INSTITUTE OF PHYSICS PUBLISHING JOURNAL OF PHYSICS A: MATHEMATICAL AND GENERAL

J. Phys. A: Math. Gen. **35** (2002) 7559–7574 PII: S0305-4470(02)35979-1

Complexity transitions in global algorithms for sparse linear systems over finite fields

A Braunstein[1,2], **M Leone**[1,3], **F Ricci-Tersenghi**[1,4] **and R Zecchina**[1,5]

[1] International Center for Theoretical Physics, Strada Costiera 11, PO Box 586, I-34100 Trieste, Italy
[2] SISSA, via Beirut 9, I-34100 Trieste, Italy
[3] INFM and SISSA, via Beirut 9, I-34100 Trieste, Italy
[4] Dipartimento di Fisica, Università di Roma 'La Sapienza', Piazzale Aldo Moro 2, I-00185 Roma, Italy
[5] Laboratoire de Physique Théorique et Modèles Statistiques, Université Paris Sud, 91405 Orsay, France

E-mail: abraunst@ictp.trieste.it, micleone@ictp.trieste.it, Federico.Ricci@roma1.infn.it and zecchina@ictp.trieste.it

Received 11 April 2002, in final form 10 July 2002
Published 22 August 2002
Online at stacks.iop.org/JPhysA/35/7559

Abstract

We study the computational complexity of a very basic problem, namely that of finding solutions to a very large set of random linear equations in a finite Galois field modulo q. Using tools from statistical mechanics we are able to identify phase transitions in the structure of the solution space and to connect them to the changes in the performance of a global algorithm, namely Gaussian elimination. Crossing phase boundaries produces a dramatic increase in memory and CPU requirements necessary for the algorithms. In turn, this causes the saturation of the upper bounds for the running time. We illustrate the results on the specific problem of integer factorization, which is of central interest for deciphering messages encrypted with the RSA cryptosystem.

PACS numbers: 89.20.Ff, 75.10.Nr, 05.70.Fh, 02.70.−c

1. Introduction

The methods and concepts of statistical physics of disordered systems constitute a very useful tool for the understanding of the onset of computational complexity in randomly generated hard combinatorial problems. Once the optimization problems are translated into zero temperature spin glass problems, one may study the geometrical changes in the space of solutions as symmetry breaking phenomena. In this context one may view the exponential regimes of randomized search algorithms as out-of-equilibrium phases of stochastic processes.

However, combinatorial problems are not always exponentially hard: problems that can be solved in polynomial time, even in their worst-case realizations, compose the so-called polynomial (P) class [1]. Such problems are often of great practical relevance and are tackled using large scale computations. Examples can be found in all disciplines: in physics, just to make one example, one may study ground states of 2D spin glass like Hamiltonians resorting to a polynomial max-cut algorithm [2]. The major applications are obviously found in engineering: examples are design problems (finite elements methods), control theory (convex optimization), coding theory (parity check equations) and cryptography (integer factorization).

Due to the practical relevance of the problems and typically large number of variables used for their encoding, that is the size of the problems, it is of basic interest to look at the fine structure of the class P in order to concretely optimize the computational strategies. For instance, in error correcting codes it is crucial to have algorithms that converge in *linear* time with respect to the number of encoded bits, any power larger than 1 being considered of no practical interest.

Quite in general, the trade-off between time and memory resources is the guiding criterion which selects the algorithms used in real-world applications. Roughly speaking, polynomial algorithms can be divided into different groups depending on the solving strategy they implement. The main groups are local algorithms (e.g. greedy/gradient methods), global algorithms (e.g. Gaussian elimination or Fourier transform methods), iterative algorithms (e.g. Lanczos method) and parallel algorithms (see [3] for a basic introduction to the subject).

In what follows we shall study a prototype problem of the P class, that is the problem of solving large and random sparse systems in some Galois field GF(q). Working in GF(q) is completely equivalent to performing any operation modulo q.

Firstly, we give a precise analysis of the computational features for nontrivial ensembles of random instances. By a statistical mechanics study, we look into the—symmetry breaking—geometrical structure of the space of solution thereby providing an explanation for the changes in the power law behaviour observed in different algorithms. Moreover, we are able to predict and explain in terms of clustering of solutions, the memory catastrophe found in global algorithms such as Gaussian elimination. Such an effect seriously hampers the application of this sort of global algorithm in many circumstances, one example being symbolic manipulations. This memory catastrophe in turn induces an even more dramatic increase in CPU time, which makes large problems unaffordable above the dynamical threshold γ_d (see below for its definition).

Secondly, we consider a specific 'real-world' application, namely the integer factorization problem used in RSA public key cryptography [4]. Using a nontrivial mapping of the factoring problem on a sparse linear system modulo 2, endowed with a quite peculiar statistical distribution of matrix elements, we analyse which are the characteristic geometrical properties of solutions that are responsible for the usage of specific algorithms and constitute the possible bottleneck for the near future.

Interestingly enough, the changes in both time or memory requirements during the solution process of sparse systems can be interpreted in physical terms as a dynamical transition at which the phase space of the associated physical systems splits into an exponential number of ergodic components. While it is to be expected that local algorithms get stuck by local minima at such phase boundary, it is less obvious to predict which is the counterpart of the dynamical transition in global algorithms, for which polynomial time convergence is guaranteed even for the hardest instances. Indeed the dynamical transition manifests itself as a phase transition in the computational requirements which in turn leads to a slowing down phenomenon that saturates the upper bound for the convergence time. Such a change of scale

in memory requirements constitutes a serious problem for hardware implementations of large scale simulations.

Hopefully, the paper is written in a style accessible to a cross disciplinary audience.

2. Random linear systems in GF(2): rigorous results and statistical mechanics analysis

The theory of random equations in finite fields is shared by probability, combinatorics and algebra [5].

For the sake of simplicity we limit our statistical mechanics analysis to GF(2) rather than GF(q), the extension to $q > 2$ being straightforward though technically complex.

As is well known in the context of error correcting codes [6], solving a sparse linear system modulo 2 is equivalent to finding the zero temperature ground states of a class of multiple degree interactions p-spin models on diluted random graphs.

Let us consider a random linear system in GF(2) in the form $\hat{A}\vec{x} = \vec{y} \bmod 2$, where $\hat{A}$ is a 0–1 matrix of dimension $M \times N$. For each of its specific choices $\hat{A}$ can be interpreted as the contact matrix of a particular random (hyper-)graph belonging to a specific ensemble. The class of random matrices we shall deal with is defined by the fraction of rows a_k with k nonzero elements. The latter are placed uniformly at random within each row.

