The Backtracking Survey Propagation Algorithm for Solving Random K-SAT Problems

Raffaele Marino\textsuperscript{1,2}, Giorgio Parisi\textsuperscript{3,4,5}, and Federico Ricci-Tersenghi\textsuperscript{3,4,5}

\textsuperscript{1}NORDITA, KTH Royal Institute of Technology and Stockholm University, Roslagstullsbacken 23, SE-10691 Stockholm, Sweden. Email: rmarino@kth.se
\textsuperscript{2}AlbaNova University Centre, KTH-Royal Institute of Technology, Dept. of Computational Biology, SE-10691 Stockholm, Sweden
\textsuperscript{3}Dipartimento di Fisica, Sapienza Universit`a di Roma, P.le Aldo Moro 5, I-00185, Roma, Italy
\textsuperscript{4}Sezione di Roma 1, Istituto Nazionale di Fisica Nucleare, P.le Aldo Moro 5, I-00185, Roma, Italy
\textsuperscript{5}Unit`a di Roma, Nanotec, Consiglio Nazionale delle Ricerche, P.le Aldo Moro 5, I-00185, Roma, Italy, Emails: giorgio.parisi@roma1.infn.it, federico.ricci@roma1.infn.it

April 21, 2016

Abstract

Satisfiability of large random Boolean formulas with \( K \) variables per clause (random \( K \)-satisfiability) is a fundamental problem in combinatorial discrete optimization. Here we study random \( K \)-satisfiability for \( K = 3, 4 \) by the Backtracking Survey Propagation (BSP) algorithm, which is able to find, in a time almost linear in the problem size, solutions within a region never reached before: for \( K = 3 \) the algorithmic threshold practically coincides with the SAT-UNSAT threshold, while for \( K = 4 \) it extrapolates beyond the rigidity threshold, where most of the solutions acquire a positive density of frozen variables. All solutions found by BSP have no frozen variables, thus supporting the conjecture that only unfrozen solutions can be found in linear time. The iterative algorithm for determining which variables are frozen in a solution (whitening) reaches the all-variables-unfrozen fixed point following a two step process and has a relaxation time diverging at the algorithmic threshold.

Optimization problems with discrete variables are widespread among scientific disciplines and often among the hardest to be solved. The \( K \)-satisfiability (\( K \)-SAT) problem is a combinatorial discrete optimization problem of \( N \) Boolean variables, \( \vec{x} = \{x_i\}_{i=1}^{N} \), submitted to \( M \) constraints. Each constraint, called clause, is in the form of an OR logical operator of \( K \) literals (variables or their negations), and the problem is solvable when there exists at least one configuration of the variables, among the \( 2^N \) possible ones, that satisfies all constraints. The \( K \)-SAT problem for \( K \geq 3 \) is a central problem in combinatorial optimization: it was among the first problems shown to be \( NP \)-complete \cite{1} \cite{2} \cite{3} and is still very much studied. Efforts from the theoretical computer science community have been especially devoted to the study of the random \( K \)-SAT ensemble \cite{4} \cite{5}, where each formula is generated by randomly choosing \( M = \alpha N \) clauses of \( K \) literals \cite{6}; indeed formulas from this ensemble become extremely hard to solve when the clause to variable ratio \( \alpha \) grows \cite{6} and still the locally tree-like structure of the factor graph \cite{7}, representing the interaction network among variables, makes the random \( K \)-SAT ensemble a perfect candidate for an analytic solution. The study of random \( K \)-SAT problems and of the related solving algorithms is likely to shed light on the origin of the computational complexity and to allow for the development of improved solving algorithms.

Both numerical \cite{8} and analytical \cite{9} \cite{10} evidences suggest that a threshold phenomenon takes place in random \( K \)-SAT ensembles: in the limit of very large formulas, \( N \rightarrow \infty \), a typical formula has a solution for \( \alpha < \alpha_s(K) \), while it is unsatisfiable for \( \alpha > \alpha_s(K) \). It has been very recently proved in Ref. \cite{11} that for \( K \) large enough the SAT-UNSAT threshold \( \alpha_s(K) \) exists in the \( N \rightarrow \infty \) limit and coincides with the prediction from the cavity method in statistical physics \cite{12}. A widely accepted conjecture is that the SAT-UNSAT threshold \( \alpha_s(K) \) exists for any value of \( K \). The main problem is that finding solutions close to \( \alpha_s \) is very hard, and all known algorithms running in polynomial time fail to find solutions for \( \alpha > \alpha_s \). The algorithmic threshold \( \alpha_a \) depends on the specific algorithm, but it is well below \( \alpha_s \) for most algorithms. There are
two main open questions: (i) finding improved algorithms having a larger $\alpha_a$, and (ii) understanding what is the theoretical upper bound to $\alpha_a$. In the present manuscript we present progresses on both issues.

The best prediction about the SAT-UNSAT threshold comes from the cavity method \cite{12,13,14,15}: for example, $\alpha_s(K=3) = 4.2667$ \cite{14} and $\alpha_s(K=4) = 9.931$ \cite{15}. Actually the statistical physics study of random $K$-SAT ensembles also provides us with a very detailed description of how the space of solutions changes when $\alpha$ spans the whole SAT phase ($0 \leq \alpha \leq \alpha_s$). Considering typical formulas in the large $N$ limit and the vast majority of solutions in these formulas (i.e. typical solutions), we known that, at low enough $\alpha$ values, the solutions are many and well connected, such as to form a single cluster.

