every effort on the more realistic task of searching for good approximate solutions (say, look for low local minima instead of looking for global minima by an exhaustive enumeration which might require many times the age of the universe).

We sketch below a few notions relevant to a quantitative measure of the notion of "practically solvable or unsolvable" vaguely given above. For definiteness sake we focus on decision problems (thus giving at least lower bounds on the "difficulty" of the optimization problems) and refer to the computational model provided by deterministic Turing machines (referring to your PC or to a state of the art mainframe would not change the picture in any essential respect).

To state the decision problem $\Pi$ in such a way that Turing machines can work on it, it is necessary first of all to codify the instances over some "suitable" alphabet $\Sigma$: for our prototype Ising decision problem any fixed reasonable binary description of the numbers $n$ and $k$ and of the incidence matrix of the graph $(\Lambda,E)$ will do, so that in such a case each instance $(n,\Lambda,E,k)$ is easily encoded by a finite sequence $x$ of