We focus on matrices that lead to graphs with an average connectivity value $\langle k \rangle = \sum_k a_k k$ finite and much less than both M and N. We are interested in the limit of very large matrices, where we can assume $N, M \to \infty$ with a finite ratio $\gamma \equiv M/N$.

This is the regime in which a study of the computational cost is important, in that it applies directly to large scale computations. In the limit $N, M \to \infty$ average quantities characterizing the system (e.g. the average fraction of violated equations) are known to be equal to the most probable values (i.e. their probability distribution is strongly peaked [7]) and therefore single random large systems behave as the average over the ensemble.

We will always assume $a_1 = 0$ at the beginning, since rows with a single 1 correspond to trivial equations which can be removed *a priori* from the set.

The equivalence between linear systems and spin models is quite straightforward. We start from a set of linear equations in GF(2), $\hat{A}\vec{x} = \vec{y}$, and we build up a spin Hamiltonian whose ground state energy E_{gs} counts the minimal number of unsatisfied equations. In the case where $E_{gs} = 0$, ground state configurations will correspond to solutions of the original set of linear equations and the zero-temperature entropy will count the number of such solutions.

The construction is done as follows: for every equation, labelled by $i \in [1 \dots M]$, let us define the set of variables $\vec{x}$ entering equation i as

$$v(i) \equiv \{j \in [1 \dots N] : A_{ij} = 1\}. \tag{1}$$

With the transformation $s_j = (-1)^{x_j}$ and $J_i = (-1)^{y_i}$, we have that every equation can be converted into a term of the Hamiltonian through

$$\sum_{j=1}^{N} A_{ij} x_j = y_i \qquad \Leftrightarrow \qquad \sum_{j \in v(i)} x_j = y_i \qquad \Leftrightarrow \qquad \prod_{j \in v(i)} s_j = J_i \tag{2}$$

where the multi-spin interaction contains at least two spins since we set $a_1 = 0$. Then the Hamiltonian

$$H = \frac{1}{2}\left[M - \sum_{i=1}^{M} J_i \prod_{j \in v(i)} s_j \right] \tag{3}$$

fits the above requirements and can be used in the analytical treatment.

A better form for the above Hamiltonian can be obtained by grouping together k-spin terms with the same k, that is

$$H = \frac{1}{2}\left[M - \sum_{k} \sum_{i_1<i_2<\cdots<i_k} J_{i_1 i_2 \cdots i_k} s_{i_1} \cdots s_{i_k} \right] \tag{4}$$

where $s_i = \pm 1$ are Ising spins and the couplings $J_{i_1 i_2 \cdots i_k}$ are i.i.d. quenched random variables taking values in $\{0, \pm 1\}$. The total number of interactions, that is of terms with $J \neq 0$, is M, and the energy is zero if and only if all the interactions are satisfied. For each unsatisfied interaction the energy increases by 1.

The fraction of interactions of k-spin kind is a_k and thus the probability of having $J_{i_1 i_2 \cdots i_k} \neq 0$ equals $a_k M \big/ \binom{N}{k} \simeq \gamma a_k k! \big/ N^{k-1}$, while the sign of $J_{i_1 i_2 \cdots i_k}$ depends on the probability distribution of the components of $\vec{y}$,

$$P\left(J_{i_1 i_2 \cdots i_k}\right) = \left[1 - \frac{\gamma a_k k!}{N^{k-1}}\right] \delta\left(J_{i_1 i_2 \cdots i_k}\right) + \frac{\gamma a_k k!}{N^{k-1}} \left[p\delta\left(J_{i_1 i_2 \cdots i_k} - 1\right) + (1-p)\delta\left(J_{i_1 i_2 \cdots i_k} + 1\right)\right] \tag{5}$$

where $p \in [0, 1]$ controls the fraction of 0s in $\vec{y}$. As long as the system admits at least one solution, it can always be brought by a gauge transformation in the form with $p = 1 \Leftrightarrow \vec{y} = \vec{0}$. This corresponds to positive or null couplings only, as in a diluted ferromagnetic model.

In order to make a connection between the behaviour of solving algorithms and the structure of the matrix $\hat{A}$, we study the geometrical properties of the space of solution, i.e. ground states of (4), as a function of γ for nontrivial choices of $\{a_k\}$. We may have access to the structure of such a space by just performing the $T = 0$ statistical mechanics analysis of the spin glass model, with control parameter γ.

For γ large enough, say at γ_c, the system of equations becomes over-determined and some of the equations can no longer be satisfied. This fact is reflected in the ground state energy of the associated spin glass model becoming positive. The interesting aspect of the problem is that, under proper conditions, there appears a clustering phenomenon with macroscopic algorithmic consequences at some intermediate value $0 < \gamma = \gamma_d < \gamma_c$. We will focus our attention on the latter transition, thus assuming *a priori* that at least one solution always exists. This allows us to fix $\vec{y} \equiv \vec{0}$ hereafter.

The complete picture of the typical structure of the solution space can be obtained through a replica calculation, following a well-tested scheme in diluted systems, as in [8, 9]. The results of such calculations have recently been confirmed by a rigorous mathematical derivation [10, 11]. Here we avoid repeating standard calculations already presented in [8, 9] and extensively reviewed in [12], and we directly present the results.

Due to the zero energy condition ($E_{gs} = 0$ for $\gamma < \gamma_c$), the dominance of thermodynamical states is purely to be determined in entropic terms. Defining $S_0(\gamma)$ as the logarithm of the number of solutions to $\hat{A}\vec{x} = \vec{0}$ divided by N, we have that

$$S_0(\gamma) = S(m, \gamma) = \log(2) \left[(1-m)[1 - \log(1-m)] - \gamma \sum_{k \geqslant 2} a_k (1 - m^k) \right] \tag{6}$$

where m solves

$$G(m) = 1 - m - \exp\left(-\gamma \sum_{k \geqslant 2} k a_k m^{k-1} \right) = 0. \tag{7}$$

When more than one solution to equation (7) exists, the one maximizing $S(m, \gamma)$ must be chosen.

At fixed $\{a_k\}$, one can study the phase diagram as a function of γ. At γ low enough, equation (7) has only the trivial solution $m = 0$ and the system is paramagnetic with entropy $S(0, \gamma) = \log(2)(1 - \gamma)$. Typically a nontrivial magnetized solution for the order parameter, $m^* > 0$, appears at a value γ_d such that

$$G(m^*) = 0 \qquad \text{and} \qquad \left.\frac{\partial G(m)}{\partial m}\right|_{m=m^*} = 0. \tag{8}$$

This solution becomes entropically favoured at a value γ_c found by solving

$$S(0, \gamma_c) = S(m^*, \gamma_c). \tag{9}$$

The crucial observation is the following. At γ_d, together with the magnetized solution, there appear other spin glass solutions to the saddle-point equation. In particular, it can be shown [10, 13] that the difference between the paramagnetic and the ferromagnetic entropies,

$$\Sigma(\gamma) = S(0, \gamma) - S(m^*, \gamma) \tag{10}$$

gives the configurational entropy of the problem, that is the number of clusters of solutions[6]. There exist $\exp[\Sigma(\gamma)N]$ well-separated clusters (Hamming distances $\sim \mathcal{O}(N)$), each one containing a number $\exp[S(m^*, \gamma)N]$ of closed solutions (Hamming distances $\sim \mathcal{O}(1)$).