Increasing $\alpha$ not only the number of solutions decreases, but at $\alpha_d$ the random $K$-SAT ensemble undergoes a phase transition and the space of solutions shatters in an exponentially large (in the problem size $N$) number of clusters: two solutions belonging to different clusters have a Hamming distance $O(N)$. Defining the energy function $E(x)$ as the number of unsatisfied clauses in configuration $x$, it has been found \cite{12} that for $\alpha > \alpha_d$ there are exponentially many metastable states of positive energy, which may trap algorithms that look for solutions by energy relaxation (e.g. Monte Carlo simulated annealing).

Further increasing $\alpha$, each cluster loses solutions and shrinks, but the most relevant change is in the number of clusters. The cavity method allows us to count clusters of solutions as a function of the number of solutions they contain \cite{17} and from this very detailed description several other phase transitions have been identified \cite{15} \cite{15}. For example, at $\alpha_c$ a condensation phase transition takes place, such that for $\alpha > \alpha_c$ the vast majority of solutions belong to a sub-exponential number of clusters, leading to effective long-range correlations among variables in typical solutions, which are hard to approximate by any algorithm with a finite horizon. In general $\alpha_d \leq \alpha_c \leq \alpha_s$ hold. Most of the above picture of the solutions space has been proven rigorously in the large $K$ limit \cite{19,20}.

Moving to the algorithmic side, a very interesting question is whether such a rich structure of the solutions space affects the performances of searching algorithms. While clustering at $\alpha_d$ may have some impact on Monte Carlo based algorithms and on algorithms that try to sample solutions uniformly \cite{21}, many algorithms exist that can find at least one solution with $\alpha > \alpha_d$ \cite{12,22,23}.

A very likely conjecture is that the hardness of a formula is related to the existence of a subset of highly correlated variables, which is very hard to assign correctly altogether; the worst case being a subset of variables that can have a unique assignment. This concept was introduced with the name of backbone in Ref. \cite{24}. The same concept applied to solutions within a single cluster lead to the definition of frozen variables (within a cluster) as those variables taking the same value in all solutions of the cluster \cite{25}. It has been proven in Ref. \cite{20} that the fraction of frozen variables in a cluster is either zero or lower bounded by $(\alpha e^2)^{-1/(K-2)}$; in the latter case the cluster is called frozen.

According to the above conjecture, finding a solution in a frozen cluster is hard (in practice it should require a time growing exponentially with $N$). So the smartest algorithm running in polynomial time should search for unfrozen clusters as long as they exist. Unfortunately counting unfrozen cluster is not an easy job, and indeed a large deviation analysis of their number has been achieved only very recently \cite{27} for a different and simpler problem (bicoloring random regular hypergraphs). For random $K$-SAT only partial results are known, that can be stated in terms of two thresholds: for $\alpha > \alpha_c$ (rigidity) typical solutions are in frozen cluster (but a minority of solutions may still be unfrozen), while for $\alpha > \alpha_f$ (freezing) all solutions are frozen. It has been rigorously proven \cite{25,20} that $\alpha_f < \alpha_c$ holds strictly for $K > 8$. For small $K$, which is the interesting case for benchmarking solving algorithms, we only know that $\alpha_f(K=3) = 4.254(9)$ from exhaustive enumerations in small formulas ($N \leq 100$) \cite{30} and $\alpha_f(K=4) = 9.883(15)$ from the cavity method \cite{15}. In general $\alpha_f \leq \alpha_r \leq \alpha_f \leq \alpha_s$ hold.

The conjecture above implies that no polynomial time algorithm can solve problems with $\alpha \geq \alpha_f$, but also finding solutions close to the rigidity threshold $\alpha_r$ is expected to be very hard, given that unfrozen solutions becomes a tiny minority. And this is indeed what happens for all known algorithms. Since we are interested in solving very large problems we only consider algorithms whose running time scales almost linearly with $N$ and we measure performances of each algorithm in terms of its algorithmic threshold $\alpha_a$.

Solving algorithms for random $K$-SAT problems can be roughly classified in two main categories: (a) algorithms that search for a solution by performing a biased random walk in the space of configurations and (b) algorithms that try to build the solutions by assigning variables, according to some estimated marginals. WalkSat \cite{31}, focused Metropolis search (FMS) \cite{22}.

\footnote{In SAT problems we say 2 solutions are neighbors if they differ in the assignment of just one variable; in other problems (e.g. XOR-SAT \cite{15}) this notion of neighboring needs to be relaxed, because a pair of solutions differing in just one variable are not allowed by construction. As long as the notion of neighborhood is relaxed to Hamming distances $o(N)$ all the picture of the solutions space based on statistical physics remains unaltered.}
and ASAT [23] belong to the former category; while in the latter category we find Belief Propagation guided Decimation (BPD) [21] and Survey Inspired Decimation (SID) [32]. All these algorithms are rather effective in finding solutions to random K-SAT problems: e.g. for K = 4 we have α_{BPD}^c = 9.05, α_{FMS}^c ≃ 9.55 and α_{SID}^c ≃ 9.73 to be compared with a much lower algorithmic threshold α_{UC}^c = 5.54 achieved by Generalized Unit Clause, the best algorithm whose convergence to a solution can be proven rigorously [33]. Among the efficient algorithms above, only BPD can be solved analytically [21] to find the algorithmic threshold α_{BPD}^c; for all the others we are forced to run extensive numerical simulations in order to measure α_n.

