Construction and VHDL Implementation of a Fully Local Network with Good Reconstruction Properties of the Inputs

Joël Chavas[1], Demian Battaglia[2], Andres Cicuttin[3], and Riccardo Zecchina[4]

[1] ISI Foundation, Viale San Severo 65, I-10133 Torino, Italy
chavas@isiosf.isi.it
[2] SISSA, via Beirut 9, I-34100 Trieste, Italy
battagli@sissa.it
[3] ICTP-INFN MLab Laboratory, via Beirut 31, I-34100 Trieste, Italy
cicuttin@ictp.trieste.it
[4] ICTP, strada della Costiera 11, I-34100 Trieste, Italy
zecchina@ictp.trieste.it

Abstract. This report shows how one can find a solution to the K-SAT equations with the use of purely local computations. Such a local network, inspired by the Survey Propagation equations driven by an external input vector, potentially has an exponential number of attractors. This gives the network powerful classification properties, and permits to reconstruct either noisy or incomplete inputs. It finds applications from bayesian inference to error-correcting codes and gene-regulatory networks, and its local structure is ideal for an implementaion on FPGA. Here we write its algorithm, characterize its main properties and simulate the corresponding VHDL code. One shows that the time of convergence towards a solution optimally scales with the size of the network.

1 Introduction

The fast development of components efficiently performing parallely simultaneous computations (FPGAs, CPLDs) increases the interest of computations making massively use of local parallelism, such as computations on graphs [8].

Belief propagation algorithms are well known algorithms which permit, by the use of purely local messages, to converge towards the exact distribution of *a posteriori* probabilities in tree-like graphicals models, or towards a good approximation even in loopy networks [11].

But when the graph is too constrained, such classical local algorithms fail to converge towards the desired distribution of probabilities.

The graphical model K-SAT (described below), recently studied and solved using techniques of statistical mechanics, can be viewed as a toy-model of such a constrained graph. The Survey Propagation equations are a generalization of the Belief Propagation equations which permit to solve the K-SAT formulas in a range of parameters in which the graph is highly constrained, and in which the Belief Propagation equations fail [9, 10, 4].

J. Mira and J.R. Álvarez (Eds.): IWINAC 2005, LNCS 3562, pp. 385–394, 2005.

But while the Survey Propagation equations use fully local computations, the decimation based on these and which permits to converge to one particular solution of the K-SAT problem uses a global criterion [10, 4], and is as such not suitable for low-level parallelization.

In the present article, one builds a fully local network based on the Survey Propagation equations, the dynamic of which lets it converge towards a desired instance of the K-SAT equations. We then propose an algorithm and its VHDL code, which will permit the implementation of this network on a FPGA. To conclude, we will then shortly review the fields in which such an implementation can be applied.

2 Solving and Properties of the K-SAT Equations

2.1 K-SAT Formulas

The K-SAT formula consists of N Boolean variables $x_i \in \{0,1\}$, $i \in \{1,...,N\}\}$, with M constraints, in which each constraint is a clause, which is the logical OR ($\vee$) of the variables it contains or of their negations. A clause is written as

$$(z_{i_1} \vee ... \vee z_{i_r} \vee ... \vee z_{i_K}) \tag{1}$$

where $z_{i_r} = x_{i_r}$ (resp. $\bar{x}_{i_r}$) if the variable is directed (resp. negated) in the clause. The problem is to find an assignment (if it exists) of the $x_i \in \{0,1\} = \{\text{directed,negated}\}$ which is such that all the M clauses are true. We define the energy E of a configuration $\mathbf{x} = (x_1, ..., x_N)$ as the number of violated clauses.

2.2 The Solution Space Becomes Divided into an Exponential Number of Clusters

When the number of constraints $M = \alpha N$ is small, the solutions of the K-SAT formulas are distributed close one to each other over the whole $N-$dimensional space, and the problem can be solved by the use of classical local search algorithms. When α is included in a narrow region $\alpha_d < \alpha < \alpha_c$, the problem is still satisfiable but the now limited solution phase breaks down in an exponential number of clustered components. Solutions become grouped together into clusters which are fart apart one from the other.

2.3 Survey Propagation Equations with External Inputs

Recently, statistical physics methods derived from spin glass physics (making use of the cavity method) have permitted to derive closed set of equations, which allow, after decimation, the retrieval of a large number of different solutions belonging to different clusters [9, 10, 4, 2, 3].