This clusterization has two main consequences. Local algorithms for finding solutions running in linear time in N stop converging [8]: this is the typical situation for a greedy algorithm which gets stuck in one of the most numerous local minima at a positive energy.

Global algorithms, which are guaranteed to converge in polynomial time, need to keep track of the computation of this complex structure of solutions and a memory linear in N turns out to be insufficient, as we will show below.

The simple cases $\{a_2 = 1; a_{k\neq 2} = 0\}$ and $\{a_3 = 1; a_{k\neq 3} = 0\}$ are particularly illuminating [9]. Self-consistency equations for the order parameter and for the ground state entropy can immediately be retrieved from the general formulae (7) and (6).

The first case represents a simple Ising spin model on a random graph. The analysis of thermodynamic phases shows a trivial paramagnetic region at low γ, followed by a second-order phase transition at $\gamma_d = \gamma_c = 1/2$ where both the ground state energy and the magnetization become positive in a continuous way. From the linear system point of view, working with sparse equations with only two variables per row is always easy, i.e. algorithms have no slowing down. The solving process progressively fixes variables and smoothly goes on until a complete solution is found. Indeed, fixing one variable immediately fixes the other one within the equation, and this goes on in a cascade process that prevents the accumulation of too long symbolic memory-taking expressions.

The second case of the 3-spin model was tackled in full detail in [8, 13]. It shares the general characteristics of all models without, or at most with a very small fraction of, 2-spin terms. In this case a gap between γ_d and γ_c opens up and a nonvanishing configurational entropy Σ appears there. At γ_c the magnetization jumps to a finite value.

For a general choice of $\{a_k\}$, the configurational entropy reads

$$\Sigma(\gamma) = \log(2)\left[1 - (1 - m)[1 - \log(1 - m)] + \gamma \sum_{k\geqslant 2} a_k m^k\right] \tag{11}$$

where m is the largest solution to equation (7). As discussed in [13], the values of γ_d and γ_c are found as the points where $\Sigma(\gamma)$ first appears with a nonzero value and where it reaches zero again.

[6] Two solutions belong to the same cluster (resp. to different clusters) if their Hamming distance is $\mathcal{O}(1)$ (resp. $\mathcal{O}(N)$).

The algorithmic consequences of having $\Sigma(\gamma) > 0$ have been already exposed in [8, 13]: for $\gamma > \gamma_d$ a glassy state with positive energy arises, which traps any local dynamics, preventing it converging towards the ground state of zero energy. We conjecture the counterpart on global algorithms, such as Gaussian elimination, to be that the resolution time increases with N faster than linear.

In the next section we will check the above conjecture with two different Gaussian elimination algorithms, neither of which is able to solve the system in linear time for $\gamma > \gamma_d$.

3. Algorithm behaviour

In this section we analyse the performances of a couple of different 'Gaussian elimination' algorithms, their difference being in the order equations are solved. We will measure the number of operations and the size of the memory required for the solution of a set of linear equations, that is the complexity of finding all solutions to $\hat{A}\vec{x} = \vec{y}$.

We will see that, for a generic ensemble of random problems, any algorithm undergoes an easy/hard transition at a certain γ value, which cannot be pushed beyond the dynamical transition threshold γ_d. In this context we call such problems *easy* which are solvable with a CPU time and memory of order N, and *hard* those requiring resources scaling with N^α, where $\alpha > 1$.

Given a set of M linear equations in N variables, Gaussian elimination proceeds as follows (for concreteness we will always work in GF(2)): at each step, it takes an equation, e.g. $x_1 + x_2 + x_3 = y_1$, solves it with respect to a variable, e.g. $x_1 = x_2 + x_3 + y_1$, and then substitutes variable x_1 with the expression $x_2 + x_3 + y_1$ in all the equations still unsolved. This procedure gives all the solutions to any set of linear equations in, at most, $\mathcal{O}(N^3)$ steps and using $\mathcal{O}(N^2)$ memory. Nevertheless this bound only holds in the worst case, namely when the matrix $\hat{A}$ is dense. Very often, in actual applications, the matrix is sparse and the algorithm is faster. We define a matrix with $\mathcal{O}(N)$ ones sparse and that with $\mathcal{O}(N^2)$ ones dense.

In order to analyse the computational complexity of this problem and its connections to phase transitions, we focus on a specific ensemble of random problems, generalizations to other ensembles being straightforward. We choose sets of $M = \gamma N$ linear equations, each one containing exactly $k = 3$ of the N variables, taking values in GF(2). Thus the connectivity of a variable, defined as the number of equations this variable enters, takes values from a Poissonian distribution of mean 3γ.

For very large N, that is in the thermodynamical limit, we are interested in how the complexity changes with γ. Moreover, for a fixed γ such that the problem is hard, we would like to know when (in terms of the running time t) and why the algorithm becomes slower and slower.

The running time t is measured as the number of equations already solved, normalized by N, and thus takes values in $[0, \gamma]$. $\hat{A}_t$ is the matrix representing the set of equations after tN steps, and it has the form shown in figure 1. See figure 2 for the actual shape of $\hat{A}_t$ in a specific case with 1024 equations in 1024 variables. For simplicity, we have re-ordered the variables and the equations of the system, such that, at the ith step, we solve the ith equation with respect to x_i. With this choice the left part of the matrix $\hat{A}_t$ has 1s on the diagonal and 0s below. The right part can be naturally divided into an upper part U and a lower one L. The density of 1s in the L matrix—let us call it $\rho(t; \gamma)$—is uniform and depends on the initial γ, the time t and the algorithm used for solving the linear system. The density of 1s in the U part is not uniform and varies from row to row, as shown in figure 1 with grey tones. For continuity reasons the density at the mth row of U is exactly $\rho(m/N; \gamma)$. Then U is sparse or dense depending on whether L is. Defining $k(t; \gamma) = \rho(t; \gamma)N(1 - t)$ the average number of

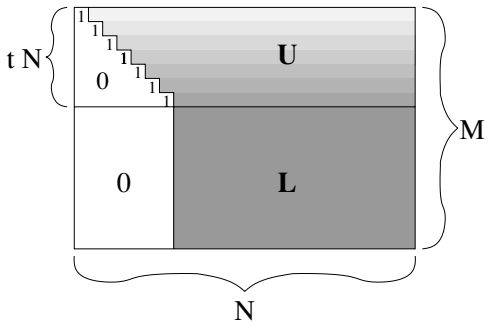

Figure 1. Typical shape of the $\hat{A}_t$ matrix after tN steps of Gaussian elimination.