At present the algorithm achieving the best performances on several constraint satisfaction problems is SID, which has been successfully applied to the random K-SAT problem [12] and to the coloring problem [34]. The statistical properties of the SID algorithm for K = 3 have been studied in details in Refs. [33][32]. Numerical experiments on random 3-SAT problems with a large number of variables, up to N = 3 × 10^5, show that in a time that is approximately linear in N the SID algorithm finds solutions up to α_{SID}^c ≃ 4.2525 [35], that is definitely smaller, although very close to, α_s(K = 3) = 4.2667. In the region α_{SID}^c < α < α_s the problem is satisfiable for large N, but at present no algorithm can find solutions there.

To fill this gap we study a new algorithm for finding solutions to random K-SAT problems, the Backtracking Survey Propagation (BSP) algorithm. This algorithm (fully explained in the Methods section) is based, as SID, on the survey propagation (SP) equations derived within the cavity method [12,35][32] that provide an estimate on the total number of clusters N_{clus} = exp(Σ). The BSP algorithm, like SID, aims at assigning gradually the variables such as to keep the complexity Σ as large as possible, i.e. trying not to kill too many clusters [35]. While in SID each variable is assigned only once, in BSP we allow unsetting variables already assigned such as to backtrack on previous non-optimal choices. In BSP the r parameter is the ratio between the number of backtracking moves (unsetting one variable) and the number of decimation moves (assigning one variable), r < 1 must hold and for r = 0 we recover the SID algorithm. Running times scale as 1/(1 − r), but the algorithm complexity remains practically linear in N for any r < 1.

The idea supporting backtracking [36] is that a choice made at the beginning of the decimation process, when most of the variables are unassigned, may turn to be suboptimal later on, and re-assigning a variable that is no longer consistent with the current best estimate of its marginal probability may lead to a better satisfying configuration. We do not expect the backtracking to be essential when correlations between variables are short ranged, but approaching α_s we know that correlations become long ranged and thus the assignment of a single variable may affect a huge number of other variables: this is the situation when we expect the backtracking to be essential.

This idea may look similar in spirit to the survey propagation reinforcement (SPR) algorithm [37], where variables are allowed to change their most likely value during the run, but in practice BSP works much better. In SPR, once reinforcement fields are large, the re-assignment of any variable becomes unfeasible, while in BSP variables can be re-assigned to better values until the very end, and this is a major advantage.

**Results**

In this section we show the results of extensive numerical simulations using BSP. In each plot having α on the abscissa, the right end of the plot coincides with the best estimate of α_s in order to provide an immediate indication of how close to the SAT-UNSAT threshold the algorithm can work.

**Probability of finding a SAT assignment**

The standard way to study the performances of a solving algorithm is to measure the fraction of instances it can solve as a function of α. We show in Fig. 1 such a fraction for BSP run with three values of the r parameter (r = 0, 0.5 and 0.9) on random 4-SAT problems of two different sizes (N = 5000 and N = 50000). The probability of finding a solution increases both with r and N, but an extrapolation to the large N limit of these

![Figure 1: Fraction of random 4-SAT instances solved by BSP. Such a fraction is computed over 100 instances at each α value for N = 5 000 (solid symbols) and N = 50 000 (empty symbols). On instances not solved, a second run rarely (< 1%) finds a solution.](image)
data is unlikely to provide a reliable estimation of the algorithmic threshold $\alpha_a$.

Order parameter and algorithmic threshold

In order to obtain a reliable estimate of $\alpha_a$, we look for an order parameter vanishing at $\alpha_a$ and having very little finite size effects. We identify this order parameter with the quantity $\Sigma_{\text{res}}/N_{\text{res}}$, where $\Sigma_{\text{res}}$ and $N_{\text{res}}$ are respectively the complexity (i.e. log of number of clusters) and the number of unassigned variables in the residual formula. As explained in Methods, BSP assigns and re-assigns variables, thus modifying the formula, until the formula simplifies enough that the SP fixed point has only null messages: the residual formula is defined as the last formula with non-null SP fixed point messages. We have experimentally observed that the BSP algorithm (as the SID one [35]) can simplify the formula enough to reach the trivial SP fixed point only if the complexity $\Sigma$ remains strictly positive during the whole decimation process. In other words, on every run where $\Sigma$ becomes very close to zero or negative, SP stops converging or a contradiction is found. This may happen either because the original problem was unsatisfiable or because the algorithm made some wrong assignments incompatible with the few available solutions. Thanks to the above observation we have that $\Sigma_{\text{res}} \geq 0$ and thus a null value for the mean residual complexity signals that the BSP algorithm is not able to find any solution, and thus provides a valid estimate for the algorithmic threshold $\alpha_{a}^{\text{BSP}}$.