The principles of the method are schemed on Fig. 1. The message $\eta_{a \to i}$ represents the probability that the clause a sends a warning onto the variable i, *i.e.* the probability that i is forced to be fixed at a value which satisfies a. As the

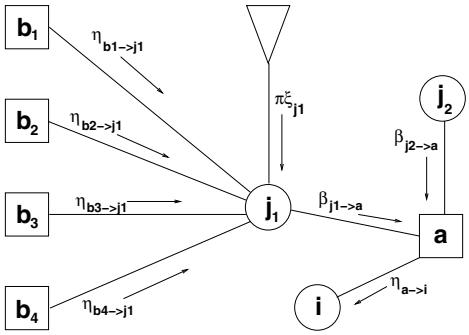

Fig. 1. Factor graph representation of the K-SAT formula. A constraint (called a clause) is represented by a square ; the variables are represented by a circle

clause a is an OR function, it corresponds to the probability that both j_1 and j_2 do not satisfy a (resp. $\beta_{j_1 \to a}$ and $\beta_{j_2 \to a}$). If one assumes independence between j_1 and j_2 when the link (the "cavity") $a - i$ is absent, this writes

$$\eta_{a \to i} = \prod_{j \in V(a) \setminus i} \beta_{j \to a} \ , \tag{2}$$

in which $V(a)$ means "all the variables belonging to the clause a".

One then writes without further derivation the dependency of $\beta_{j \to a}$ (the probability that j does not satisfy a) on the $\eta_{b \to j}$ (see Ref.[4, 10] for further details) :

$$\beta^{\pm}_{j \to a} = \frac{\Pi^{\pm}_{j \to a}}{\Pi^{\pm}_{j \to a} + \Pi^{\mp}_{j \to a} + \Pi^{0}_{j \to a}} \ , \tag{3}$$

$\beta^{+}_{j \to a}$ (resp. $(-)$) meaning that j is negated (directed) in the clause a. And :

$$\begin{aligned} \Pi^{\pm}_{j \to a} = & \left[1 - \lambda \left(1 - \pi \delta_{\xi_j, \pm} \right) \prod_{b \in V_{\pm}(j) \setminus a} (1 - \eta_{b \to j}) \right] \cdot \\ & \cdot \left(1 - \pi \delta_{\xi_j, \mp} \right) \prod_{b \in V_{\mp}(j)} (1 - \eta_{b \to j}) \\ \Pi^{0}_{j \to a} = & \ \lambda \left(1 - \pi \delta_{\xi_j, +} \right) \left(1 - \pi \delta_{\xi_j, -} \right) \prod_{b \in V(j) \setminus a} (1 - \eta_{b \to j}) \end{aligned} \tag{4}$$

in which $V^{+}(j) \setminus a$ means "all the clauses in which j is directed, except the clause a". By putting the real parameter λ to 0, one retrieves the well known Belief Propagation equations. Last, these equations include the external input field imposed onto the variable i, which has an intensity π and a direction $\xi_i \in \{-1, 0, 1\}$. The fact that $\pi \xi_i = \pm \pi$ means that an *a priori* probability π of assuming the value ± 1 is being assigned to the variable i. If, on the other hand, ξ_i equals 0, the variable i is *a priori* left unconditioned.

The Equations (2, 3) form a closed set of equations between the $\eta_{a\to i}$, called "Survey Propagation (SP) equations with external inputs". Iterating these equations permits to find the fixed point $\eta^*_{a\to i}$, probability that, among all the clusters of solutions, a sends a warning to i.

2.4 Temporary Local Field

At each sweep, to perform the local updates of the intensity of the external inputs, as will be explained below, one needs to compute the local temporary field for each variable i, *i.e.* the temporary tendency of the variable i to be equal to $1^{(+)}$, $0^{(-)}$ or left unconstrained.

The tendency of the variable i to be equal to $1^{(+)}$ (resp. $0^{(-)}$) then reads:

$$W_i^{(\pm)} = \frac{\hat{\Pi}_i^{\pm}}{\hat{\Pi}_i^{\pm} + \hat{\Pi}_i^{\mp} + \hat{\Pi}_i^{0}} \ . \tag{5}$$

in which $\hat{\Pi}_i^{\pm,0}$ is equivalent to $\Pi_i^{\pm,0}$ of Equ. 4, but by now considering all the inputs including the clause a.