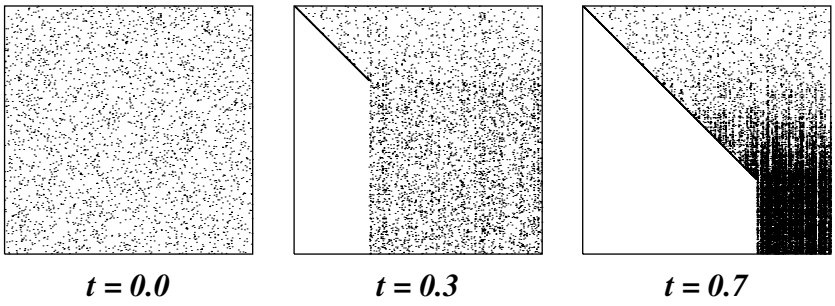

Figure 2. The evolution of the $\hat{A}_t$ matrix for a specific 1024×1024 random system. Every dot corresponds to a 1 entry.

1s per row in L, we have that a sparse (resp. dense) matrix corresponds to having a finite k (resp. ρ).

At each time step, the number of operations required is directly related to the density of the matrix $\hat{A}_t$ and thus to that of L. More specifically, solving with respect to the variable in the upper left corner of L, the number of operations is proportional to the number of 1s in the first row of L, i.e. $k(t;\gamma)$, times the number of rows of L having a 1 in the first column, i.e. $\rho(t;\gamma)N(\gamma - t)$, and thus equals

$$k(t;\gamma)\rho(t;\gamma)N(\gamma - t) = k^2\frac{\gamma - t}{1 - t} = N^2\rho^2(\gamma - t)(1 - t). \tag{12}$$

Then, if the matrix L is sparse a finite number of operations per step is enough, while $\mathcal{O}(N^2)$ operations are required when L is dense. Integrating over time $t \in [0, \gamma]$, we have that the total complexity is given by

$$N\int_0^\gamma \frac{\gamma - t}{1 - t}k^2(t;\gamma)\,\mathrm{d}t = N^3\int_0^\gamma (\gamma - t)(1 - t)\rho^2(t;\gamma)\,\mathrm{d}t. \tag{13}$$

Since the function $\rho(t;\gamma)$ is continuous in t, we conclude that

$$\left.\begin{array}{l}\rho(t;\gamma) \propto 1/N \\ \\ k(t;\gamma)\ \text{finite}\end{array}\right\} \quad \text{for all} \quad t \in [0,\gamma] \quad \Leftrightarrow \quad \rho_{\max}(\gamma) = 0 \quad \Leftrightarrow \quad \begin{cases}\text{CPU time} \propto N \\ \text{memory} \propto N\end{cases} \tag{14}$$

$$\left.\begin{array}{l}\rho(t;\gamma)\text{finite} \\ \\ k(t;\gamma) \propto N\end{array}\right\} \quad \text{for some} \quad t \in [0,\gamma] \quad \Leftrightarrow \quad \rho_{\max}(\gamma) > 0 \quad \Leftrightarrow \quad \begin{cases}\text{CPU time} \propto N^3 \\ \text{memory} \propto N^2\end{cases} \tag{15}$$

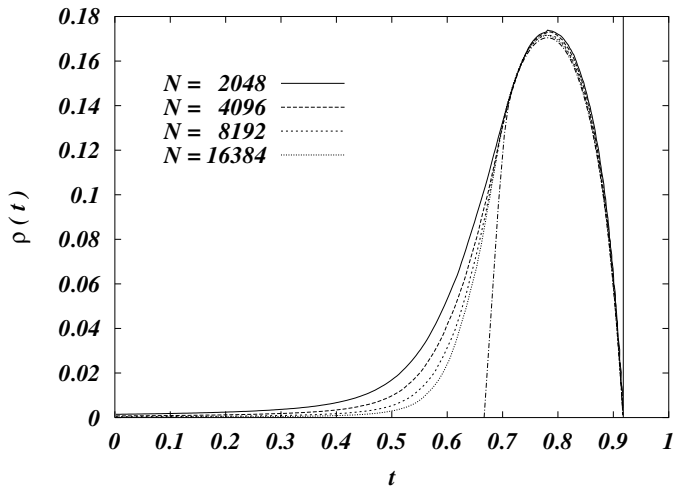

Figure 3. Density of 1s in the L matrix during the solving process with the simplest Gaussian elimination algorithm. The vertical bar marks the analytical critical point $\gamma_c = 0.918$.

where

$$\rho_{\max}(\gamma) = \lim_{N\to\infty} \max_{t\in[0,\gamma]} \rho(t;\gamma) \tag{16}$$

is the order parameter signalling the onset of the hard regime.

Having found the relation between the density of 1s in L and the computational complexity we are interested in, we can now run the algorithms and measure the density $\rho(t;\gamma)$. The easy/hard transition should manifest itself with $\rho_{\max}(\gamma)$ becoming different from zero.

3.1. Simplest Gaussian elimination

Let us start with the simplest algorithm, which solves the equations in the same (random) order they appear in the set and with respect to a randomly chosen variable. In this very simple case, one can easily show that the complexity for solving a set of linear equations with initial parameter $\gamma = \gamma_0$ is exactly the same as for solving a larger system with $\gamma > \gamma_0$ up to time $t = \gamma_0$. For this reason, in this case the function $\rho(t;\gamma)$ does not depend on γ and can be calculated once for all the relevant γ values.

Moreover, it is known [8] that this algorithm, in the limit of very large N, keeps the matrix sparse for all $\gamma < 2/3$.

In figure 3 we show the function $\rho(t)$ for many large N values. The dotted-dashed line is a guide to the eyes and it should not be too much different from the thermodynamical limit: it goes through the two points ($\gamma = 2/3$ and $\gamma = 0.918$) where $\rho(t)$ must vanish and coincide with numerical data in the region, where data for different sizes seem to be quite close to the asymptotic shape.

In the thermodynamical limit, the algorithm keeps the matrix sparse for times $t \leqslant 2/3$ and so it undergoes an easy/hard transition at $\gamma = 2/3$: as long as $\gamma \leqslant 2/3$, $\rho_{\max}(\gamma) = 0$, while $\rho_{\max}(\gamma) > 0$ for $\gamma > 2/3$. As we will see below, the location of the transition depends on the algorithm used and, in this case, does not correspond to any underlying thermodynamical transition.