As we see in the data shown in the upper panel of Fig. 2 the mean value of the intensive mean residual complexity $\Sigma_{\text{res}}/N_{\text{res}}$ is practically size-independent and a linear fit provides a very good data interpolation: tiny finite size effects are visible in the largest $N$ datasets only close to the dataset right end. The linear extrapolation predicts $\alpha_{a}^{\text{BSP}} \approx 9.9$ (for $K = 4$ and $r = 0.9$), which is slightly above the rigidity threshold $\alpha_{r} = 9.883(15)$ computed in Ref. [15]. This means that BSP is able to find solutions in a region of $\alpha$ where the majority of solutions is in frozen clusters and thus hard

Figure 2: BSP algorithmic threshold on random 4-SAT problems. The upper panel shows $\Sigma_{\text{res}}/N_{\text{res}}$ as a function of $\alpha$, for random 4-SAT problems solved by BSP with $r = 0.9$. This order parameter is mostly size-independent (few deviations appear only at the right end) and vanishes at $\alpha_{a}^{\text{BSP}} \approx 9.9$, slightly beyond the rigidity threshold $\alpha_{r} = 9.883(15)$, marked by a vertical line. The lower panel shows that the same linear extrapolation holds for other values of $r$, the black line being the fit to $r = 0.9$ data shown in the upper panel, but SID without backtracking ($r = 0$) has a much lower algorithmic threshold, $\alpha_{a}^{\text{SID}} \approx 9.83$.

Figure 3: BSP algorithmic threshold on random 3-SAT problems. Same as Fig. 2 for $K = 3$. BSP does not show any change at the freezing threshold $\alpha_{f} = 4.254(9)$ measured on small problems [30], and marked by the vertical line. A linear fit to the residual complexity (red line) extrapolates to zero slightly beyond the SAT-UNSAT threshold, at $\alpha_{a}^{\text{BSP}} \approx 4.268$, strongly suggesting BSP can find solutions in the entire SAT phase for $K = 3$ in the large $N$ limit. The black line is a linear fit vanishing at $\alpha_{a}$.
for random 3-SAT the algorithmic threshold of BSP, run with $r = 0.9$, practically coincide with the SAT-UNSAT threshold $\alpha_s$ (see Fig. 3), thus providing a strong evidence that BSP can find solutions in the entire SAT phase. The estimate for the freezing threshold $\alpha_f = 4.254(9)$ obtained in Ref. [30] with $N \leq 100$ is likely to be too small, given that all solutions found by BSP for $N = 10^6$ are unfrozen, even for $\alpha > \alpha_f$.

Computational complexity As explained in Methods, the BSP algorithm performs $O(f^{-1}(1-r)^{-1})$ elementary steps, where at each step $fN$ variables are either assigned [with prob. $1/(1+r)$] or released [with prob. $r/(1+r)$]. At the beginning of each step, the algorithm solves the SP equations with a mean number $\eta$ of iterations and then sort local marginals. The total number of elementary operations to solve an instance grows as $f^{-1}(1-r)^{-1}(\eta N + N \log N)$, i.e. the algorithm running time is at most $O(N \log N)$. Fig. 4 shows that $\eta$ is actually a small number changing mildly with $\alpha$ and $N$ both for $K = 3$ and $K = 4$. The main change

![Figure 4: BSP convergence time. The mean number $\eta$ of iterations to reach a fixed point of SP equations grows very mildly with $\alpha$ and $N$, both for $K = 3$ (upper panel) and $K = 4$ (lower panel).](image)

to find. We show below that BSP actually finds solutions in atypical unfrozen clusters, as it has been observed already in very smart algorithms solving other kind of constraint satisfaction problems [38 39].

The effectiveness of the backtracking can be appreciated in the lower panel of Fig. 2 where the order parameter $\Sigma_{\text{res}}/N_{\text{res}}$ is shown for $r = 0$ and $r = 0.5$, together with linear fits to these datasets and to the $r = 0.9$ dataset (black line). We observe that the algorithmic threshold for BSP is much larger (on the scale measuring the relative distance from the SAT-UNSAT threshold) that the one for SID (i.e. $r = 0$ dataset).

For random 3-SAT the algorithmic threshold of BSP, run with $r = 0.9$, practically coincide with the SAT-UNSAT threshold $\alpha_s$ (see Fig. 3), thus providing a strong evidence that BSP can find solutions in the entire SAT phase. The estimate for the freezing threshold $\alpha_f = 4.254(9)$ obtained in Ref. [30] with $N \leq 100$ is likely to be too small, given that all solutions found by BSP for $N = 10^6$ are unfrozen, even for $\alpha > \alpha_f$.

Computational complexity As explained in Methods, the BSP algorithm performs $O(f^{-1}(1-r)^{-1})$ elementary steps, where at each step $fN$ variables are either assigned [with prob. $1/(1+r)$] or released [with prob. $r/(1+r)$]. At the beginning of each step, the algorithm solves the SP equations with a mean number $\eta$ of iterations and then sort local marginals. The total number of elementary operations to solve an instance grows as $f^{-1}(1-r)^{-1}(\eta N + N \log N)$, i.e. the algorithm running time is at most $O(N \log N)$. Fig. 4 shows that $\eta$ is actually a small number changing mildly with $\alpha$ and $N$ both for $K = 3$ and $K = 4$. The main change
Figure 5: Whitening random 3-SAT solutions.
The upper panel shows the mean fraction of non-\# variables during the whitening procedure applied to all solutions found by the BSP algorithm. The line is the SP prediction for the fraction of frozen variables in typical solutions at \( \alpha = 4.262 \), and shows that solutions found by BSP are atypical. The central panel shows the histograms of the whitening times, defined as the number of iterations required to reach a fraction 0.4 of non-\# variables. Increasing \( \alpha \) both the mean value and the variance of the whitening times grow. The lower panel shows the same data used for the upper one, but averaged at fixed \( \tau(0) - t \), the time to reach the all-\# fixed point. Errors are tiny since the whitening dynamics below the plateau is the same for each solution.