3 Presentation of the Network

3.1 A Fully Parallel Decimation of the K-SAT Equations

We shall now introduce the important corollary of the SP algorithm with external forcing field, which allows the retrieval of solutions close to any N-dimensional point $\boldsymbol{\xi}$ in $ln(N)$ time when implemented on a fully distributed device (SP-parallel). On each variable i, the direction of the forcing field is fixed at the external input value $\xi_i \in \{-1, 1\}$, while the intensity π_i of its external forcing field is now regularly updated : it equals π with probability 1 (resp. with probability p) when the temporary local field is aligned (resp. unaligned) with the external forcing ; its value is 0 otherwise. In words, if, locally, the network wants to align with the external forcing, one further stabilizes it in this direction ; if the network really does not want, one still tries to drive it in that direction (so that one obtains a solution as close as possible from the external forcing), but not too much (with a probability $p < 1$) in order not to generate contradictions.

Good properties of convergence are obtained by updating the direction of the forcing field once every two steps, *i.e.* once every two parallel updates of all the etas. At the end of a unique convergence, using a right and formula-dependent choice of π and p, most variables are completely polarized, and a solution of the K-SAT formula is finally found by fixing each variable in the direction of its local field. As the update of the forcing is reminiscent of the Perceptron learning, one calls such an algorithm the "Perceptron-like algorithm" : it is an efficient distributed solver of the K-SAT formulas in the hard-SAT phase.

3.2 "Perceptron-Like Decimation" Algorithm

INPUT: the K-SAT formula, an external input vector; a maximal number of iterations t_{max}, a requested precision ϵ, and the parameters π and p for the update of the external forcing
OUTPUT: if it has converged before t_{max} sweeps: one assigment, close from the input vector, which satisfies all clauses.

0. At time $t = 0$: for every edge $a \to i$ of the factor graph, randomly initialize the cavity bias $\eta_{a\to i}(t = 0) \in [0, 1]$. The intensities of the input forcing field are initially set to $\pi_i = 0$.
1. For $t = 1$ to $t = t_{max}$:
   1.1 update parallelely the $\eta_{a\to i}(t)$ on all the edges of the graph, using subroutine CBS-UPDATE.
   1.2 half of the times, update parallelely the intensity $\pi \in [0, 1]$ for all variables x_i, using subroutine FORC-UPDATE.
   1.3 If $|\eta_{a\to i}(t) - \eta_{a\to i}(t-1)| < \epsilon$ on all the edges, the iteration has converged and generated $\eta^*_{a\to i} = \eta_{a\to i}(t)$: GOTO label 2.
2. If $t = t_{max}$ return UN-CONVERGED. If $t < t_{max}$ return the satisfying assignment which is obtained by fixing the boolean variable x_i parallel to its local field : $x_i = sign\{W_i^{(+)} - W_i^{(-)}\}$.

Subroutine CBS-UPDATE$(\eta_{a\to i})$**.**
INPUT: Set of all $\eta_{b\to j}$ arriving onto each variable node $j \in V(a) \setminus i$
OUTPUT: new value for the $\eta_{a\to i}$.

1. For every $j \in V(a) \setminus i$, compute the values of $\Pi^{\pm}_{j\to a}, \Pi^{\mp}_{j\to a}, \Pi^{0}_{j\to a}$ using Eq. (4).
2. Compute $\eta_{a\to i}$ using Eq. (2, 3).

Subroutine FORC-UPDATE(π_i)**.**
INPUT: Set of all cavity bias surveys arriving onto the variable node i, including the forcing field (π_i, ξ_i).
OUTPUT: new value for the intensity π_i of the additional survey.

1. Compute the local fields $W_i^{(+)}, W_i^{(-)}, W_i^{(0)}$ using Eq.5.
2. Compute π_i : $\pi_i = \pi$ with probability $P(\pi_i = \pi) = p + (1 - p) \times \theta(W_i^{(\xi_i)} - W_i^{(-\xi_i)})$, $\pi_i = 0$ otherwise.

3.3 Choice of the Network

The results of the experiments presented below are performed with $K5$ "regular" formulas, in which : first, the number of clauses where any given variable appears negated or directed is kept strictly equal ; second, the number of clauses to which belongs a given variable (the connectivity of the variable) is kept strictly constant for all variables (here one chooses $c = 84$, *i.e.* $\alpha = 16.8$).

The choice has been motivated by the good reconstruction properties of such graphs when used to implement "lossy data compression" [2, 3]. Except if stated otherwise, ones takes networks of $N = 10000$ variables, above which size convergence properties don't dramatically change (Fig. 7).