We note *en passant* that the γ value where the L matrix becomes sparse again seems to correspond to the critical point $\gamma_c = 0.918$ [8, 10, 11] (marked with a vertical line in

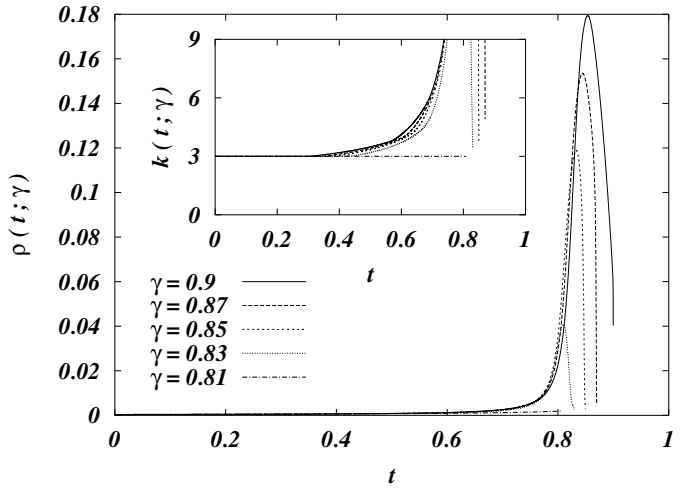

Figure 4. Density of 1s in the L matrix during the solving process with a smart Gaussian elimination algorithm ($N = 8192$). Inset: zoom on the low-density part (with a different normalization).

figure 3). An explanation of this observation will be given in a forthcoming publication. It implies that the value of the critical point γ_c, which is relevant e.g. in the XORSAT model [14] in theoretical computer science, could also be obtained by solving differential equations for $\rho(t)$.

3.2. Smart Gaussian elimination

Now we turn to a more clever Gaussian elimination algorithm, which works as follows: at each time step, it chooses the variable x having the smallest connectivity in L, i.e. corresponding to the less dense column of L, and solves with respect to x any of the equations where x enters.

Clearly, in this case, the dynamics and thus the density of 1s in L depend on the initial γ value: a smaller γ implies that for a longer time we can choose variables of connectivity 1, which do not increase the average number of 1s per row in L. It can rigorously be shown [10, 11] that this procedure keeps the density of the L matrix constant, $\rho(t;\gamma) = \rho(0;\gamma)$, for times smaller than $t^* = \gamma(1 - m^3)$, where m is the largest solution to $1 - m = \exp(-3\gamma m^2)$. The last equation is exactly equation (7) with $\{a_3 = 1, a_{k\neq 3} = 0\}$.

Running the algorithm for different γ values, we obtain the densities reported in the main panel of figure 4. For $\gamma < \gamma_d = 0.818$ the density remains $\mathcal{O}(1/N)$ all along the run, while for $\gamma > \gamma_d$ there is a time when the density becomes finite and the problem hard to handle.

In order to better show what happens around t^*, we have plotted in the inset of figure 4 the mean number of 1s per row, $k(t)$. It is clear that for $\gamma < \gamma_d$ this number remains constant, since one can solve the system choosing variables only of connectivity 1, not altering the L matrix. In contrast, for $\gamma \geqslant \gamma_d$ there is a time $t^*(\gamma)$ when variables of connectivity 1 terminate, and the algorithm has to start making substitutions in L, thus increasing the density of 1s.

Then γ_d marks the onset of computational hardness, both in memory and CPU time. One may also object that this value for the easy/hard transition may depend on the particular algorithm. Note, however, that a completely different linear algorithm described in [8] (which firstly works with high-connectivity variables) seems to work up to γ_d. Moreover, as seen

in the previous section, we have analytically found that at γ_d a transition takes place, which drastically changes the structure of the solutions space, and so we argue that *any* algorithm running in linear time can converge only up to γ_d. Indeed is shown in [10] that solutions spontaneously form clusters for $\gamma > \gamma_d$ and this particular structure requires a larger memory to be stored.

4. The RSA cryptosystem and factorization

In this section we shall validate the above scenario on a concrete application, namely integer factorization problems arising in the RSA cryptosystem. Such problems allow for a nontrivial mapping onto huge linear systems in GF(2) with a rather peculiar structure of the underlying contact matrix. In order to be as self-contained as possible, we firstly give a short review of the problem and the methodology (a detailed description of the RSA cryptosystem can be found in [4]).

The only known method for breaking RSA implies factorization of the private key, which consists in a natural number which is the product of two big prime numbers, $n = p \cdot q$, with p and q approximately of the same size $\simeq\sqrt{n}$. Keys currently used in applications are numbers n ranging from 1024 bits (309 decimal digits) to 2048 bits (617 digits) length.

The first attempt at a massive parallel factorization was the RSA129 (129 digits, 428 bits) challenge, solved in 1994 with the *quadratic sieve* (QS) algorithm. More recently, in August 1999 the RSA155 challenge was solved using the *general number field sieve* (GNFS) algorithm. This has forced the abandonment of the 512 bit (155 digits) length for sensitive information security.

There are now several sub-exponential algorithms for solving the factorization problem, the faster of which is GNFS. QS and GNFS share the same structure, consisting of two phases: the first one in which a big (the size depending mostly on the size of n) linear system in GF(2) is produced, and the second one in which this system is solved.

Although the first phase is definitely more costly, the solving phase (which affects this paper) takes a respectable part of the total time and memory requirement. Especially as numbers get bigger this becomes a limitation, because the fastest solving methods used employ a sole workstation, with the consequent memory restriction. Moreover, in recent factorizations a new filtering phase has been placed between the previous two, in which pieces of the system (specifically columns of the {0, 1}-matrix) get discarded in order to simplify the solving phase, effectively transferring part of the total time from the second phase to the first one.

In this section we will study statistical characteristics of linear systems produced by the QS algorithm. The next subsection is dedicated to a schematic description of QS.

4.1. The QS algorithm

For a nice description of the QS algorithm, see [15]. Synthetically, QS works as follows. It builds a list of integer numbers $\{y_i\}_{i\in I}$ such that

- $y_i \equiv x_i^2 (\mathrm{mod}\, n)$ for some x_i and $y_i \neq x_i$;
- y_i is completely factorizable in a given (relatively small) subset of B primes called the factor base.

This is called the *sieving* phase. The algorithm then searches a subset $J \subset I$ of elements of the list such that $\prod_{i\in J} y_i = z^2$ is a square (*solving* phase). Once found, $z^2 \equiv x^2 (\mathrm{mod}\, n)$ (here $x = \prod_{i\in J} x_i$) and this implies that n divides $(x + z)(x - z)$ and then $\gcd(x - z, n)$ will likely (further trials will increase the probability) be a nontrivial factor of n.