Fraction of non-\# variables as a function of the time to reach the all-\# configuration, \( \tau(0) - t \), we see that fluctuations are strongly suppressed and the relaxation is the same for each solution (see lower panel in Fig. 5).

Critical exponent for the whitening time divergence. The whitening time \( \tau(c) \), defined as the mean time needed to reach a fraction \( c \) of non-\# variables in the whitening procedure, is plotted in a double logarithmic scale as a function of \( \Sigma_{res} \) for random 3-SAT problems with \( N = 10^6 \) (upper dataset) and random 4-SAT problems with \( N = 5 \times 10^4 \) (lower dataset). The lines are power law fits with exponent \( \nu = 0.281(6) \) for \( K = 3 \) and \( \nu = 0.269(5) \) for \( K = 4 \).

Nonetheless a word of caution is needed since the solutions we are using as starting points for the whitening procedure are atypical solutions (otherwise they would likely contain frozen variables and would not flow to the all-\# configuration under the whitening procedure). So, while finding universal critical properties in a dynamical process is definitely a good news, how to relate...
it to the behavior of the same process on typical solutions it is not obvious (and indeed for the whitening process starting from typical solutions one would expect the naive mean field exponent $\nu = 1/2$, which is much larger than the one we are finding).

**Discussion**

We have studied the Backtracking Survey Propagation (BSP) algorithm for finding solutions in very large random $K$-SAT problems and provided numerical evidence that it works much better than any previously available algorithm. That is, BSP has the largest algorithmic threshold known at present. The main reason for its superiority is the fact that variables can be re-assigned at any time during the run, even at the very end. In other solving algorithms that may look similar, as e.g. survey propagation reinforcement [37], re-assignment of variables actually takes place mostly at the beginning of the run, and this is far less efficient in hard problems. Even doing a lot of helpful backtracking, the BSP running time is still $O(N \log N)$ in the worst case, and thanks to this it can be used on very large problems with millions of clauses.

For $K = 3$ the BSP algorithm finds solutions practically up to the SAT-UNSAT threshold $\alpha_s$, while for $K = 4$ a tiny gap to the SAT-UNSAT threshold still remains, but the algorithmic threshold $\alpha^\text{BSP}_a$ seems to be located beyond the rigidity threshold $\alpha_r$ in the large $N$ limit. Beating the rigidity threshold, i.e. finding solutions in a region where the majority of solutions belongs to clusters with frozen variables, is hard, but not impossible. Even under the assumption that finding frozen solutions takes an exponential time in $N$, very smart polynomial time algorithms can look for a solution in the sub-dominant unfrozen clusters [38, 39]. BSP belongs to this category, as we have shown that all solutions found by BSP have no frozen variables.

One of the main questions we tried to answer with our extensive numerical simulations is whether BSP is reaching (or approaching closely) the ultimate threshold $\alpha_a$ for polynomial time algorithms solving large random $K$-SAT problems. Under the assumption that frozen solutions can not be found in polynomial time, such an algorithmic threshold $\alpha_a$ would coincide with the freezing transition at $\alpha_f$ (i.e. when the last unfrozen solution disappears). Unfortunately for random $K$-SAT the location of $\alpha_f$ is not known with enough precision to allow us to reach a definite answer to this question. It would be very interesting to run BSP on random hypergraph bicoloring problems, where the threshold values are known [43] and a very recent work has shown that the large deviation function for the number of unfrozen clusters can be computed [27].

It is worth noticing that the BSP algorithm is easy to parallelize, since most of the operations are local and do not require any strong centralized control. Obviously the effectiveness of a parallel version of the algorithm would largely depend on the topology of the factor graph representing the specific problem: if the factor graph is an expander, then splitting the problem on several cores may require too much inter-core bandwidth, but in problems having a natural hierarchical structure the parallelization may lead to further performance improvements.

The backtracking introduced in the BSP algorithm helps a lot in correcting errors made during the partial assignment of variables and this allow the BSP algorithm to reach solutions at large $\alpha$ values. Clearly the price we pay is that a too frequent backtracking makes the algorithm slower. A natural direction to improve this class of algorithms would be to used biased marginals focusing on solutions which are easier to be reached by the algorithm. For example in the region $\alpha > \alpha_r$, the measure is concentrated on solutions with frozen variables, but these can not be really reached by the algorithm. The backtracking thus intervenes and corrects the partial assignment until a solution with unfrozen variables is found by chance. If the marginals could be computed from a new biased measure which is concentrated on the unfrozen clusters, this could make the algorithm go immediately in the right direction and much less backtracking would be hopefully needed.