4 Characterization of the Network

4.1 Distance Between the Input Forcing and the Output Solution

One presents to the network an input vector $\boldsymbol{i}$. The network, driven by $\boldsymbol{i}$, progressively converges towards a stable solution $\boldsymbol{\sigma_o}$, which is also the output of the network.

The optimal intensity π_{opt} and the optimal probability of flip p_{opt}, for which one obtains in average the closest solution $\boldsymbol{\sigma_o}$ from the input vector $\boldsymbol{i}$, are respectively equal to 0.52 and 0.72. For such choices of the parameters, the average Hamming distance $d(\boldsymbol{i}, \boldsymbol{\sigma_o}) = \frac{1}{2N}(N - \sum_{i=1}^{N} i_i \sigma_{o,i})$ between the input vector and the corresponding solution equals 0.314, which is slightly better (but in agreement with) the value 0.329 found by performing the classical serial decimation [2, 3].

4.2 Time of Convergence

At each sweep (*i.e.* at each parallel update of all etas), one constructs a temporary assignment $\boldsymbol{\sigma_t}$ in which each coordinate $\sigma_{t,i} = sign(W_i^+ - W_i^-)$ is fixed parallel to the temporary local field. Fig. 2 represents the evolution respectively of the energy E of the temporary assignment $\boldsymbol{\sigma_t}$ and of the Hamming distance $d(\boldsymbol{i}, \boldsymbol{\sigma_t})$ between the input vector and $\boldsymbol{\sigma_t}$. The algorithm converges towards a solution after an average of 350 sweeps.

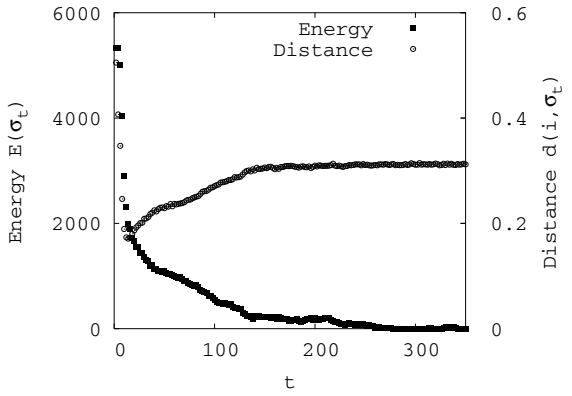

Fig. 2. Time of convergence towards a solution, by imposing a random forcing input

4.3 Stability of the Network - Resistance to Noise

The stability of the network is examined by taking a solution $\boldsymbol{\sigma_o}$ of the K-SAT, and forcing the network with a noisy input vector $\boldsymbol{i}$ increasingly distant from the solution $\boldsymbol{\sigma_o}$. One recovers a solution $\boldsymbol{\sigma}$ which tends to belong to the same cluster as $\boldsymbol{\sigma_o}$ (Fig. 3). In the language of the attractor network, the critical distance $d_c = 0.265$, for which the cluster converges in half of the cases towards the initial cluster $Cl(\boldsymbol{\sigma_o})$ is a measure of the average radius of the basin of

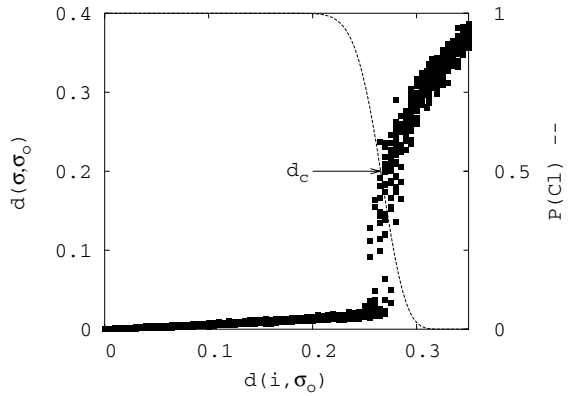

Fig. 3. Stability of the cluster of solutions $Cl(\boldsymbol{\sigma_o})$ to noise. Forcing the system with the input $\boldsymbol{i}$ increasingly distant from a solution $\boldsymbol{\sigma_o}$ permits to recover a solution $\boldsymbol{\sigma}$. $P(Cl)$ is the probability that $\boldsymbol{\sigma}$ belongs to the same cluster as $\boldsymbol{\sigma_o}$

attraction of the cluster $Cl(\boldsymbol{\sigma_o})$. Conversely, the curve P(Cl) of the probability of convergence towards the same cluster $Cl(\boldsymbol{\sigma_o})$ as a function of the distance $d(\boldsymbol{\sigma}, \boldsymbol{\sigma_o})$ between the initial solution and the output solution is a measure of the sharpness of the boundary of the basis of attraction of $Cl(\boldsymbol{\sigma_o})$.