4.1.1. Sieving. In order to find element pairs x_i, y_i such that $y_i \equiv x_i^2 (\mathrm{mod}\, n)$ we can use the polynomial $y = f(x) = x^2 - n$ and evaluate it at different values of x, keeping only values of y which completely factorize between the first B primes (the factor base). The sieving will allow us to do this efficiently.

The idea is that, given p, it is easy to find which are the values of $f(x)$ which are divisible by p, because p divides $f(x)$ if and only if $f(x) = x^2 - n \equiv 0 \,(\mathrm{mod}\, p)$ and this is a quadratic equation in GF(p), having at most two solutions. These solutions are nothing but the square roots of n modulo p (if they exist).

This has a first consequence, i.e. a prime p will not divide $f(x)$ if n is not a square mod p independently of the value of x. So if we can detect these primes, we can eliminate them directly from our set of primes. Detecting them is very easy: using Fermat's little theorem, we know that

$$n^{p-1} \equiv 1 \pmod{p} \tag{17}$$

assuming that p does not divide n (which is trivially a reasonable assumption, anyway, because we are searching a divisor of n). If p is an odd prime, i.e. not 2 (all n are squares $(\mathrm{mod}\, 2)$), then calling $m = n^{\frac{p-1}{2}}$ we have that $m^2 \equiv 1 (\mathrm{mod}\, p)$, so $m \equiv \pm 1 (\mathrm{mod}\, p)$. This m will prove to be handy.

If $n \equiv s^2 (\mathrm{mod}\, p)$ then $m \equiv s^{p-1} \equiv 1 (\mathrm{mod}\, p)$. Conversely, if $m \equiv 1$ then n is a square modulo p (not proved here). The number m is called the Lagrange symbol and can be computed efficiently in one of the first stages of the algorithm. Useful primes (those with $m = 1$) are roughly a random half of all the first B primes. So now we will keep only this half and redefine 'the first B primes' as 'the first B primes with $m = 1$'.

Computing the square root modulo p is a bit more difficult than knowing that it exists, but can also be done efficiently. For instance, the easiest case is when $\frac{p+1}{4}$ is an integer, then $\left(n^{\frac{p+1}{4}}\right)^2 = n^{\frac{p+1}{2}} \equiv mn (\mathrm{mod}\, p)$. As $m = 1$ (or else there is no solution) then $\pm n^{\frac{p+1}{4}} (\mathrm{mod}\, p)$ are the required square roots.

Once we have computed the two solutions $f\left(x_p^{1,2}\right) \equiv 0 (\mathrm{mod}\, p)$, then adding $p, 2p, 3p, \ldots$ to them we will obtain *all* x such that $f(x)$ is divisible by p.

The sieve idea is to initialize an array with values of $f(x)$ for consecutive $x \in [[\sqrt{n}], [\sqrt{n}] + M]$ indexed by x, and then for each p in our factor base to divide the corresponding arithmetic progression of $\left\{ f(x_p^{1,2} + kp), k = 1, \ldots \right\}$ by p. At the end, those values which are completely factored between the primes in the factor base will become 1 (Well, not exactly. Some of them can have multiple times the same prime factor. But we can set up a threshold instead of 1 below which we consider the number completely factored. We can recheck afterwards.). We take those values and put their factorization in an array

$$\begin{array}{c} \\ p_1 \\ \vdots \\ p_B \end{array} \begin{array}{c} \begin{array}{ccc} f(x_1) & \cdots & f(x_m) \end{array} \\ \begin{bmatrix} \alpha_1^{(1)} & \cdots & \alpha_1^{(m)} \\ \vdots & \ddots & \vdots \\ \alpha_B^{(1)} & \cdots & \alpha_B^{(m)} \end{bmatrix} \end{array} \pmod{2}.$$

4.1.2. Solving. The *solving* phase is conceptually simple: a solution of the homogeneous linear system $\hat{A}v = 0$ is a $\{0, 1\}$ vector v which represents correctly the subset J, in the sense that $v_i = 1$ if and only if $i \in J$.

4.2. The matrix ensemble

4.2.1. Correlations. We have implemented the simplest QS described in [15] in order to analyse the output matrix ensemble. We attempted to look for correlations in the presence/absence of different primes in the set of divisors of the variables y_i. Specifically, we checked that there is virtually no correlation between rows of the matrix: we have taken one such output matrix (resulting from the factorization of a product of two 20 digit primes) and computed the covariance between the corresponding spin variables s_1, s_2 of two rows r_1, r_2, the averages being taken along different columns,

$$\langle s_1 s_2 \rangle - \langle s_1 \rangle \langle s_2 \rangle.$$

Once repeated for all $r_1 < r_2$, we found that all pairs have correlations in the interval 0 ± 0.06, a proportion of 0.9999 pairs having correlations in 0 ± 0.02.

4.2.2. Dependence on 'factorization hardness'. We then examined the dependence of the resulting distributions of 1s per row on the 'factorization hardness' of the number n. Typically (depending on the algorithm) the complexity of the factorization depends on the size of the smallest prime divisor of n:[7] for instance, *trial division* ends in exactly this number of steps. It was conjectured that this would be reflected in the structure of the output matrix.

We have constructed 25 numbers n with different factor sizes (from now on, factor type 10 + 10 + 10 will mean a 30 digit number constructed as a product of three 10 digit primes) organized as follows:

- 5 of type 20 + 20
- 5 of type 10 + 30
- 5 of type 13 + 13 + 13
- 5 of type 10 + 10 + 10 + 10
- 5 of type 5 + 5 + 5 + 5 + 5 + 5 + 5 + 5.

All 25 numbers differed between them by less than 0.01%. We then made QS compute the factorization matrices, with a factor base of size 1500. This value for the size of the factor base has been chosen experimentally in order to minimize the sieving phase duration. The resulting matrices were of size 1500×1510 and were then post-processed in order to remove rows and columns with a single 1. The final size is thus reduced to about 200 columns and rows. The resulting distributions of 1s per row are plotted in figure 5, showing very little variation. They can be very well described by a unique distribution, which is substantially a power law with some small deviations in the range of type-2 and type-3 rows. The best fit in the region $X > 3$ gives an exponent ~ -2.2.

Our conclusion is that statistical properties of the resulting matrix do not depend on the factorization hardness. The bottleneck for factorizing a large hard number is mainly determined by the time required by QS to generate the matrix, which indeed strongly depends on the size of the smallest factor. In the rest of the paper we will analyse the solving phase, assuming the factorization matrix to have uncorrelated rows and the number of 1s per row to be a random variable extracted from the distribution in figure 5. These two assumptions have been experimentally verified.

