nature COMMUNICATIONS

ARTICLE

Received 14 Oct 2015 | Accepted 23 Aug 2016 | Published 3 Oct 2016

DOI: 10.1038/ncomms12996 OPEN

The backtracking survey propagation algorithm for solving random K-SAT problems

Raffaele Marino[1], Giorgio Parisi[2] & Federico Ricci-Tersenghi[2]

Discrete combinatorial optimization has a central role in many scientific disciplines, however, for hard problems we lack linear time algorithms that would allow us to solve very large instances. Moreover, it is still unclear what are the key features that make a discrete combinatorial optimization problem hard to solve. Here we study random *K*-satisfiability problems with $K = 3,4$, which are known to be very hard close to the SAT-UNSAT threshold, where problems stop having solutions. We show that the backtracking survey propagation algorithm, in a time practically linear in the problem size, is able to find solutions very close to the threshold, in a region unreachable by any other algorithm. All solutions found have no frozen variables, thus supporting the conjecture that only unfrozen solutions can be found in linear time, and that a problem becomes impossible to solve in linear time when all solutions contain frozen variables.

[1] NORDITA and AlbaNova University Centre, Department of Computational Biology, KTH-Royal Institute of Technology and Stockholm University, Roslagstullsbacken 23, SE-10691 Stockholm, Sweden. [2] Dipartimento di Fisica, Sapienza Università di Roma and Istituto Nazionale di Fisica Nucleare, Sezione di Roma1 and CNR-Nanotec, Unità di Roma, P.le Aldo Moro 5, I-00185 Roma, Italy. Correspondence and requests for materials should be addressed to G.P. (email: giorgio.parisi@roma1.infn.it).

Optimization problems with discrete variables are widespread among scientific disciplines and often among the hardest to solve. The K-satisfiability (K-SAT) problem is a combinatorial discrete optimization problem of N Boolean variables, $\underline{x} = \{x_i\}_{i=1,N}$, submitted to M constraints. Each constraint, called clause, is in the form of an OR logical operator of K literals (variables and their negations): 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[1–3] and is still very much studied. A growing collaboration between theoretical computer scientists and statistical physicists has focused on the random K-SAT ensemble[4,5], where each formula is generated by randomly choosing $M = \alpha N$ clauses of K literals. Formulas from this ensemble become extremely hard to solve when the clause to variable ratio α grows[6]: nevertheless, even in this region, the locally tree-like structure of the factor graph[7], representing the interaction network among variables, makes the random K-SAT ensemble a perfect candidate for analytic computations. 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[8] and analytical[9,10] evidence suggest that a threshold phenomenon takes place in random K-SAT ensembles: in the limit of very large formulas, $N \to \infty$, 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. 11 that for K large enough the satisfiability to unsatisfiability (SAT-UNSAT) threshold $\alpha_s(K)$ exists in the $N \to \infty$ limit and coincides with the prediction from the cavity method of statistical physics[12]. A widely accepted conjecture is that the SAT-UNSAT threshold $\alpha_s(K)$ exists for any value of K. Finding solutions close to α_s is very hard, and all known algorithms running in polynomial time fail to find solutions when $\alpha > \alpha_a$, for some $\alpha_a < \alpha_s$. Actually, each algorithm ALG has it own algorithmic threshold α_a^{ALG}, such that the probability of finding a solution vanishes for $\alpha > \alpha_a^{\mathrm{ALG}}$ in the large N limit. For most algorithms α_a^{ALG} is well below α_s. We define $\alpha_a = max_{\mathrm{ALG}} \alpha_a^{\mathrm{ALG}}$ the threshold beyond which no polynomial-time algorithm can find solutions. There are two main open questions: to find improved algorithms having a larger α_a^{ALG}, and to understand what is the theoretical upper bound α_a. Here we present progress on both issues.

The best prediction about the SAT-UNSAT threshold comes from the cavity method[12–15]: for example, $\alpha_s(K=3) = 4.2667$ (ref. 14) and $\alpha_s(K=4) = 9.931$ (ref. 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 α spans the whole SAT phase ($0 \leq \alpha \leq \alpha_s$). Let us consider typical formulas in the large N limit and the vast majority of solutions in these formulas (that is, typical solutions), we know that, at low enough α values, the set of solutions is connected, so that they form a single cluster. In SAT problems we say two solutions are neighbours if they differ in the assignment of just one variable; in other problems (for example, in the XORSAT model[16]) this definition of neighbour needs to be relaxed, because a pair of solutions differing in just one variable are not allowed by the model definition. As long as the notion of neighbourhood is relaxed to Hamming distances $o(N)$ all the picture of the solution space based on statistical physics remains unaltered.