**Methods**

The Backtracking Survey Propagation (BSP) algorithm is an extension of the Survey Inspired Decimation (SID) algorithm for finding solutions to random $K$-satisfiability problems. A detailed description of the SID algorithm can be found in Refs. [12, 13]. Here, we briefly recall SID and describe the novelties leading to the BSP algorithm [36].

**Survey Inspired Decimation (SID)** The SID algorithm is based on the survey propagation (SP) equations derived by the cavity method [12, 13], that can be written in a compact way as

$$m_{a \rightarrow i} = \prod_{j \in \partial_a \backslash i} m_{j \rightarrow a},$$

$$m_{i \rightarrow a} = \frac{\pi_{ia}^- (1 - \pi_{ia}^+)}{1 - \pi_{ia}^+ \pi_{ia}^-},$$

with $\pi_{ia}^\pm = 1 - \prod_{b \in \partial_a \backslash i} (1 - \tilde{m}_{b \rightarrow i}),$
where $\partial_a$ is the set of variables in clause $a$, and $\partial_{a^\dagger}$ (resp. $\partial^{-1}_{a}$) is the set of clauses containing $x_i$, excluding $a$ itself, satisfied (resp. not satisfied) when the variable $x_i$ is assigned to satisfy clause $a$.

Physical interpretation of the SP equations: $\hat{m}_{a \rightarrow i}$ represents the fraction of clusters where clause $a$ is satisfied solely by variable $x_i$ (that is, $x_i$ is frozen by clause $a$), while $m_{i 
rightarrow a}$ is the fraction of clusters where $x_i$ is frozen to an assignment not satisfying clause $a$.

The SP equations impose $2KM$ self-consistency conditions on the $2KM$ variables $\{m_{i 
rightarrow a}, \hat{m}_{a \rightarrow i}\}$ living on the edges of the factor graph [7], that are solved in an iterative way, leading to a message passing algorithm (MPA) [3], where outgoing messages from a factor graph node (variable or clause) are functions of the incoming messages. Once the MPA reaches a fixed point $\{m^*_{i 
rightarrow a}, \hat{m}^*_{a \rightarrow i}\}$ that solves the SP equations, the number of clusters can be estimated via the complexity divergence

$$\Sigma = \log N_{\text{clus}} = \sum_i \Sigma_i + \sum_a (1 - K_a) \Sigma_a,$$

$$\Sigma_i = \log(1 - \pi_i^+ \pi_i^-), \quad \pi_i^+ = 1 - \prod_{b \in \partial_i^+} (1 - \hat{m}_{b \rightarrow i}),$$

$$\Sigma_a = \log \left(1 - \prod_{j \in \partial a} m^*_{j 
rightarrow a}\right),$$

where $K_a$ is the length of clause $a$ (initially $K_a = K$) and $\partial_i^+$ (resp. $\partial_i^-$) is the set of clauses satisfied by setting $x_i = 1$ (resp. $x_i = -1$). The SP fixed point messages also provide information about the fraction of clusters where variable $x_i$ is forced to be positive ($w_i^+$), negative ($w_i^-$) or not forced at all ($1 - w_i^+ - w_i^-$)

$$w_i^+ = \frac{\pi_i^+ (1 - \pi_i^-)}{1 - \pi_i^+ \pi_i^-}.$$

The SID algorithm then proceed by assigning variables (decimation step). According to SP equations, assigning a variable $x_i$ to its most probable value (i.e., setting $x_i = 1$ if $w_i^+ > w_i^-$ and viceversa), the number of clusters gets multiplied by a factor, called *bias*

$$b_i = 1 - \min(w_i^+, w_i^-).$$

With the aim of decreasing the lesser the number of cluster and thus keeping the largest the number of solutions in each decimation step, SID assigns/decimate variables with the largest $b_i$ values. In order to keep the algorithm efficient, at each step of decimation a small fraction $f$ of variables is assigned, such that in $O(\log N)$ steps of decimation a solution can be found.

After each step of decimation, the SP equations are solved again on the subproblem, which is obtained by removing satisfied clauses and by reducing clauses containing a false literal (unless a zero-length clause is generated, and in that case the algorithm returns a failure). The complexity and the biases are updated according to the new fixed point messages, and a new decimation step is performed.

The main idea of the SID algorithm is that fixing variables which are almost certain to their most probable value, one can reduce the size of the problem without reducing too much the number of solutions. The evolution of the complexity $\Sigma$ during the SID algorithm can be very informative [35]. Indeed it is found that, if $\Sigma$ becomes too small or negative, the SID algorithm is likely to fail, either because the iterative method for solving the SP equations no longer converges to a fixed point or because a contradiction is generated by assigning variables. In these cases the SID algorithm returns a failure. On the contrary, if $\Sigma$ always remains well positive, the SID algorithm reduces so much the problem, that eventually a trivial SP fixed point, $m_{i 
rightarrow a} = \hat{m}_{a \rightarrow i} = 0$, is reached. This is a strong hint that the remaining subproblem is easy and the SID algorithm tries to solve it by WalkSat [31].

A careful analysis of the SID algorithm for random 3-SAT problems of size $N = O(10^5)$ shows that the algorithmic threshold achievable by SID is $\alpha^*_\text{SID} = 4.2525$ [35], which is close, but definitely smaller than the SAT-UNSAT threshold $\alpha_s = 4.2667$.

