Journal of Statistical Physics, Vol. 126, Nos. 4/5, February/March 2007 (© 2007)
DOI: 10.1007/s10955-006-9255-z

Some Spin Glass Ideas Applied to the Clique Problem

Antonio Iovanella,[1] Benedetto Scoppola,[2] and Elisabetta Scoppola[3]

Received April 28, 2006; accepted November 3, 2006
Published Online: January 18, 2007

In this paper we introduce a new algorithm to study some NP-complete problems. This algorithm is a Markov Chain Monte Carlo (MCMC) inspired by the cavity method developed in the study of spin glass. We will focus on the maximum clique problem and we will compare this new algorithm with several standard algorithms on some DIMACS benchmark graphs and on random graphs. The performances of the new algorithm are quite surprising. Our effort in this paper is to be clear as well to those readers who are not in the field.

KEY WORDS: Markov Chain Monte Carlo, Clique problem

Be careful men
Search every cook and nanny
Uh, hook and granny
Uh, crooked fan...
uh, search everywhere!
Doc (Snow White and the seven dwarfs,
Walt Disney, 1937)

CONTENTS

1 INTRODUCTION **896**
1.1 Definitions 896
1.2 The Case of Random Graphs 897
1.3 The Statistical Mechanics Approach 898

[1] Dipartimento di Ingegneria dell'Impresa, University of Rome "Tor Vergata," Via del Politecnico, 1-00133 Rome, Italy; e-mail: iovanella@disp.uniroma2.it.
[2] Dipartimento di Matematica, University of Rome "Tor Vergata," Via della Ricerca Scientifica - 00133 Rome, Italy; e-mail: scoppola@mat.uniroma2.it.
[3] Dipartimento di Matematica, University of Rome "Roma Tre," Largo San Murialdo, 1 - 00146 Rome, Italy; e-mail: scoppola@mat.uniroma3.it.

2 A HAMILTONIAN FOR THE CLIQUE PROBLEM, THE VERTEX COVERING AND THE INDEPENDENT SET 900

3 SOME ALGORITHMS FOR THE CLIQUE PROBLEM 901
3.1 A Greedy Algorithm, $\mathcal{G}$ 902
3.2 A Dismantling Algorithm, $\mathcal{D}$ 902
3.3 A Monte Carlo Algorithm $\mathcal{MC}$ 902

4 A NEW ALGORITHM INSPIRED BY THE CAVITY METHOD, $\mathcal{C}$ 903
4.1 Implementation of the Algorithm $\mathcal{C}$ and Some Remarks on its Mixing Time 904

5 NUMERICAL COMPARISON 907
5.1 Experimental Details 907
5.2 Numerical Results on DIMACS Benchmark Graphs 907
5.3 Numerical Results on Random Graphs 913

1. INTRODUCTION

In the last years Mezard, Parisi, Zecchina[12,13] introduced a class of optimization algorithms to deal with K-satisfiability problems. Their strategy was based on the cavity method introduced in spin glass theory a long time ago and in particular on its zero-temperature version, more recently developed in Ref. 10. An important ingredient in their approach seems to be the locally tree-like structure of the interaction graph.

In the case of the clique problem, i.e., the study of the maximal complete subgraph of a given graph G, we expect to be very far from a tree-like structure of the interaction graph even locally, for instance when G is a random graph. We introduce in this paper a new algorithm to treat this problem, based again on the cavity method but in a completely different way.

This algorithm represents a first step in the application of the cavity idea that will be developed in a forthcoming paper. On the other hand this algorithm is sufficiently simple so that its behavior can be studied at least on random graphs providing some explanation of the difficulty of the problem. The algorithm introduced in this paper represents an heuristical search of cliques in the sense that the optimality of the result is not guaranteed. For a recent review on the numerical approach to the clique problem see e.g., Ref. 3 and references therein.

1.1. Definitions

Let $G = (V, E)$ be a graph. A graph g is a *subgraph* of G, $g \subseteq G$, if its vertex set $V(g) \subseteq V$ and its edges $E(g) \subseteq E$. For any $A \subset V$ we denote by $G[A]$

the *graph induced by* A in G:

$$G[A] = (A, E(G[A])), \quad \text{with} \quad E(G[A]) := \{(i, j) \in E : i, j \in A\} \tag{1}$$

We will denote by $\mathcal{K}(G)$ the set of *complete subgraphs* or *cliques* of G and by $\mathcal{M}ax\mathcal{C}l(G)$ the set of *maximum cliques* in G:

$$\mathcal{M}ax\mathcal{C}l(G) := \{g \in \mathcal{K}(G) : \ |V(g)| = \max_{g' \in \mathcal{K}(G)} |V(g')|\} \tag{2}$$

where $|B|$ denotes the cardinality of the set B.

We call *clique number* of the graph G, $\omega(G)$, the cardinality of the vertex set of any maximum clique in G, i.e., $\omega(G) = |V(g)|$ with $g \in \mathcal{M}ax\mathcal{C}l(G)$.

There are several versions of the problem of the determination of the clique number and of the maximum clique set of a given graph G. We recall here the most cited form.

> **Clique problem:** given a graph $G = (V, E)$ and a positive integer $k \leq |V|$, does G contain a complete subgraph of size k or more? That is, does $\omega(G) \geq k$ hold?

As it is well known (see Ref. 5) the clique problem is a NP-complete problem. There are other famous NP-complete problems equivalent to the clique problem as the vertex covering and the independent set, defined as follows:

> **Vertex covering:** given a graph $G = (V, E)$ and a positive integer $k \leq |V|$, is there a vertex cover of size k or less for G, i.e., a subset $V' \subseteq V$ with $|V'| \leq k$ such that for each edge $(u, v) \in E$ at least one of u and v belongs to V'?
> **Independent set:** given a graph $G = (V, E)$ and a positive integer $k \leq |V|$, does G contain an independent set of size k or more, i.e., a subset $V' \subseteq V$ such that $|V'| \geq k$ and such that no two vertices in V' are joined by an edge in E?

The equivalence of these problems is proved for instance in Ref. 5 lemma 3.1 pg.54.

1.2. The Case of Random Graphs

Consider the set $\mathbf{G}(n, d)$ of random graphs with fixed density d, i.e. of graphs $G(V, E)$ having as vertex set $V = \{1, 2, \ldots, n\}$ and in which the edges are chosen independently with probability d.