4.4 Associative Properties of the Network

The associative properties of the network have been used (in the context of serialized computation) to build "lossy data compressors" [2, 3]. By presenting a truncated part of the solution, one recovers after convergence a whole solution from the same cluster. Without further optimization, and by choosing an intensity $\pi = 0.99$ and a probability of flip $p = 0.99$, Fig. 4 shows that 18% of the solutions is enough to recover a whole solution from the same cluster.

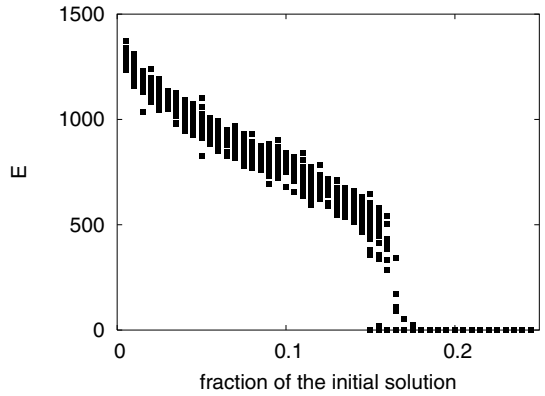

Fig. 4. Stability to missing inputs. One forces the system with a fraction of the initial solution $\boldsymbol{\sigma_o}$. E is the energy of the assignment found at the end of the convergence

5 Electronic Implementation

The previous algorithm is translated into a VHDL code, in view of a FPGA implementation. All calculations are performed synchronously.

5.1 Structure of the VHDL Code

At the descriptive level : a component called *graph* is linked to the external word (which can be a computer), receiving the vector $\boldsymbol{i}$ as input, and providing the corresponding solution $\boldsymbol{\sigma_o}$ as output. *Graph* links together components called *computational-unit*, each of them performing the calculations involving directly a given variable. If two variables i and j belong to the same clause in the K-SAT formula, then the two *computational-unit* representing respectively each of the variables are linked by a signal (Fig. 5).

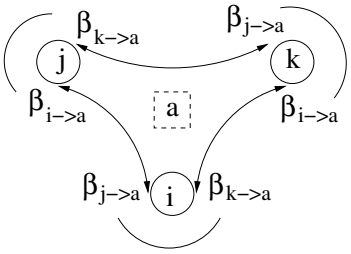

Fig. 5. Scheme of the transmission of the $\beta_{j\to a}$ when COMMAND reads "01"

Main input/output lines of "computational-unit" representing the variable j are drawn in Fig. 6. The INPUT-BETA-IA line transmits the value $\beta_{i\to a}$ from the variable i belonging to the same clause a as j. The OUTPUT-BETA-JA line transmits to all variables belonging to clause a the value $\beta_{j\to a}$. The EPS line is not compulsory : it transmits the information about local convergence.

Moreover, all components are wired together through a common clock signal (CLK), an enable signal (ENABLE) and a two-bit command line (COMMAND).

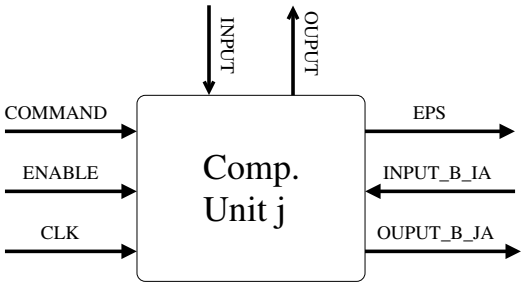

Fig. 6. Main input/output of the "computational-unit" component

At the behavioral level, each unit constantly repeats four synchronized operations, determined by the value of the COMMAND line. When COMMAND equals :

- "00" : calculate $\beta_{j\to a}$ using Eq. (3), for all clauses a to which belongs j
- "01" : transmit $\beta_{j\to a}$
- "10" : read $\beta_{i\to a}$, calculate $\eta_{a\to j}$ using Eq. 2
- "11" : update intensity of the forcing π_i

5.2 Bit Representation of the Reals

For space optimization, the reals $(\beta_{j\to a}, \eta_{a\to j} \in [0,1]^2$ have to be represented by the possibly smallest number of bits. Convergence can be consistently achieved if the reals are encoded in 12 bits, and if the update of the forcing is performed once every two updates of the etas.