As α increases, not only the number of solutions decreases, but at α_d the random K-SAT ensemble undergoes a phase transition: the space of solutions shatters into an exponentially large (in the problem size N) number of clusters; two solutions belonging to different clusters have a Hamming distance $O(N)$. If we define the energy function $E(\underline{x})$ as the number of unsatisfied clauses in configuration $\underline{x}$, it has been found[12] that for $\alpha > \alpha_d$ the energy $E(\underline{x})$ has exponentially many (in N) local minima of positive energy, which may trap algorithms that look for solutions by energy relaxation (for example, Monte Carlo simulated annealing).

Further increasing α, 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[17]: using this very detailed description several other phase transitions have been identified[15,18]. For example, there is a value α_c where 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 $\alpha_d \leq \alpha_c \leq \alpha_s$ holds. Most of the above picture of the solution space has been proven rigorously in the large K limit[19,20].

Moving to the algorithmic side, a very interesting question is whether such a rich structure of the solution space affects the performance of searching algorithms. While clustering at α_d may have some impact on algorithms that sample solutions uniformly[21], many algorithms exist that can find at least one solution with $\alpha > \alpha_d$ (refs 12,22,23).

A solid conjecture is that the hardness of a formula is related to the existence of a subset of highly correlated variables, which are 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. 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[25]. It has been proven in ref. 26 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 clusters is not an easy job, and indeed a large deviation analysis of their number has been achieved only very recently[27] for a different and simpler problem (bicolouring random regular hypergraphs). For random K-SAT only partial results are known, that can be stated in terms of two thresholds: for $\alpha > \alpha_r$ (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[28,29] that $\alpha_f < \alpha_s$ holds strictly for $K > 8$. For small K, which is the interesting case for benchmarking solving algorithms, we know $\alpha_r = 9.883(15)$ for $K = 4$ from the cavity method[15], while for $K = 3$ the estimate $\alpha_f = 4.254(9)$ comes from exhaustive enumerations in small formulas ($N \leq 100$; ref. 30) and is likely to be affected by strong finite size effects. In general $\alpha_d \leq \alpha_r \leq \alpha_f \leq \alpha_s$ holds.

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 α_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 performance of each algorithm in terms of its algorithmic threshold α_a^{ALG}.

Solving algorithms for random K-SAT problems can be roughly classified in two main categories: algorithms that search for a solution by performing a biased random walk in the space of configurations and algorithms that try to build the solutions by

assigning variables, according to some estimated marginals. WalkSat[31], focused Metropolis search[22] 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: for example, for $K=4$ we have $\alpha_a^{BPD}=9.05$, $\alpha_a^{FMS}\simeq 9.55$ and $\alpha_a^{SID}\simeq 9.73$ to be compared with a much lower algorithmic threshold $\alpha_a^{GUC}=5.54$ achieved by Generalized Unit Clause, the best algorithm whose range of 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_a^{BPD}; for the others we are forced to run extensive numerical simulations to measure α_a^{ALG}.

At present the algorithm achieving the best performance on several constraint satisfaction problems is SID, which has been successfully applied to the random K-SAT problem[12] and to the colouring problem[34]. The statistical properties of the SID algorithm for $K=3$ have been studied in details in refs 32,35. Numerical experiments on random 3-SAT problems with a large number of variables, up to $N=3\times 10^5$, show that in a time that is approximately linear in N the SID algorithm finds solutions up to $\alpha_a^{SID}\simeq 4.2525$ (ref. 35), that is definitely smaller, although very close to, $\alpha_s(K=3)=4.2667$. In the region $\alpha_a^{SID}<\alpha<\alpha_s$ 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[12,32,35] that provide an estimate on the total number of clusters $N_{clus}=exp(\Sigma)$. The BSP algorithm, like SID, aims at assigning gradually the variables such as to keep the complexity Σ as large as possible, that is, 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. The running time scales as $N/(1-r)$, with a slight overhead for maintaining the data structures, making the running time effectively 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; if we re-assign a variable that is no longer consistent with the current best estimate of its marginal probability, we may get 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 crucial.