4.3. Linear solving methods

Plain standard Gaussian elimination execution time is cubic in the size of the matrix (our matrices are almost square). Fortunately, we can pack 32 matrix entries in a single 4 byte word,

[7] This is why in RSA we choose $n = p \cdot q$ with $p, q \simeq \sqrt{n}$.

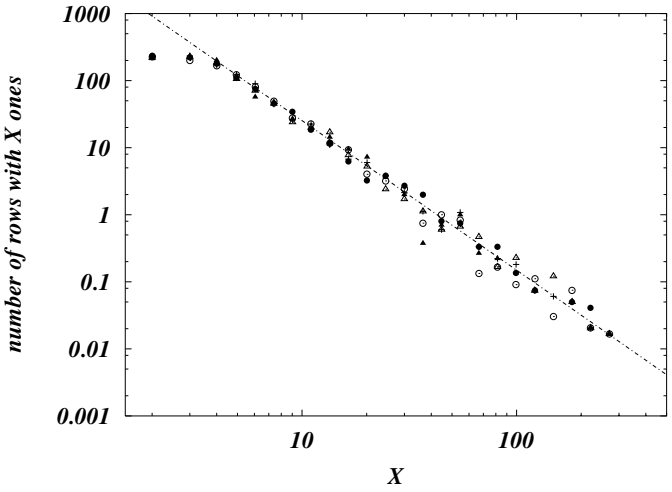

Figure 5. Probability distribution of the number of 1s per row in five different matrices of size around 1300. The line is the best power law fit on $X > 3$ data, giving an exponent ~ -2.2.

and then the sum operation is implemented as the low-cost bitwise logical XOR operation, saving a factor 32 in time.

As also the matrix is very sparse, instead of keeping all of it in memory, we can memorize only the position of 1s. This forbid us to use the factor-32 trick, but allows us to do the first steps very quickly. At some time in the Gaussian elimination process (typically more than half of the process), the remaining (non-eliminated) part will be very dense, and then it will be convenient to switch to the standard method above. This is what was done in the solving phase of RSA129.

Another option is to use in one of the stages an iterative algorithm, such as the discrete Lanczos. The Lanczos method has the advantage of having a stable $\mathcal{O}(N^2)$ total time for a sparse matrix, but finds only one solution (or a prefixed quantity in the block-Lanczos variant) instead of all of them. For factorization this is not a problem, because we need only a few solutions to have a reasonable chance. This is the method that was used in the solving phase of RSA155.

4.4. Power law distributed $\{a_k\}$: phase diagram and comparison with real application data

The previous analysis leads to the construction of matrices whose density of nonzero entries follows quite well a power law distribution with light deviations due to rows with a small number of 1s and a cut-off, $k_{\max}$, of some hundreds. Then we use the following distribution in the analytical treatment:

$$a_2 = a \tag{18}$$

$$a_k = \epsilon k^{-s} \qquad \text{for} \quad 3 \leqslant k \leqslant k_{\max} \tag{19}$$

where ϵ is a normalizing factor equal to $(1-a)/\sum_{k=3}^{k_{\max}} k^{-s}$. The factorized integers considered in the previous section lead to an exponent $s \simeq 2.2$ and to a nonzero support up to $k_{\max} \sim 200$. The choice of keeping a_2, and only a_2, as an independent parameter is dictated by the very difference in the physical behaviour of 2-spin terms and k-spin terms with $k > 2$.

The study of the phase diagram in the control parameter γ for choices of a, s and $k_{\max}$ retrieved from real data reveals a nontrivial behaviour.

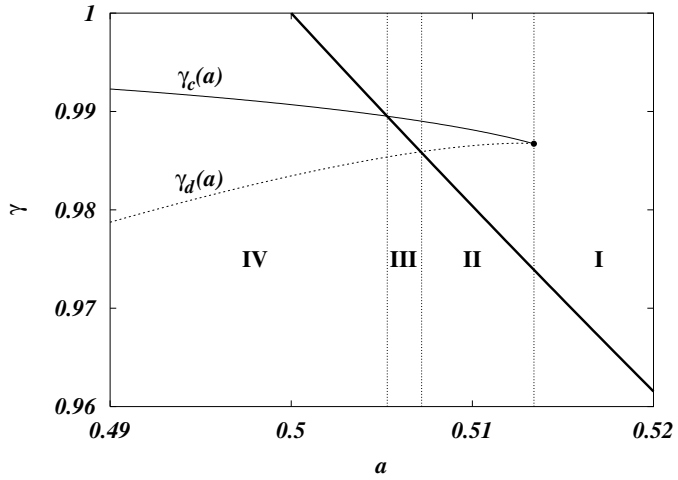

Figure 6. Phase diagram (a, γ) for a typical choice of $s = 2.2$ and $k_{\max} = 200$. The bold line $1/(2a)$ represents a continuous transition, while $\gamma_d(a)$ and $\gamma_c(a)$ correspond, respectively, to the spinodal and the critical lines of a first-order transition. The dot marks the origin of these lines.

In figure 6 we show the phase diagram for $s = 2.2$ and $k_{\max} = 200$. Only part of the entire phase diagram ($a \in [0, 1]$, $\gamma \in [0, 1]$) is shown for clarity. The lines further go on smoothly outside the drawn portion.

If a is high enough, we are in the rightmost region I of the phase diagram, where algorithms smoothly find solutions to the system and do not undergo any critical slowing down. Indeed, crossing the bold iperbole $\gamma = 1/(2a)$ given by the condition $\partial G(m)/\partial m|_{m=0} = 0$, the system undergoes a continuous transition in the order parameter m, representing the fraction of variables taking the same value in all the solutions. The problem of finding solutions is always easy, as for the case $\{a_2 = 1; a_{k\neq 2} = 0\}$ explained before.

Decreasing a we meet a first intermediate region II, where the birth of a meta-stable nontrivial saddle-point solution at $\gamma = \gamma_d(a)$ is given by the solution of equation (8). However, algorithms should not be much affected by this meta-stable state, because the system starts magnetizing continuously, before crossing the bold line. Increasing γ up to the critical value $\gamma_c(a)$ one meets a first-order transition, where the magnetization, which was already nonzero, undergoes a further jump.

The second central region III shows an inversion between $\gamma_d(a)$ and the bold line $1/(2a)$. These two intermediate regions have not been exhaustively studied yet, because real data all fall in the leftmost one. The shape of the central part of the phase diagram is very sensitive to the choice of the control parameters s and $k_{\max}$, as shown in figure 7.