To study the size of the largest clique of a graph $G(V, E) \in \mathbf{G}(n, d)$ one can argue as follows. Let Y_r be the number of complete subgraph with r vertices in a

graph $G(V, E) \in \mathbf{G}(n, d)$. It is immediate to show that

$$E(Y_r) = \binom{n}{r} d^{\binom{r}{2}} \tag{3}$$

Let us consider the value $r_0(n)$ of r such that $E(Y_r) = 1$. Writing (3) in terms of Stirling approximation and denoting $b = 1/d$ we have that such value $r_0(n)$ is given by

$$r_0(n) = 2\log_b n - 2\log_b \log_b n + 2\log_b(e/2) + 1 + o(1) \tag{4}$$

The clique number $\omega(G)$ of a graph $G(V, E) \in \mathbf{G}(n, d)$ tends, for $n \to \infty$, to be very near to $r_0(n)$. More precisely, it is possible to prove the following result (see Ref. 2): for almost all the graphs $G \in \mathbf{G}(\mathbb{N}, d)$ there is a constant $m_0(G)$ such that for all $n \geq m_0(G)$ and for almost all G_n subgraph of G with vertex set $|V| = n$

$$|\omega(G_n) - 2\log_b n + 2\log_b \log_b n - 2\log_b(e/2) - 1| \leq \frac{3}{2} \tag{5}$$

Despite the fact that the asymptotic value of $\omega(G_n)$ has such a small variability, it is well known that the large cliques of a random graph are very difficult to find. This is due to the fact that the expression of $E(Y_r)$, which has its maximum for an r that is roughly $r_0(n)/2$, decreases very rapidly when $r > r_0(n)/2$. Hence, while it is easy (e.g. with a greedy algorithm) to find cliques whose size is of the order of $\log_b n$, the probability that one of such cliques is a subset of a clique with the size $(1 + \varepsilon)\log_b n$ is of the order of $n^{-\alpha(\varepsilon)\log n}$ for all $\varepsilon > 0$, and hence is more than polynomially small (see also Ref. 6).

This difficulty in finding large clique of random graphs has a numerical evidence even for n quite small, as it will be shown later.

1.3. The Statistical Mechanics Approach

We recall here very briefly the main ideas of the statistical mechanics approach to combinatorial optimization problems.

The cost function of the optimization problem (OP) can be understood as the energy function $H(x)$, usually called *Hamiltonian*, of a statistical mechanics (SM) model where instances of the OP are considered as configurations $x \in \mathcal{X}$ of the SM model. The optimal configurations correspond to the ground states in the SM language. (See for instance[11]). Ground states in SM are the configurations where the Gibbs measure $\pi(x) = \frac{1}{Z} e^{-\beta H(x)}$ is concentrated in the limit of zero temperature ($\beta \to \infty$, being β the inverse temperature); the normalization constant Z is usually called *partition function*. This means that to determine the ground states is sufficient to perform a random sampling at low temperature. To this purpose we can apply the Monte Carlo method. The main idea of this method

is to define a *Markov chain Monte Carlo* (MCMC) on the configuration space $\mathcal{X}$, with transition probabilities $P(x, x')$ such that the transition probability in n steps, $P^n(x, x')$, of the chain converges to $\pi(x')$ as $n \to \infty$. This convergence is due to the ergodic theorem if for instance the transition probabilities satisfy a *detailed balance condition* w.r.t. the Gibbs measure π:

$$\pi(x)P(x, x') = \pi(x')P(x', x) \tag{6}$$

The strategy of the MCMC method is then the following

- start from a configuration x_0
- look at the random evolution of the chain starting from it, $x_0, x_1, \ldots x_n$, for a "sufficiently long time" n
- for the final state x_n we have $P(x_n = x) \sim \pi(x)$.

The main difficulty in applying this procedure is due to metastable states. Indeed local minima of the energy $H(x)$ can capture the evolution x_t of the chain for very large time intervals if the temperature is low. So the main problem in applying MCMC method is to define what "sufficiently long time" means.

A strategy to escape the problem of metastable states is to change the temperature during the evolution of the chain. This is known as *simulated annealing*. Since for high temperature the process leaves local minima much easily, one can look at a suitable annealing in order to avoid to remain captured in metastable states. See for instance[9] for the use of simulated annealing in optimization problems.

From a rigorous point of view the main point in applying the MCMC method is to estimate the *mixing time* of the chain, that is the time n necessary to have that $P(x_n = x)$ and $\pi(x)$ are sufficiently close each other, uniformly in x_0. (See for instance[7] for precise definitions.)

As an example for the clique problem on a graph $G \in \mathbf{G}(n, 1/2)$ we can consider as in Ref. 6 the following MCMC. The state space $\mathcal{X}$ of the chain is the collection of all cliques in G. To each clique $x \in \mathcal{X}$ we associate a weight $w(x) = \lambda^{|x|}$ where $|x|$ denotes the number of vertices of x and $\lambda \geq 1$ is a real parameter. We can describe this weight in terms of a Gibbs measure $\pi(x) = \frac{w(x)}{Z}$ with $H(x) = -|x|$ and $\lambda = e^{\beta}$. The transition probability $P(x, x')$ is different from zero only if the cliques x and x' have a symmetric difference (as sets of vertices) less or equal to one. In this case if $x' \supset x$ we put $P(x, x') = \frac{1}{n}$ and if $x' \subset x$ we put $P(x, x') = \frac{1}{\lambda n}$. The probability $P(x, x)$ is obtained by normalization. It is immediate to verify that these transition probabilities satisfy the detailed balance condition (6).

For this dynamics Jerrum proves that there exists an initial state from which the expected time to reach a clique of size at least $(1 + \varepsilon) \log_2 n$ is super-polynomial in n. The crucial point in this proof is to show that there are few cliques that can grow up to this size $(1 + \varepsilon) \log_2 n$. More precisely a clique of size k is called

m-gateway if there exists a path of the chain going from this clique to a clique of size m through cliques of size at least k, then it is proved in Ref. 6 that the density of m-gateways in the set of k-cliques is super-polynomially small for $k = \lceil (1 + \frac{2}{3}\varepsilon) \log_2 n \rceil$ and $m = \lceil (1 + \varepsilon) \log_2 n \rceil$. Due to the fact that m-gateways have to be visited in reaching cliques larger or equal to m, then these m-gateways represent a bottleneck for the dynamics and their low density can be used to prove that the mixing time is super-polynomial in n.

2. A HAMILTONIAN FOR THE CLIQUE PROBLEM, THE VERTEX COVERING AND THE INDEPENDENT SET

We consider the space $\mathcal{X} := \{0, 1\}^V$ of lattice gas configurations on V; on the configuration space (or state space) $\mathcal{X}$ we define an Ising Hamiltonian with an antiferromagnetic interaction between non-neighbor sites:

$$H(\sigma) := \sum_{(i,j)} J_{ij}\sigma_i\sigma_j - \frac{h}{2}\sum_{i\in V}\sigma_i \tag{7}$$

where (i, j) is an unordered pair in $V \times V$ and

$$J_{ij} = \begin{cases} 0 & \text{if} \quad (i, j) \in E \\ 1 & \text{if} \quad (i, j) \notin E \end{cases} \tag{8}$$

and $h > 0$.