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

Probability of finding a SAT assignment. The standard way to study the performance 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 ($N=5{,}000$ and $N=50{,}000$). The probability of finding a solution increases both with r and N, but an extrapolation to the large N limit of these data is unlikely to provide a reliable estimation of the algorithmic threshold α_a^{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.

Order parameter and algorithmic threshold. In order to obtain a reliable estimate of α_a^{BSP} we look for an order parameter vanishing at α_a^{BSP} and having very little finite size effects. We identify this order parameter with the quantity Σ_{res}/N_{res}, where Σ_{res} and N_{res} are respectively the complexity (for example, 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 Σ remains strictly positive during the whole decimation process. In other words, on every run where Σ 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 $\Sigma_{res}\geq 0$ 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 α_a^{BSP}. From

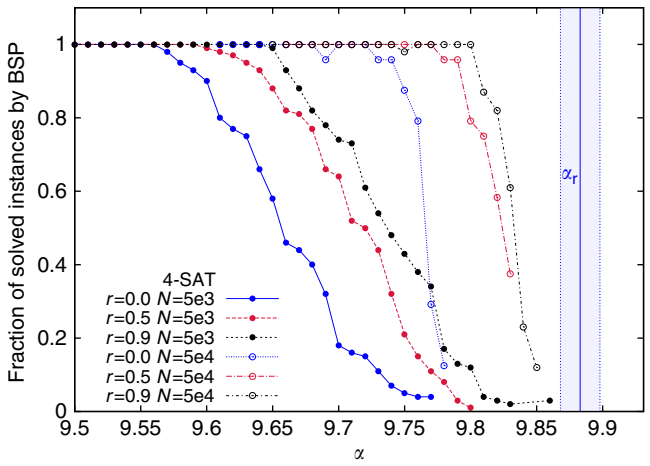

Figure 1 | Fraction of random 4-SAT instances solved by BSP as a function of the constraints per variable ratio α. The average is computed over 100 instances with $N=5{,}000$ (solid symbols) and $N=50{,}000$ (empty symbols) variables. The vertical line is the best estimate for α_r and the shaded region is the statistical error on this estimate. For each instance, the algorithm has been run once; on instances not solved on the first run, a second run rarely (<1%) finds a solution. The plot shows that the backtracking ($r>0$) definitely makes the BSP algorithm more efficient in finding solutions. Although data become sharper by increasing the problem size N, a good estimation of the algorithmic threshold from these data sets is unfeasible.

the statistical physics solution to random K-SAT problems we expect Σ_{res} to vanish linearly in α.

As we see in panel (**a**) of Fig. 2 the mean value of the intensive mean residual complexity Σ_{res}/N_{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 data sets only close to the data set right end. The linear extrapolation predicts $\alpha_a^{BSP} \approx 9.9$ (for $K=4$ and $r=0.9$), which is slightly above the rigidity threshold $\alpha_r = 9.883(15)$ computed in ref. 15 and reported in the plot with a shaded region corresponding to its statistical error (the value of α_f in this case is not known, but $\alpha_a^{BSP} < \alpha_f \leq \alpha_s$ should hold). Although for the finite sizes studied no solution has been found beyond α_r, Fig. 2 suggests that in the large N limit BSP may be able to find solutions in a region of α where the majority of solutions is in frozen clusters and thus very hard to find. We show below that BSP actually finds solutions in atypical unfrozen clusters, as it has been observed for some smart algorithms solving other kind of constraint satisfaction problems[38,39].