The $\gamma_c(a)$ curve in the second and third regions is found solving

$$S(m^*, \gamma_c) = S(m_*, \gamma_c) \tag{20}$$

where m_* is the smallest positive solution to $G(m) = 0$, which corresponds to the magnetization of the ferromagnetic state arisen from the second-order transition (bold line). The points of crossing showing the onset of different regions, from right to left, are found respectively as $\frac{\partial G(m)}{\partial m} = 0$ and $S(m^*, \gamma) = S(m_*, \gamma)$, $\frac{\partial G(m)}{\partial m} = 0$ and $\gamma = \frac{1}{2a}$ and $S(m^*, \gamma) = S(0, \gamma)$ and $\gamma = \frac{1}{2a}$.

The leftmost part IV shows the typical behaviour described in [8]. Increasing γ the system never reaches the continuous transition on the bold line, but it undergoes a first dynamical

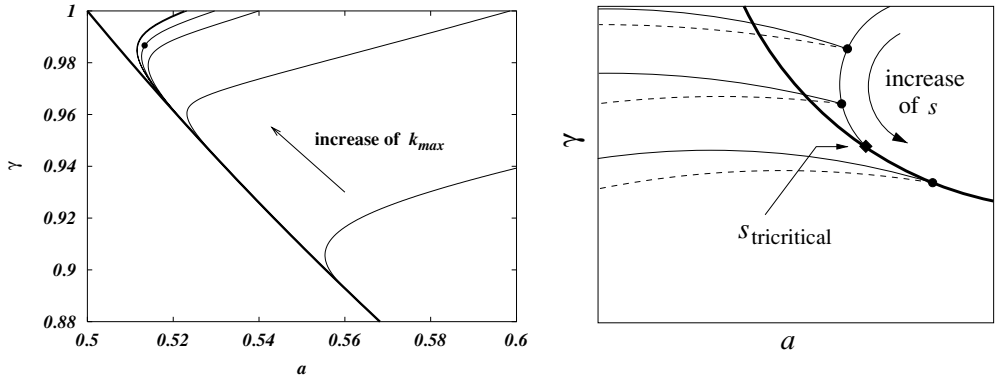

Figure 7. Dependence on s and $k_{\max}$ of the origin of first-order critical lines. The bold curve is the continuous phase transition $\gamma = 1/(2a)$. Each solid bell-shaped curve in the left plot is the ensemble of such origins, defined as the point where, decreasing a, another nontrivial solution to the saddle-point equations appears. Each curve from right to left is indexed by a different value of $k_{\max} = 10$, 30, 100, 200, 1000, 2000, 10 000. Each point on the curve corresponds to a particular value of s (the dot is for $s = 2.2$ and $k_{\max} = 200$ as in figure 6). Along the curve s increases for decreasing γ (see right plot). From each point of the curve originate the two first-order critical lines shown for $s = 2.2$ and $k_{\max} = 200$ in figure 6, and pictorially drawn for different s values in the right plot. When the origin joins the second-order iperbole the system is at a tricritical point. $s_{\text{tricritical}}$ scales very rapidly with $k_{\max}$ converging to ∼2.73–2.74 already for $k_{\max} \sim 100$.

transition at $\gamma_d(a)$ and a second thermodynamical one at $\gamma_c(a)$, found via equation (20) with $m_* = 0$ since we are still below the second-order transition line. Configurational entropy is nonzero between $\gamma_d(a)$ and $\gamma_c(a)$ and solving algorithms are affected by it. There are typically other spinodal lines in the phase diagram, but they always correspond to sub-optimal solutions, and were, therefore, not shown in the picture.

In real data the fraction of 2-variable equations is typically of the order of 0.2 and $\gamma \simeq 1$. So we always work deep into phase IV where, during the solving procedure, the system undergoes a first dynamical transition, that corresponds to a slowing down of the solving algorithms, before finding solutions.

5. Conclusions

We have analysed the behaviour of different types of polynomial algorithms in the solutions of large-scale linear systems over finite fields. The connection between memory requirements and clustering phase transitions has been made clear on both artificially generated problems as well as on 'real-world' applications. While the role of the dynamical glass transition in the local search algorithm was already well known (trapping in local minima), we have provided a clear example of the role of such a type of glass transition in global dynamical processes which are guaranteed to converge to the global optimum in some polynomial time. The memory catastrophe found in such cases constitutes a concrete limitation for the performance of single-machine programs.

References

[1] Garey M and Johnson D S 1979 *Computers and Intractability: A Guide to the Theory of NP-Completeness* (San Francisco, CA: Freeman)
Papadimitriou C H 1994 *Computational Complexity* (Reading, MA: Addison-Wesley)

[2] Alava M J *et al* 2001 *Phase Transitions and Critical Phenomena* vol 18 ed C Domb and J L Lebowitz (San Diego, CA: Academic)
[3] Cormen T H, Leiserson C E and Rivest R L 1990 *Introduction to Algorithms* (Cambridge, MA: MIT Press)
[4] Rivest R L, Shamir A and Adleman L 1978 A method for obtaining digital signatures and public-key cryptosystems webpage http://mit.edu/˜rivest/rsapaper.ps
[5] Kolchin V F 1999 *Random Graphs* (Cambridge: Cambridge University Press)
[6] Sourlas N 1994 Statistical mechanics and error-correcting codes *From Statistical Physics to Statistical Inference and Back: Proc. Cargèse 1992* ed P Grassberger and J P Nadal (Dordrecht: Kluwer) pp 195–204
[7] Broder A Z, Frieze A M and Upfal E 1993 *Proc. 4th Ann. ACM-SIAM Symp. on Discrete Algorithms* p 322
[8] Ricci-Tersenghi F, Weigt M and Zecchina R 2001 *Phys. Rev.* E **63** 026702
[9] Leone M, Ricci-Tersenghi F and Zecchina R 2001 *J. Phys. A: Math. Gen.* **34** 4615
[10] Mézard M, Ricci-Tersenghi F and Zecchina R 2002 *Preprint* cond-mat/0207140
[11] Cocco S, Dubois O, Mandler J and Monasson R 2002 *Preprint* cond-mat/0206239
[12] Leone M 2002 *PhD Thesis* SISSA, Trieste, in preparation
[13] Franz S, Leone M, Ricci-Tersenghi F and Zecchina R 2001 *Phys. Rev. Lett.* **87** 127209
[14] Schaefer T J 1978 *Proc. 10th STOC (San Diego, CA)* (New York: ACM) p 216
Creignou N, Daude H and Dubois O 2001 *Preprint* DM/0106001
[15] Pomerance C and Goldwasser S 1990 *Cryptology and Computational Number Theory (AMS Proc. Symp. in Applied Mathematics vol 42)* pp 27–48