It is easy to prove that if $h < 1$ then the minimal value of $H(\sigma)$ is obtained on configurations with support on the vertices of a maximum clique. First of all we prove that $H(\sigma)$ is minimal on configurations σ such that $G(\sigma) \in \mathcal{K}(G)$. We denote with the same letter a configuration and its support; for instance when we write $i \in \sigma$ we mean a site i in the support of σ. Indeed for every σ such that $G(\sigma) \notin \mathcal{K}(G)$ we can write $\sigma = C \cup A$ with $G(C)$ a maximum clique in $G(\sigma)$ and $|A| \geq 1$, then for any $i \in A$ we have $H(\sigma) > H(\sigma\backslash i)$. This is due to the fact that

$$H(\sigma) = H(\sigma\backslash i) + \sum_j J_{ij}\sigma_j - \frac{h}{2} \tag{9}$$

and if $G(C)$ is a maximum clique in $G(\sigma)$ then $\sum_j J_{ij}\sigma_j \geq \sum_{j\in C} J_{ij} \geq 1$. As a second step we note that if σ is such that $G(\sigma) \in \mathcal{K}(G)$ then $H(\sigma) = -\frac{h}{2}|\sigma|$, so that we can immediately conclude that H is minimal on the maximum cliques.

If we consider the opposite interaction:

$$\bar{J}_{ij} = \begin{cases} 0 & \text{if} \quad (i, j) \notin E \\ 1 & \text{if} \quad (i, j) \in E \end{cases} \tag{10}$$

then the same Hamiltonian (7) with interaction $\bar{J}$ is minimal on configurations with zeros on a minimal vertex cover and ones on the maximum independet set.

In the case of a random graph G, i.e., when the interaction variables J_{ij} are i.i.d.r.v., the Hamiltonian (7) is similar to the Hamiltonian of the SK model. The main differences are that our configurations are in lattice gas variables instead of spin variables and the interaction variables have no zero mean. Instead of a symmetry property we have now a control on the sign of the interaction term of the Hamiltonian.

Note that we are considering interactions $O(1)$ instead of $O(1/\sqrt{n})$ as in SK, but these interactions vanish for $(i, j) \in E$. This means that when $\sum_{(i,j)} J_{ij} = O(n^2)$, as in the random case with fixed density, we are considering a non-diluted model.

3. SOME ALGORITHMS FOR THE CLIQUE PROBLEM

We will not furnish here a complete review of algorithms for the clique problem. We only want to recall that there are two different families of algorithms

- the complete methods
- the heuristic methods

The complete methods are those algorithms that use different clever strategies to perform a complete analysis of all the possible subgraphs. Even when all the possible subgraphs are not actually enumerated, the time that these algorithms take grows very fast with the size of the graph, fact made evident because the clique problem is NP.

The heuristic methods are those algorithms that do not ensure that the result is optimal, providing in this way only a lower bound for the clique number. An exemple of such algorithms is the MCMC discussed in Ref. 6. These are usually faster algorithms.

In this section we define three different heuristic algorithms for the clique problem that will be used for the numerical comparison developed in the final section. The first and the second are "standard" algorithms; the third algorithm is a MCMC defined by means of the Hamiltonian (7).

In the discussion contained in the numerical comparison we will also consider three more algorithms: two of them are complete, and are the Cliquer algorithm (see Ref. 14) and the algorithm introduced in Ref. 15 by Régin based on a very efficient application of constraint programming. The last is one of the best known heuristic algorithms, introduced by Battiti and Protasi in Ref. 1.

3.1. A Greedy Algorithm, $\mathcal{G}$

The first algorithm we introduce is a fast and greedy heuristic, denoted from now on by $\mathcal{G}$. The underlying idea is the following: start from a configuration σ with $\sigma_i = 0, \forall i \in V$ and then select at random a vertex j, set $\sigma_j = 1$ and then delete all its non adjacent vertices. In the next step another vertex is selected at random among the remaining vertices and again all its non-adjacent vertices are deleted. The process stops when it is not possible to select other vertices, i.e., a maximal complete subgraph is found, i.e., a clique not strictly contained in other cliques.

3.2. A Dismantling Algorithm, $\mathcal{D}$

The second algorithm, denoted by $\mathcal{D}$ in the following, is another fast heuristic. It starts with an initial configuration σ that has ones everywhere. The algorithm considers, at each step, the degree of each vertex i with $\sigma_i = 1$ and selects the one (say j) with the smallest degree. Then it sets $\sigma_j = 0$ and decreases by one unit the degree of all its adjacent nodes in the graph and repeats the procedure until the minimum value of all the degrees is $k - 1$ where k is the number of sites in σ with $\sigma_i = 1$, i.e., the sites of a clique of cardinality k. Note that in principle the resulting clique could be not maximal.

The rationale of this algorithm is to start from the whole graph and then, at each step, dismantle it vertex by vertex until a clique is found.

3.3. A Monte Carlo Algorithm $\mathcal{MC}$

We can apply the ideas developed in Sec. 1.3 to the Hamiltonian defined in Sec. 2 for the clique problem. For clarity we consider the Metropolis choice: for $\sigma' \neq \sigma$ we take

$$P(\sigma', \sigma) = q(\sigma', \sigma) e^{-\beta[H(\sigma) - H(\sigma')]_+}, \tag{11}$$

where $[\cdot]_+$ denotes the positive part and $q(\sigma', \sigma)$ is a symmetric, positive connectivity matrix independent of β with $q(\sigma', \sigma) > 0$ only if σ and σ' are different in a single site.

We note that in the limit $\beta \to \infty$ and $h \in (0, 1)$ fixed, starting from the configuration which is zero everywhere, this algorithm is equivalent to the greedy algorithm since $P(\sigma, \sigma') = 0$ if $H(\sigma') > H(\sigma)$. Thus $\{\sigma(t)\}_{t \in \mathbb{N}}$ is a growing sequence of complete graphs. In the case $\beta \to \infty$ but $h \to 0$ as $\frac{1}{\beta}$ we see that this Monte Carlo algorithm is equivalent to the Jerrum algorithm on cliques recalled as an example in Sec. 1.3.

4. A NEW ALGORITHM INSPIRED BY THE CAVITY METHOD, $\mathcal{C}$

In this section we introduce a new heuristic algorithm to find maximum cliques of a graph. The key idea is inspired by the notion of *cavity field* introduced in statistical mechanics to analyze the ground states, that is configurations minimizing the energy.[11] The cavity method at zero temperature is described in detail in Ref. 10 in the case of a spin glass on a lattice with a local tree like structure. This method is equivalent to the replica method and can be used at different levels of approximation corresponding to the replica symmetric solution and to the one step replica symmetry breaking level. The main idea is to compute in the limit of infinite number of spins the value of the energy density of the ground state by an iterative procedure. Indeed one can study the effect of the addition of a spin or of a bond to the system looking for equations for the corresponding average energy shift.