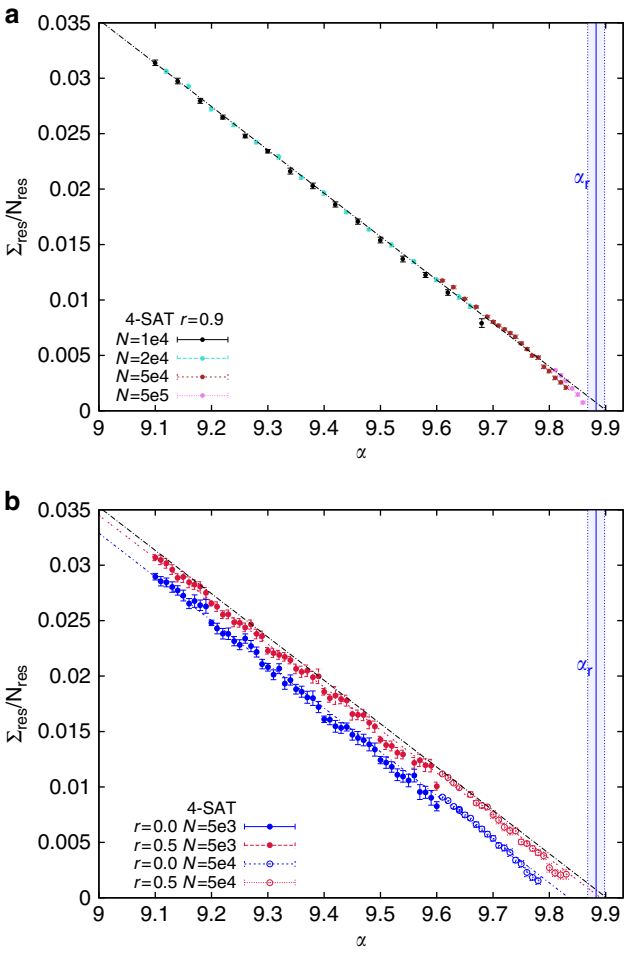

Figure 2 | BSP algorithmic threshold on random 4-SAT problems. The residual complexity per variable, Σ_{res}/N_{res}, goes to zero at the algorithmic threshold α_a^{BSP}. (**a**) The very small finite size effects, mostly producing a slight downward curvature at the right end, allow for a very reliable estimate of α_a^{BSP} via a linear fit. For random 4-SAT problems solved by BSP with $r=0.9$ we get $\alpha_a^{BSP} \approx 9.9$, slightly beyond the rigidity threshold $\alpha_r = 9.883(15)$, marked by a vertical line (the shaded area being its s.e.). (**b**) The same linear extrapolation holds for other values of r (red dotted line for $r=0.5$ and blue dashed line for $r=0$). The black line is the fit to $r=0.9$ data shown in panel (**a**). SID without backtracking ($r=0$) has a much lower algorithmic threshold, $\alpha_a^{SID} \approx 9.83$. Error bars in both panels are the s.e.m.

The effectiveness of the backtracking can be appreciated in panel (**b**) of Fig. 2, where the order parameter Σ_{res}/N_{res} is shown for $r=0$ and $r=0.5$, together with linear fits to these data sets and to the $r=0.9$ data set (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 (that is, $r=0$ data set).

For random 3-SAT the algorithmic threshold of BSP, run with $r=0.9$, practically coincide with the SAT-UNSAT threshold α_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 from $N \leq 100$ data is likely to be too small and affected by strong finite size effects, given that all solutions found by BSP for $N = 10^6$ are unfrozen, even beyond the estimated α_f. Moreover we have estimated $\alpha_r = 4.2635(10)$ improving the data of ref. 15 and the inequality $\alpha_r \leq \alpha_f \leq \alpha_s$ makes the above estimate for α_f not very meaningful.

Computational complexity. As explained in Methods, the BSP algorithm performs $f^{-1}(1-r)^{-1}$ steps roughly, 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 η of iterations. The average η is computed only on instances where SP always converges, as is usually done for incomplete algorithms (on the remaining problems the number of iterations reaches the upper limit set by the user, and then BSP exit, returning failure). Figure 4 shows that η is actually a small number changing mildly with α and N both for $K=3$ and $K=4$. The main change that we observe is in the fluctuations of η that become much larger approaching α_s. We expect η to eventually grow as $O(\log(N))$, but for the sizes studied we do not observe such a growth.

After convergence to a fixed point, the BSP algorithm just need to sort local marginals, thus the total number of elementary operations to solve an instance grows as $f^{-1}(1-r)^{-1}(a_1\eta N + a_2 N\ logN)$, where a_1 and a_2 are constants. Moreover,