The running time of the SID algorithm experimentally measured is $O(N \log(N))$. However it is not excluded than an extra factor $\log(N)$ could be present in the convergence time of the iterative solution to SP equations, but for sizes up to $N = O(10^7)$ has not been observed [32].

**Backtracking Survey Propagation (BSP)** Willing to improve the SID algorithm to find solutions also in the region $\alpha^*_\text{SID} < \alpha < \alpha_s$, one has to reconsider the way variables are assigned. It is clear that the fact the SID algorithm assigns each variable only once is a strong limitation, especially in a situation where correlations between variables becomes extremely strong and long-ranged. In difficult problems it can easily happen that one realizes that a variable is taking the wrong value only after having assigned some of its neighbours variables. However the SID algorithm is not able to solve this kind of frustrating situations.

The BSP algorithms tries to solve this kind of problematic situations by introducing a new backtracking step, where a variable already assigned can be released and eventually re-assigned in a future decimation step. It is not difficult to understand when it is worth releasing a variable. The bias $b_i$ in terms of the SP fixed point messages $\{m^*_{a \rightarrow i}\}_{a \in \partial_i}$ arriving in $i$ can be computed also for a variable $x_i$ already assigned: if the bias $b_i$, that was large at the time the variable $x_i$ was assigned, gets strongly reduces by the effect of assign-
ing other variables, then it is likely that releasing the variable $x_i$ may be beneficial in the search for a solution. So both variables to be fixed in the decimation step and variables to be released in the backtracking step are chosen according to their biases $b_i$: variables to be fixed have the largest biases and variables to be released have the smallest biases.

The BSP algorithm then proceeds similarly to SID, by alternating the iterative solution to the SP equations and a step of decimation or backtracking on a fraction $f$ of variables in order to keep the algorithm efficient (in all our numerical experiments we have used $f = 10^{-3}$). The choice between a decimation or a backtracking step is taken according to a stochastic rule (unless there are no variables to unset), where the parameter $r \in [0, 1)$ represents the ratio between backtracking steps to decimation steps. Obviously for $r = 0$ we recover the SID algorithm, since no backtracking step is ever done. Increasing $r$ the algorithm becomes slower by a factor $1/(1-r)$, because most variables are reassigned $O(1/(1-r))$ times before the BSP algorithm finishes, but its time complexity remains $O(N \log N)$ in the problem size.

The BSP algorithm can stop for the same reasons the SID algorithm does: either the SP equations can not be solved iteratively or the generated subproblem has a contradiction. Both cases happen when the complexity $\Sigma$ becomes too small or negative. On the contrary if the complexity remain always positive the BSP eventually generate a subproblem where all SP messages are null and on this subproblem WalkSat is called.

The code used for collecting data shown in this study is available from the authors upon request.

References

[1] S. A. Cook, The complexity of theorem proving procedures, Proc. 3rd Ann. ACM Symp. on Theory of Computing, Assoc. Comput. Mach., New York, pp. 151-158 (1971).

[2] M. Garey and D. S. Johnson, Computers and Intractability: A guide to the theory of NP-completeness, Freeman, San Francisco (1979).

[3] C. H. Papadimitriou, Computational Complexity, Addison-Wesley (1994).

[4] M. M´ ezard and A. Montanari, Information, Physics, and Computation, Oxford University Press (2009).

[5] C. Moore and S. Mertens, The Nature of Computation, Oxford University Press (2011).

[6] S. A. Cook and D. G. Mitchell, Finding Hard Instances of the Satisfiability Problem: A Survey, In: Satisfiability Problem: Theory and Applications. J. Du, D. Gu and P. Pardalos (Eds). DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Volume 35 (1997).

[7] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, Factor Graphs and the Sum-Product Algorithm, IEEE Trans. Infor. Theory 47, 498-519 (2001).

[8] S. Kirkpatrick and B. Selman, Critical Behaviour in the satisfiability of random Boolean expressions, Science 264, 1297-1301 (1994).

[9] O. Dubois, Y. Boufkhad, J. Mandler, Typical random 3-SAT formulae and the satisfiability threshold, in Proc. 11th ACM-SIAM Symp. on Discrete Algorithms, 124-126 (San Francisco, CA, 2000).

[10] O. Dubois, R. Monasson, B. Selman and R. Zecchina (Eds.), Phase Transitions in Combinatorial Problems, Theoret. Comp. Sci. 265 (2001).

[11] J. Ding, A. Sly, and N. Sun, Proof of the satisfiability conjecture for large $k$, http://arxiv.org/abs/1411.0650 (2014).

[12] M. M´ ezard, G. Parisi and R. Zecchina, Analytic and algorithmic solution of random satisfiability problems, Science 297, 812-815 (2002).

[13] M. M´ ezard and R. Zecchina, The random K-satisfiability problem: from an analytic solution to an efficient algorithm, Phys. Rev. E 66, 056126 (2002).

[14] S. Mertens, M. M´ ezard and R. Zecchina, Threshold Values of Random K-SAT from the Cavity Method, Random Struct. Alg. 28, 340-373 (2006).

[15] A. Montanari, F. Ricci-Tersenghi and G. Semerjian, Clusters of solutions and replica symmetry breaking in random $k$-satisfiability, J. Stat. Mech. P04004 (2008).