We do not use the cavity method in our algorithm but we use the idea that if you select a spin the effect of the other spins can be described in terms of a local field, that we will call cavity field, as in the case of the cavity method.

More precisely, consider the Hamiltonian defined in (7) and consider the canonical ensemble, i.e., the set of configurations $\sigma \in \mathcal{X}$ such that $\sum_{i \in V} \sigma_i = k$. Up to a constant we have that $H(\sigma) = \sum_{(i,j)} J_{ij}\sigma_i\sigma_j$. If for each $i \in V$ we define the cavity field:

$$h_i(\sigma) = \frac{1}{2}\left[\sum_{j:j\neq i} J_{ij}\sigma_j + h(1-\sigma_i)\right] \tag{12}$$

we have immediately that $H(\sigma) = \sum_{i\in V} h_i(\sigma)\sigma_i$. For a given choice of the fields $\{h_i\}_{i\in V}$ the minimal energy is clearly obtained on the configurations with support on the sites corresponding to the k minimal values of h_i. But here the cavity fields depend on the configuration itself and then it is more difficult to determine the ground states. To this purpose we introduce a new Hamiltonian:

$$H(\sigma, \sigma', h, k) = \sum_{(i,j)} J_{ij}\sigma_i\sigma'_j + \frac{h}{2}\left(k - \sum_i \sigma_i\sigma'_i\right) \tag{13}$$

defined on pairs of configurations σ, σ' such that $\sum_i \sigma_i = \sum_i \sigma'_i = k$, with $k \in \mathbb{N}$ and $h > 0$. The hamiltonian can be rewritten in terms of the interaction of the configuration σ' with each site i (cavity field h_i) in the following way

$$H(\sigma, \sigma', h, k) = \frac{1}{2}\left[\sum_{i,j:i\neq j} J_{ij}\sigma_i\sigma'_j + h\left(k - \sum_i \sigma_i\sigma'_i\right)\right] = \sum_i h_i(\sigma)\sigma'_i \tag{14}$$

with $h_i = h_i(\sigma)$ defined in (12). Hence the cavity field h_i in the site i represents the number of sites j with $\sigma_j = 1$ that are not nearest neighbors of the site i plus a

contribution h that is present when the configuration σ is not supported on the site i. Note that $H(\sigma, \sigma, h, k)$ corresponds to the Hamiltonian (7) in the framework of the canonical ensemble corresponding to k. We also want to stress that this new Hamiltonian is non-negative and if $k \leq \omega(G)$ its value is zero (so minimal) only on pairs of configurations σ, σ' such that $\sigma = \sigma'$ with support on a clique with k vertices.

The idea of the algorithm is the following: start from a random configuration σ with fixed k, and choose a new configuration σ' picking randomly k sites, each site having a relative weight $w_i = e^{-\beta h_i(\sigma)}$ for some $\beta > 0$, and define for this sites $\sigma'_i = 1$, while for the others $\sigma'_i = 0$. Then repeat this procedure iteratively. After each iteration compute the quantity $H(\sigma, \sigma', h, k)$. This dynamics defines a MCMC on $\mathcal{X}$ that satisfies the detailed balance condition with respect to the stationary measure

$$\Pi_\sigma = \frac{\sum_\tau e^{-\beta H(\sigma,\tau,h,k)}}{\sum_{\tau,\sigma} e^{-\beta H(\sigma,\tau,h,k)}} \tag{15}$$

Indeed since each vertex j is chosen to have $\sigma'_j = 1$ with weight w_i, the transition probability of the process $P(\sigma, \sigma')$ has the following form

$$P(\sigma, \sigma') = \frac{e^{-\beta H(\sigma,\sigma',h,k)}}{\sum_\tau e^{-\beta H(\sigma,\tau,h,k)}} \tag{16}$$

Due to the symmetry of the couplings $J_{ij} = J_{ji}$ we have

$$\Pi_\sigma P(\sigma, \sigma') = \frac{\sum_\tau e^{-\beta H(\sigma,\tau,h,k)}}{\sum_{\tau,\sigma} e^{-\beta H(\sigma,\tau,h,k)}} \frac{e^{-\beta H(\sigma,\sigma',h,k)}}{\sum_\tau e^{-\beta H(\sigma,\tau,h,k)}} \tag{17}$$

$$= \frac{\sum_\tau e^{-\beta H(\sigma',\tau,h,k)}}{\sum_{\tau,\sigma'} e^{-\beta H(\sigma',\tau,h,k)}} \frac{e^{-\beta H(\sigma',\sigma,h,k)}}{\sum_\tau e^{-\beta H(\sigma',\tau,h,k)}} = \Pi_{\sigma'} P(\sigma', \sigma) \tag{18}$$

and therefore Π_σ is the unique stationary measure of our process.

Note that if the parameter β is very large and $k \leq \omega(G)$, the stationary measure is concentrated exponentially in β on the σ's such that there exists a clique supported by the configuration σ: actually if the support of σ is not a clique $H(\sigma, \tau, h, k) > 0$ for all configurations τ and the probability of the configuration σ is exponentially small.

4.1. Implementation of the Algorithm $\mathcal{C}$ and Some Remarks on its Mixing Time

To realize a single step of the Markov chain with transition probabilities defined in (16) we proceed as follows.

1. Starting from a configuration σ, compute the cavity field $h_i(\sigma)$ for each vertex i.
2. To sample the new configuration σ' with probability (16) we perform a Kawasaki-like algorithm $\eta(0), \eta(1), \ldots \eta(s), \ldots$, starting at $\eta(0) = \sigma$. At each step s this Kawasaki procedure is the following: pick randomly a couple of vertices (i, j) such that $\eta_i(s) = 1$ and $\eta_j(s) = 0$ and define $\eta(s)^{(i,j)}$, the configuration obtained by $\eta(s)$ by exchanging the occupation variables in the sites i and j. Then $\eta(s+1) = \eta(s)^{(i,j)}$ with probability $e^{-\beta[h_j(\sigma)-h_i(\sigma)]_+}$. Since $H(\sigma, \eta(s), h, k) = \sum_{i \in V} h_i(\sigma)\eta(s)_i$ we have $H(\sigma, \eta(s+1), h, k) - H(\sigma, \eta(s), h, k) = h_j(\sigma) - h_i(\sigma)$ so that the invariant measure of this Kawasaki chain is

$$\Pi^K_{\sigma'} = \frac{e^{-\beta H(\sigma,\sigma',h,k)}}{\sum_\tau e^{-\beta H(\sigma,\tau,h,k)}} \tag{19}$$

as requested in (16).