5.3 Logarithmic Dependence of the Time of Convergence on the Size of the Graph

Fig. 7 shows that the time to converge to a solution depends logarithmically on the size of the graph : $t \simeq 37 \times ln(N)$ for $N \geq 2000$.

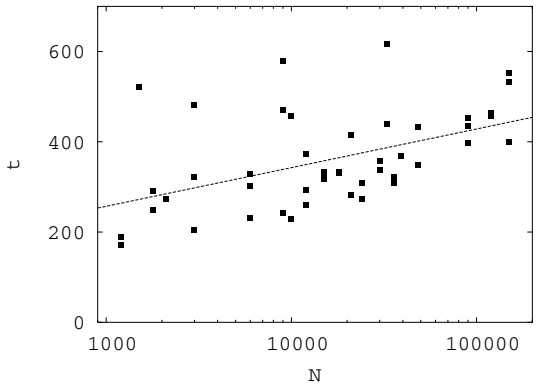

Fig. 7. Logarithmic dependence of the time of convergence on the size of the graph. The fit is performed for $N \geq 2000$. Graphs until a size of 3000 inc. have been done using ModelSim and the VHDL code. Beyond, one used a C code performing the same task

For such constrained graphs, information from local messages has to propagate through the whole graph before convergence. This can't be done faster than in a logarithmic time. Thus, this time of convergence to find a K-SAT solution on a distributed device is, in terms of scaling, the best possible.

6 Conclusion

One has built a network and written its VHDL code which permit to find solutions of the K-SAT formulas in $ln(N)$ time, the best achievable scaling on a distributed device. A physical interpretation of alternative methods of parallelization will be described in detail in another manuscript [5]. This algorithm or its derivatives, which define a local network with an exponential number of attractors, has applications in error-correcting codes [2, 3] and gene-regulatory networks [6]. Its powerful reconstruction properties are also likely to be used in probabilistic inference for constrained graphs [8], artificial intelligence and theoretical neuroscience [1, 7].

Acknowledgments

We warmly thank Alexandre Chapiro and Maria Liz Crespo for useful discussion and technical advises. We thank Alfredo Braunstein for discussions, Michele Leone and Andrea Pagnani for comments on the manuscript.

References

1. Amit,D.,J.: Modeling brain function: the world of attractor neural network. Cambridge University Press (1992)
2. Battaglia, D., Braunstein,A., Chavas,J., Zecchina,R.: Source coding by efficient selection of ground states. Submitted to Phys.Letters (2004) cond-mat/0412652
3. Battaglia, D., Braunstein, A., Chavas,J., Zecchina,R.: Exact probing of glassy states by Survey Propagation. Proceedings of the conference SPDSA-2004,Hayama, Japan (2004)
4. Braunstein,A., Mézard,M., Zecchina,R.: Survey propagation: an algorithm for satisfiability. Preprint, to appear in Random Structures and Algorithms (2002) cs.CC/0212002
5. Chavas,J., Furtlehner,C.,Mézard,M.,Zecchina,R.: Survey-propagation based decimation through purely local computations. J. Stat.Phys. (2005) in preparation
6. Correale, L., Leone, M., Pagnani, A., Weigt, M., Zecchina, R.: Complex regulatory control in boolean networks. Sumitted to Phys.Letters (2004) cond-mat/0412443
7. Latham,P.E., Deneve, S., Pouget,A.: Optimal computation with attractor networks. J. Physiol. Paris **97** (2003) 683–694
8. Madsen, M., L., Jensen,F.,V.: Parallelization of inference in bayesian networks.
9. Mézard,M., Parisi,G., Zecchina, R.: Analytic and algorithmic solution of random satisfiability problems. Science **297** (2002) 812
10. Mézard,M., Zecchina,R.: The random K-satisfiability problem : from an analytic solution to an efficient algorithm. Phys. Rev. Letters E **66** (2002) cs.CC/056126
11. Yedidia,J.S., Freeman,W.T., Weiss,Y.: Generalized belief propagation. Advances in Neural Information Processing Systems **13** (2001) 689–695