[16] M. M´ ezard, F. Ricci-Tersenghi, and R. Zecchina, Two solutions to diluted $p$-spin models and XOR-SAT problems, J. Stat. Phys. 111, 505-533 (2003).

[17] F. Krzakala, A. Montanari, F. Ricci-Tersenghi, G. Semerjian, and L. Zdeborova, Gibbs States and the Set of Solutions of Random Constraint Satisfaction Problems, PNAS 104, 10318-10323 (2007).

[18] L. Zdeborova and F. Krzakala, Phase transitions in the coloring of random graphs, Phys. Rev. E 76, 031131 (2007).
[19] D. Achlioptas and A. Coja-Oghlan, *Algorithmic Barriers from Phase Transitions*, in FOCS ’08: Proceedings of the 49th annual IEEE symposium on Foundations of Computer Science, 793-802 (2008).

[20] D. Achlioptas, A. Coja-Oghlan, and F. Ricci-Tersenghi, *On the Solution-Space Geometry of Random Constraint Satisfaction Problems*, Random Struct. Alg. 38, 251-268 (2011).

[21] F. Ricci-Tersenghi and G. Semerjian, *On the cavity method for decimated random constraint satisfaction problems and the analysis of belief propagation guided decimation algorithms*, J. Stat. Mech. P09001 (2009).

[22] S. Sakari, M. Alava, and P. Orponen, *Focused local search for random 3-satisfiability*, J. Stat. Mech. P06006 (2005).

[23] J. Ardelius and E. Aurell, *Behavior of heuristics on large and hard satisfiability problems*, Phys. Rev. E 74, 037702 (2006).

[24] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman and L. Troyansky, *Determining computational complexity from characteristic phase transitions*, Nature 400, 133-137 (1999).

[25] G. Semerjian, *On the freezing of variables in random constraint satisfaction problem*, J. Stat. Phys. 130, 251-293 (2008).

[26] E. Maneva, E. Mossel, and M. J. Wainwright, *A new look at survey propagation and its generalizations*, Journal of the ACM (JACM) 54, 17 (2007).

[27] A. Braunstein, L. Dall’Asta, G. Semerjian, and L. Zdeborova, *The large deviations of the whitening process in random constraint satisfaction problems*, http://arxiv.org/abs/1602.01700 (2016).

[28] D. Achlioptas and F. Ricci-Tersenghi, *On the Solution-Space Geometry of Random Constraint Satisfaction Problems*, STOC ’06: Proceedings of the 38th annual ACM symposium on Theory of Computing (Seattle, WA, USA), 130-139 (2006).

[29] D. Achlioptas and F. Ricci-Tersenghi, *Random Formulas Have Frozen Variables*, SIAM J. Comput. 39, 260-280 (2009).

[30] J. Ardelius and L. Zdeborova, *Exhaustive enumeration unveils clustering and freezing in the random 3-satisfiability problem*, Phys. Rev. E 78, 040101 (2008).

[31] B. Selman, H. A. Kautz, and B. Cohen, *Noise strategies for improving local search*, in Proc. AAAI-94, 337-343 (1994).

[32] A. Braunstein, M. Mézard, and R. Zecchina, *Survey propagation: An algorithm for satisfiability*, Random Struct. Alg. 27, 201-226 (2005).

[33] A. Frieze and S. Suen, *Analysis of two simple heuristics on a random instance of k-SAT*, J. Algor. 20, 312-355 (1996).

[34] R. Mulet, A. Pagnani, M. Weigt, and R. Zecchina, *Coloring Random Graphs*, Phys. Rev. Lett. 89, 268701 (2002).

[35] G. Parisi, *Some remarks on the survey decimation algorithm for K-satisfiability*, http://arxiv.org/abs/cs/0301015 (2003).

[36] G. Parisi, *A backtracking survey propagation algorithm for K-satisfiability*, http://arxiv.org/abs/cond-mat/0308510 (2003).

[37] J. Chavas, C. Furtlehner, M. Mézard, and R. Zecchina, *Survey-propagation decimation through distributed local computations*, J. Stat. Mech. P11016 (2005).

[38] L. Dall’Asta, A. Ramezanpour, and R. Zecchina, *Entropy landscape and non-Gibbs solutions in constraint satisfaction problems*, Phys. Rev. E 77, 031118 (2008).

[39] L. Zdeborová and M. Mézard, *Constraint satisfaction problems with isolated solutions are hard*, J. Stat. Mech. P12004 (2008).

[40] J. M. Chambers, *Algorithm 410: Partial Sorting*, Commun. ACM 14, 357-358 (1971).

[41] G. Parisi, *On local equilibrium equations for clustering states*, http://arxiv:cs/0212047 (2002).

[42] A. Braunstein and R. Zecchina, *Survey Propagation as local equilibrium equations*, J. Stat. Mech. P06007 (2004).

[43] T. Castellani, V. Napolano, F. Ricci-Tersenghi, and R. Zecchina, *Bicolouring random hypergraphs*, J. Phys. A 36, 11037 (2003).

Acknowledgements

We thank K. Freese, R. Eichhorn and E. Aurell for useful discussions. This research is supported by the Swedish Science Council through grant 621-2012-2982.