Since this measure $\Pi^K_{\sigma'}$ is a product measure we note that step 2, i.e., the Kawasaki procedure, quickly reaches its equilibrium, in a time of order nk. Much more complicated is an estimate for the mixing time of the chain $\mathcal{C}$. Here we can only make some initial remarks on this problem.

First of all we note that the function $H(\sigma(t), \sigma(t+1), h, k)$ is a non-increasing function of t in the limit $\beta \to \infty$ along a typical path $\{\sigma(t)\}_t$ of the chain $\mathcal{C}$. Indeed in the limit of zero temperature, the configuration $\sigma(t+1)$ minimizes the Hamiltonian

$$\min_\sigma H(\sigma(t), \sigma, h, k) = H(\sigma(t), \sigma(t+1), h, k) = H(\sigma(t+1), \sigma(t), h, k) \tag{20}$$

and $\sigma(t+2)$ is such that

$$\min_\sigma H(\sigma(t+1), \sigma, h, k) = H(\sigma(t+1), \sigma(t+2), h, k) \le H(\sigma(t+1), \sigma(t), h, k) \tag{21}$$

So the *trap configurations* for the dynamics $\mathcal{C}$ at zero temperature are the configurations σ such that $\min_\tau H(\sigma, \tau, h, k) = H(\sigma, \sigma, h, k)$. The cavity fields $h_i(\sigma)$ have values in the set $\{\frac{q+rh}{2}\}_{q \in \{0,1,\ldots,k-1\}, r \in \{0,1\}}$ so we can define the different levels of the cavity fields of σ, i.e., for each $q \in \{0, 1, \ldots, k-1\}$ and $r \in \{0, 1\}$ we define $I_{q,r} := \{i \in V : h_i(\sigma) = \frac{q+rh}{2}\}$. The configurations τ minimizing $H(\sigma, \tau, h, k)$ have support on the sites belonging to the lowest levels of the cavity fields $h(\sigma)$. This means that σ is a trap if $h_{max}(\sigma) := \max_{i \in \sigma} h_i(\sigma) < h_j(\sigma)$ for each $j \notin \sigma$. On the other hand, in the case of random graphs, we know the distribution of the cavity field in sites $j \notin \sigma$. Indeed for these sites we have $h_j(\sigma) = \frac{1}{2}(ML_j(\sigma) + h)$ where $ML_j(\sigma)$ denotes the number of missing links from j to the set σ (the support of σ). Due to the fact that $ML_j(\sigma)$ and $ML_{j'}(\sigma)$ are independent variables for $j, j' \notin \sigma$ with a binomial distribution, we also know the distribution of the numbers $|I_{q,1}|$

of sites $j \notin \sigma$ with cavity field $h_j(\sigma) = \frac{q+h}{2}$:

$$P(|I_{q,1}| = l) = \binom{n-k}{l} p^l (1-p)^{n-k-l} \tag{22}$$

where

$$p \equiv p(q,k) := P(ML_j(\sigma) = q) = \binom{k}{q} 2^{-k} \tag{23}$$

in the case of random graph with density $\frac{1}{2}$. The quantity

$$G(\sigma) := \min_{j \notin \sigma} h_j(\sigma) - h_{max}(\sigma) \tag{24}$$

can be called the *gap of the trap*.

If $k = (1+\varepsilon)\log_2 n$ with $\varepsilon > 0$ and if $q \ll k$ we have that for large n

$$P(|I_{q,1}| = 0) = (1-p)^{n-k} \sim 1 - n^{-\varepsilon} \tag{25}$$

so with large probability the lowest levels corresponding to $r = 1$, i.e., to sites not in σ, are empty.

We notice that in order to really leave a trap, we have to change many sites in a single step of the dynamics. If the changes are too few, then they produce configurations immediately coming back to the trap. Indeed starting from a trap σ with gap γ, denote by σ' the configuration obtained in a single step of the dynamics and by l the number of changed sites, i.e., $l = |\{i;\, \sigma_i \neq \sigma_i'\}|$. We have that $|h_i(\sigma) - h_i(\sigma')| \leq \frac{l+h}{2}$ for each site i. So if $l < \frac{\gamma}{2} - h$ we have that the new cavity field $h(\sigma')$ has the lowest levels again containing the sites of σ and so with large probability the dynamics in the following steps will come back in σ.

Again we can apply the Jerrum argument. If σ is almost a clique –i.e., $h_{\max}(\sigma)$ is small but the maximal clique contained in s, say σ_0, is of size $k_0 = (1+\frac{2}{3}\varepsilon)\log_2 n$ and σ_0 is not a k-gateway– with probability near to one we have a gap of σ of order ak with $a < 1$ but strictly positive. This means that to escape the trap we have an energy barrier that is a positive fraction of k^2 since, if σ is not a k-gateway, a number of sites proportional to k has to be changed in the non-empty levels of type $I_{q,1}$.

For this reason we expect that our algorithm has a non-polynomial mixing time of order $n^{a \log n}$. However in the following section we will show that this non-polynomial mixing time becomes evident only when n is very large. On DIMACS random graphs we get better results than the other algorithms.

Moreover we can gain from our analysis of traps a more precise knowledge of the energy landscape, suggesting improvements of our algorithm. This is the

Table I. User times for DIMACS machine benchmarks instances

r100.5	r200.5	r300.5	r400.5	r500.5
0.01	0.09	0.77	4.47	16.83

subject of a further paper where the numerical aspects of the problem will be discussed in more details.

5. NUMERICAL COMPARISON

In this section we briefly give some numerical results on the algorithm introduced in the previous section. In particular we will discuss the performance of our algorithm on two groups of graphs: DIMACS benchmark graphs[8] and random graphs. A more complete numerical analysis will be given in a forthcoming paper.

5.1. Experimental Details

The algorithms, greedy $\mathcal{G}$, dismantling $\mathcal{D}$, Monte Carlo $\mathcal{MC}$ and Cavity $\mathcal{C}$ are implemented in C language and performed on a 2.5GHz Power Mac G5 Quad processors machine with Mac OS X v10.4 Tiger and 8Gb of RAM and compiled with gcc and considering the $-$O2 switch. As required by the rules of the Second DIMACS Implementation Challenge,[8] we provide in Table I the user times in seconds performed by one processor on our computer.

5.2. Numerical Results on DIMACS Benchmark Graphs

The Center for Discrete Mathematics and Theoretical Computer Science (DIMACS) makes available on its web site (`ftp://dimacs.rutgers.edu/pub/challenge/graph/benchmarks`) a suite of 79 benchmark graphs for the maximum clique size problem. Such benchmarks constitute an important base point in order to evaluate the performances of new algorithms in this topic. They were generated by means of different criterions and the set includes:

- Random graph (Cn.d and DSJCn.d, being n the size and d the density);
- Steiner triple graph (MANNn);
- Brockington graph (brock$n$_y, with parameter $y = 1, 2, 3, 4$);
- Sanchis graph (gen$n$_p0.9_x, san$n$_0.y_z, sanr$n$_0.y, with parameters $x = 44, 55, 65, 75$, $y = 5, 7, 9$ and $z = 1, 2, 3$);
- Hamming graph (hamming$x$-y with parameters $x = 6, 8, 10$ and $y = 2, 4$);

- Keller graph (`keller`x, with parameter $x = 4, 5, 6$);
- P-hat graph (`p-hat`n`-`x, with parameter $x = 1, 2, 3$);
- Pardalos graph (`c-fat`n`-`x, with parameter $x = 1, 2, 5, 10$);

For additional details and references the reader could see Ref. 8.

In Table II we report the results for a selection of the 37 instances belonging to the Second DIMACS Implementation Challenge and in Tables III a selection of the remaining 42. The tables are organized as follows. The first column report the name of the instance, the following three columns its characteristics, i.e., number of nodes n, number of arcs m and density d. Successively, we report the results for the cavity algorithm $\mathcal{C}$ in terms of best achieved value and CPU time. The following columns are related to the values achieved by the algorithm presented in Sec. 3. In particular, column $\mathcal{G}$ reports the best value achieved on 100 run of the greedy algorithm and columns $\mathcal{D}$ and $\mathcal{MC}$ report both best achieved clique and CPU time, respectively. Note that for the sake of simplicity we do not report the CPU time for $\mathcal{G}$ because it was always equal to 0.000. The last columns of the tables are related to the results obtained by Régin[15] and Battiti and Protasi.[1] Note that observing the computational times on DIMACS machine benchmarks, the times listed in Ref. 1 have to be divided roughly by 4 in order to be compared with ours, while the times listed in Ref. 15 can be directly compared.

Let us close this section with some final remarks on the performances of the $\mathcal{D}$ and $\mathcal{C}$ algorithms. First of all we want to stress that despite its simplicity, $\mathcal{D}$ performs quite well in many instances, especially when the density is high and exact results are difficult to obtain. As far as $\mathcal{C}$ is concerned we consider its performance quite promising. In particular $\mathcal{C}$ works well on the graphs `brock800_2`, `brock800_4`, `p_hat1000-3`, `p_hat1500-2` where complete algorithms typically fail, see for instance.[15] On such graphs our computational times are quite short. On the other hand there are graphs like some Brockington graphs and some Sanchis graphs, where the performance of $\mathcal{C}$ must be improved. In a forthcoming paper we will analyze in more detail the algorithm $\mathcal{C}$; in particular we intend to vary the parameters β and h that in this first analysis are fixed in a rough way (say $\beta \sim 2,\ h \sim .8$).

Note also that the results obtained by Battiti and Protasi are often similar or better than ours, except in the case of the large random graphs, where our algorithm is much faster.

Apart from the numerical comparisons with other algorithms we think that one of the main interesting aspects of $\mathcal{C}$ is that it is a first attempt to use an algorithm where the basic step computes the new configuration on the basis of a large number of edges of the graph. In some sense $\mathcal{C}$ "sniffs" around before computing the new configuration. Moreover it moves on unfeasible configurations (i.e. uncomplete

Table II. Results for the DIMACS benchmarks (Second challenge set)

					$\mathcal{C}$		$\mathcal{G}$	$\mathcal{D}$		$\mathcal{MC}$		Ref. 15		Ref. 1	
DIMACS benchmarks	n	m	d	$\omega(G)$	k	Time (s)	k	k	Time(s)	k	Time(s)	k	Time(s)	k	Time(s)
C125.9	125	6963	0.898	34	**34**	0.060	23	32	0.000	28	0.640	—	—	34	0.004
C250.9	250	27984	0.899	44	**44**	0.270	29	39	0.000	35	2.660	—	—	44	0.029
C500.9	500	112332	0.900	57	**57**	2.670	36	47	0.000	41	15.340	—	—	57	3.124
C1000.9	1000	450079	0.901	68	**68**	20.970	43	53	0.080	46	77.840	—	—	68	41.660
C2000.9	2000	1799532	0.900	≥80	77	75.760	49	56	0.330	52	371.460	—	—	78	823.358
DSJC500.5	500	62624	0.502	14	13	1.290	9	8	0.010	11	18.770	—	—	13	0.194
DSJC1000.5	1000	499652	0.500	15	**15**	15.580	10	10	0.070	12	95.970	—	—	15	6.453
C2000.5	2000	999836	0.500	≥16	**16**	9.900	12	9	0.310	13	409.530	—	—	16	9.976
C4000.5	4000	4000268	0.500	≥18	**18**	104.020	12	12	1.380	15	1889.440	—	—	18	2183.089
MANN_a27	378	70551	0.990	126	124	6.600	90	117	0.000	110	5.380	126	18.48	126	3.116
brock200_2	200	9876	0.496	12	**12**	0.000	8	8	0.000	10	2.330	12	0.29	12	9.605
brock200_4	200	13089	0.658	17	**17**	0.040	11	12	0.000	14	2.140	17	2.13	17	19.491
brock400_2	400	59786	0.749	29	25	1.050	17	21	0.000	20	9.100	29	7,910.6	29	42.091
brock400_4	400	59765	0.749	33	25	1.340	17	20	0.010	20	9.040	33	6,051.77	33	108.638
brock800_2	800	208166	0,651	≥21	**21**	0.350	14	7	0.020	17	33.050	≥20	—	21	4.739
brock800_4	800	207643	0,650	≥21	**21**	0.610	14	13	0.040	17	333.070	≥20	—	21	6.696
gen200_p0.9_44	200	17910	0.900	44	**44**	0.360	27	31	0.000	33	1.600	—	—	44	0.037
gen200_p0.9_55	200	17910	0.900	55	**55**	0.030	28	35	0.000	41	1.600	—	—	55	0.016
gen400_p0.9_55	400	71820	0.900	≥55	50	0.130	34	29	0.010	40	7.520	—	—	55	1.204
gen400_p0.9_65	400	71820	0.900	≥65	54	0.030	34	32	0.010	40	7.520	—	—	65	0.050
gen400_p0.9_75	400	71820	0.900	≥75	**75**	0.120	36	37	0.010	52	7.530	—	—	75	0.051
hamming8-4	256	20864	0.639	16	14	0.070	10	16	0.000	16	3.200	16	4.19	16	0.003

Table II. Continued

					$\mathcal{C}$		$\mathcal{G}$	$\mathcal{D}$		$\mathcal{MC}$		Ref. 15		Ref. 1	
DIMACS benchmarks	n	m	d	$\omega(G)$	k	Time (s)	k	k	Time(s)	k	Time(s)	k	Time(s)	k	Time(s)
keller4	171	9435	0.649	11	**11**	0.000	8	8	0.000	**11**	1.340	11	0.50	11	0.002
keller5	776	225990	0.752	≥27	23	4.570	17	15	0.030	20	41.880	≥27	—	27	0.171
p_hat300-1	300	10933	0.244	8	**8**	0.350	6	7	0.000	**8**	4.650	8	0.11	8	0.018
p_hat300-2	300	21928	0.489	25	**25**	0.150	16	22	0.000	22	4.980	25	0.59	25	0.006
p_hat300-3	300	33390	0.744	36	**36**	3.160	19	31	0.000	30	4.320	36	40.71	36	0.021
p_hat700-1	700	60999	0.249	11	**11**	2.330	7	7	0.030	10	38.060	11	6.01	11	0.186
p_hat700-2	700	121728	0.498	44	**44**	7.690	24	40	0.030	35	39.110	44	255.79	44	0.028
p_hat700-3	700	183010	0.748	≥62	**62**	13.570	31	58	0.030	47	37.940	≥62	—	62	0.035
p_hat1500-1	1500	284923	0.253	12	**12**	0.780	7	9	0.180	11	221.800	12	480.84	12	30.274
p_hat1500-2	1500	568960	0.506	≥65	**65**	18.130	30	61	0.180	48	224.230	—	—	65	0.158

Table III. Results for the DIMACS benchmarks

					$\mathcal{C}$		$\mathcal{G}$	$\mathcal{D}$		$\mathcal{MC}$		Ref. 15	
DIMACS benchmarks	n	m	d	$\omega(G)$	k	Time(s)	k	k	Time(s)	k	Time(s)	k	Time(s)
brock200_1	200	14834	0.745	21	**21**	1.800	16	16	0.000	17	1.970	21	10.72
brock200_3	200	12048	0.605	15	14	0.360	10	11	0.000	13	2.240	15	0.86
brock400_1	400	59723	0.748	27	25	0.400	16	18	0.010	21	9.150	27	11,340.8
brock400_3	400	59681	0.748	31	25	0.400	16	17	0.000	20	9.200	31	4,472.23
brock800_1	800	207505	0.649	23	21	1.390	15	14	0.040	17	55.970	$\geq$21	—
brock800_3	800	207333	0.649	25	22	0.880	14	14	0.040	18	55.920	$\geq$20	—
c-fat200-1	200	1534	0.077	12	**12**	0.000	8	**12**	0.000	**12**	1.740	12	0.00
c-fat200-2	200	3235	0.163	24	**24**	0.010	14	**24**	0.000	23	1.750	24	0.00
c-fat200-5	200	8473	0.426	58	**58**	0.010	32	**58**	0.000	48	1.790	58	0.00
c-fat500-10	500	46627	0.374	126	**126**	0.020	66	**126**	0.010	96	14.900	126	0.04
c-fat500-1	500	4459	0.036	14	**14**	0.000	9	**14**	0.010	12	14.760	14	0.00
c-fat500-2	500	9139	0.073	26	**26**	0.020	15	**26**	0.010	22	14.950	26	0.00
c-fat500-5	500	23191	0.186	64	**64**	0.020	34	**64**	0.010	51	14.660	64	0.00
hamming6-2	64	1824	0.905	32	**32**	0.020	18	**32**	0.000	29	0.160	32	0.00
hamming6-4	64	704	0.349	4	**4**	0.000	3	**4**	0.000	**4**	0.210	4	0.00
hamming8-2	256	31616	0.969	128	**128**	0.020	58	**128**	0.000	97	2.270	128	0.00
johnson8-2-4	28	210	0.556	4	**4**	0.010	3	**4**	0.000	**4**	0.050	4	0.00
johnson8-4-4	70	1855	0.768	14	**14**	0.010	9	8	0.000	**14**	0.220	14	0.00
johnson16-2-4	120	5460	0.765	8	**8**	0.020	7	**8**	0.000	**8**	0.620	8	3.80
johnson32-2-4	496	107880	0.879	16	**16**	0.890	15	**16**	0.010	**16**	14.350	$\geq$16	—
MANN_a9	45	918	0.927	16	**16**	0.000	12	12	0.000	**16**	0.090	16	0.00

Table III. Continued

					$\mathcal{C}$		$\mathcal{G}$	$\mathcal{D}$		$\mathcal{MC}$		Ref. 15	
DIMACS benchmarks	n	m	d	$\omega(G)$	k	Time(s)	k	k	Time(s)	k	Time(s)	k	Time(s)
p_hat500-1	500	31569	0.253	9	**9**	0.220	7	7	0.010	**9**	18.170	9	2.30
p_hat500-2	500	62946	0.505	36	**36**	0.120	18	32	0.010	29	18.940	36	32.69
p_hat500-3	500	93800	0.752	$\geq$50	**50**	2.280	26	46	0.010	39	17.080	50	12,744.7
p_hat1000-1	1000	122253	0.245	$\geq$10	**10**	2.280	7	8	0.080	9	92.480	10	27.80
p_hat1000-2	1000	244799	0.490	$\geq$46	**46**	0.270	23	42	0.070	36	94.620	46	16,845.7
p_hat1000-3	1000	371746	0.744	$\geq$68	**68**	1.950	33	55	0.070	49	89.710	$\geq$66	—
san200_0.7_1	200	13930	0.700	30	**30**	0.010	16	15	0.000	16	2.070	30	0.36
san200_0.7_2	200	13930	0.700	18	15	0.070	13	12	0.000	13	2.060	18	0.37
san200_0.9_1	200	17910	0.900	70	62	0.020	39	45	0.000	45	1.600	70	1.04
san200_0.9_2	200	17910	0.900	60	**60**	0.020	31	35	0.000	45	1.610	60	2.62
san200_0.9_3	200	17910	0.900	44	42	0.010	26	24	0.000	30	1.600	44	182.77
san400_0.9_1	400	71820	0.900	100	96	0.020	48	50	0.000	51	7.360	100	1.70
sanr200_0.7	200	13868	0.697	18	**18**	0.080	12	16	0.000	16	2.070	18	4.30
sanr200_0.9	200	17863	0.898	42	**42**	0.150	27	36	0.000	34	1.610	42	150.08
sanr400_0.5	400	39984	0.501	13	**13**	0.910	9	8	0.000	11	10.750	13	17.12
sanr400_0.7	400	55869	0.700	21	**21**	0.150	14	16	0.000	17	9.550	21	3,139.11

Table IV. Results for random graph with density $\mathbf{d} = 0.5$

				$\mathcal{C}$		$\mathcal{G}$	$\mathcal{D}$		$\mathcal{MC}$	
Random graph	n	m	d	k	Time(s)	k	k	Time(s)	k	Time(s)
tbb128.5	128	4061	0.500	11	0.010	7	8	0.000	11	7.520
tbb256.5	256	16310	0.500	12	0.510	9	9	0.000	11	30.750
tbb512.5	512	65457	0.500	13	1.150	9	10	0.000	12	131.980
tbb1024.5	1024	262084	0.500	15	2.170	10	9	0.060	13	589.350
tbb2048.5	2048	1048289	0.500	16	9.280	11	10	0.222	14	2564.550
tbb4096.5	4096	4192863	0.500	17	73.840	12	11	1.870	15	11061.950
tbb8192.5	8192	16778527	0.500	19	311.950	13	11	8.130	16	50238.440
tbb16384.5	16384	67106538	0.500	19	170.470	14	11	33.850	17	216040.910

subgraphs), avoiding in this way many bottlenecks in the configuration space when only complete subgraphs are permitted.

5.3. Numerical Results on Random Graphs

In order to give a deeper analysis of the performance of our algorithm on random graphs, our experiments were extended to a collection of big instances built by mean of a random graph generator. In fact, even though the DIMACS collection includes some random instances, the number of nodes are no greater then 4000. For this reason, we implemented a random graph generator able to build instances with a fixed number of nodes and density limited only by the space occupancy of the graph on the physical memory existing on the computer. Our choice was to build a collection of fifteen instances with $n = 2^i$ for $i = 7, 8, \ldots, 14$ i.e., for $n \in \{128, 256, 512, 1024, 2048, 4096, 8192, 16384\}$ and density $d = \{0.5, 0.9\}$. The name of the instances considers first the prefix tbb, then the number of nodes,

Table V. Results for random graph with density $\mathbf{d} = 0.9$

				$\mathcal{C}$		$\mathcal{G}$	$\mathcal{D}$		$\mathcal{MC}$	
Random graphs	n	m	d	k	Time(s)	k	k	Time(s)	k	Time(s)
tbb128.9	128	7315	0.900	34	0.080	24	29	0.000	30	5.860
tbb256.9	256	29392	0.900	44	1.310	28	37	0.000	36	23.390
tbb512.9	512	117794	0.900	56	3.190	37	46	0.010	42	98.940
tbb1024.9	1024	471440	0.900	67	9.720	42	49	0.060	48	445.620
tbb2048.9	2048	1886256	0.900	76	35.740	49	55	0.222	54	1958.010
tbb4096.9	4096	7548970	0.900	84	41.780	56	57	1.860	59	8307.030
tbb8192.9	8192	30198965	0.900	90	182.350	61	63	8.120	66	35811.920

Table VI. Comparison between $\mathcal{C}$ and Cliquer on some random instances

	$\mathcal{C}$		Cliquer	
Random graphs	k	Time(s)	k	Time(s)
`tbb128.5`	11	0.010	11	0.000
`tbb256.5`	12	0.510	12	0.130
`tbb512.5`	13	1.150	14	10.540
`tbb1024.5`	15	2.170	15	1769.910
`tbb128.9`	34	0.080	34	61.570

the density and finally the extension `clq.b`. Again, note that the instances follows the rules provided by the DIMACS.

These instances are available for further research on the web on the home page of one of the co-author.[4]

On the smaller graphs we obtained the certified values of the clique number by using the program Cliquer. This is a branch and bound complete algorithm given in Ref. 14. As it is clear from the Table VI the computational times of Cliquer are too long to apply it to the larger instances.

In Tables IV and V T are reported the results on our instances, for $d = 0.5$ and $d = 0.9$ respectively.

ACKNOWLEDGMENTS

First of all we want to thank A.Sinclair who introduced us to the subject providing useful suggestions and references. We are in debt to G.Parisi for many fruitful discussions; in particular, he first proposed to look at the canonical ensemble, which constituted one of the main ingredients of our approach. We also benefitted from our many lunch breaks with F.Martinelli who represented a continuous rich source of salient remarks. We want also to thank R.D'Autilia and F. Zamponi for useful discussions and P.Dell'Olmo for his warm encouragement.

REFERENCES

1. R. Battiti and M. Protasi, Reactive Local Search for the Maximum Clique Problem. *Algorithmica* **29**:610-à637 (2001).
2. B. Bollobas, *Random Graph* (2nd edn.). (Cambridge University Press, 2001).
3. M. Caramia and G. Felici, *Mining relevant information on the Web: a clique based approach*, International Journal of Production Research, special issue on Data Mining, accepted 2006.

[4] http://www.disp.uniroma2.it/Users/iovanella/clique

4. P. Dankelmanna, G. S. Domkeb, W. Goddardc, P. Groblerd, J. H. Hattinghb, and H. C. Swarta, Maximum sizes of graphs with given domination parameters. *Discrete Math.* **281**:137–148 (2004).
5. M. R. Garey and D. S. Johnson, *Computer and Intractability: A guide to the theory of NP-completeness*. (Freeman, New York, 1976).
6. M. Jerrum, Large cliques elude the metropolis process. *Random Struct. Algorithms* **3**(4):347–359 (1992).
7. M. Jerrum and A. Sinclair, *The Markov chain Monte Carlo method: an approach to approximate counting and integration*, in "Approximation Algorithms for NP-hard Problems," D.S. Hochbaum ed. (PWS Publishing, Boston, 1996).
8. D. S. Johnson and M. Trick (eds.), *Cliques, coloring and satisfiability: Second DIMACS implementation challenge*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 26, American Mathematical Society, Providence, RI, 1996.
9. S. Kirkpatric, C. D. Gellatt, and M. P. Vecchi, Optimization by simulated annealing. *Science* **220**:671–680.
10. M. Mézard and G. Parisi, The cavity method at zero temperature. *J. Stat. Phys.* **111**:1 (2003).
11. M. Mézard, G. Parisi, and M. Virasoro, *Spin Glass Theory and Beyond* (World Scientific, 1987).
12. M. Mézard, G. Parisi, and R. Zecchina, Analytic and algorithmic solution of random satisfiability problems. *Science* **297**:812–815 (2002).
13. M. Mézard and R. Zecchina, The random K-satisfiability problem: from an analytic solution to an efficient algorithm. *Phys. Rev. E* **66**, 056126-1/056126-27 (2002).
14. S. Niskanen and P. R. J. Östergärd, *Cliquer User's Guide, Version 1.0*, Communication Laboratory, Helsinki University of Technology, Espoo, Finland, Tech. Rep. T48, 2003.
15. J.-C. Régin, Using constraint programming to solve the maximum clique problem. In F. Rossi (ed.), *Ninth international conference on principles and practice of constraint programming (CP'03)* vol. 2833. Lecture Notes Comput. Sci., pp. 634–648 (2003).