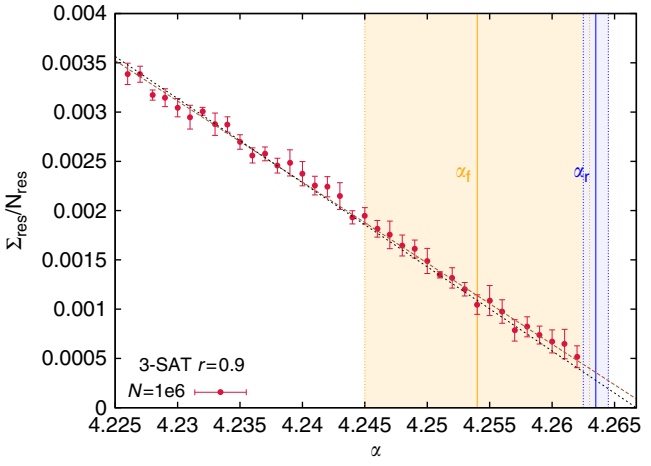

Figure 3 | BSP algorithmic threshold on random 3-SAT problems. Same as Fig. 2 for $K=3$. The estimate for the freezing threshold $\alpha_f = 4.254(9)$ measured on small problems in ref. 30 is not very meaningful, given our new estimate for the rigidity threshold $\alpha_r = 4.2635(10)$, and the observation that all solutions found by BSP are not frozen. Shaded areas are the statistical uncertainties on the thresholds. A linear fit to the residual complexity (brown line) extrapolates to zero slightly beyond the SAT-UNSAT threshold, at $\alpha_a^{BSP} \approx 4.268$, 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 α_s. Error bars are s.e.m.

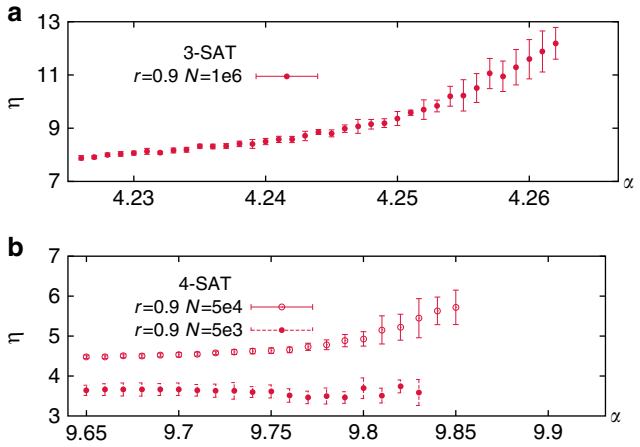

Figure 4 | BSP convergence time. The mean number η of iterations to reach a fixed point of SP equations grows very mildly with α and N, both for $K=3$ (**a**) and $K=4$ (**b**). Error bars are s.d.'s.

given that the sorting of local marginals does not need to be strict (that is, a partial sorting[40] running in $O(N)$ time can be enough), we have that in practice the algorithm runs in a time almost linear in the problem size N.

Whitening procedure. Given that the BSP algorithm is able to find solutions even very close to the rigidity threshold α_r, it is natural to check whether these solutions have frozen variables or not. We concentrate on solutions found for random 3-SAT problems with $N=10^6$, since the large size of these problems makes the analysis very clean.

On each solution found we run the *whitening procedure* (first introduced in refs 41,42 and deeply discussed in refs 26,43), that identifies frozen variables by assigning the joker state $\star$ to unfrozen (white) variables, that is, variables that can take more than one value without violating any clause and thus keeping the formula satisfied. At each step of the whitening procedure, a variable is considered unfrozen (and thus assigned to $\star$) if it belongs only to clauses which either involve a $\star$ variable or are satisfied by another variable. The procedure is continued until all variables are $\star$ or a fixed point is reached: non-$\star$ variables at the fixed point correspond to frozen variables in the starting solution.

We uncover that all solutions found by BSP are converted to all-$\star$ by running the whitening procedure, thus showing that solutions found by BSP have no frozen variables. This is somehow expected, according to the conjecture discussed in the Introduction: finding solutions in a frozen cluster would take an exponential time, and so the BSP algorithm actually finds solutions at very large α values by smartly focusing on the sub-dominant unfrozen clusters.

The whitening procedure leads to a relaxation of the number of non-$\star$ variables as a function of the number of iterations t that follows a two steps relaxation process[25] with an evident plateau, see panel (**a**) in Fig. 5, that becomes longer increasing α towards the algorithmic threshold. The time for leaving the plateau, scales as the time $\tau(c)$ for reaching a fraction c on non-$\star$ variables (with c smaller than the plateau value). The latter has large fluctuations from solution to solution, as shown in panel (**b**) of Fig. 5 for $c=0.4$ (very similar, but shifted, histograms are obtained for other c values). However, after leaving the plateau, the dynamics of the whitening procedure is the same for each solution. Indeed plotting the mean fraction of non-$\star$ variables as a function of the time to reach the

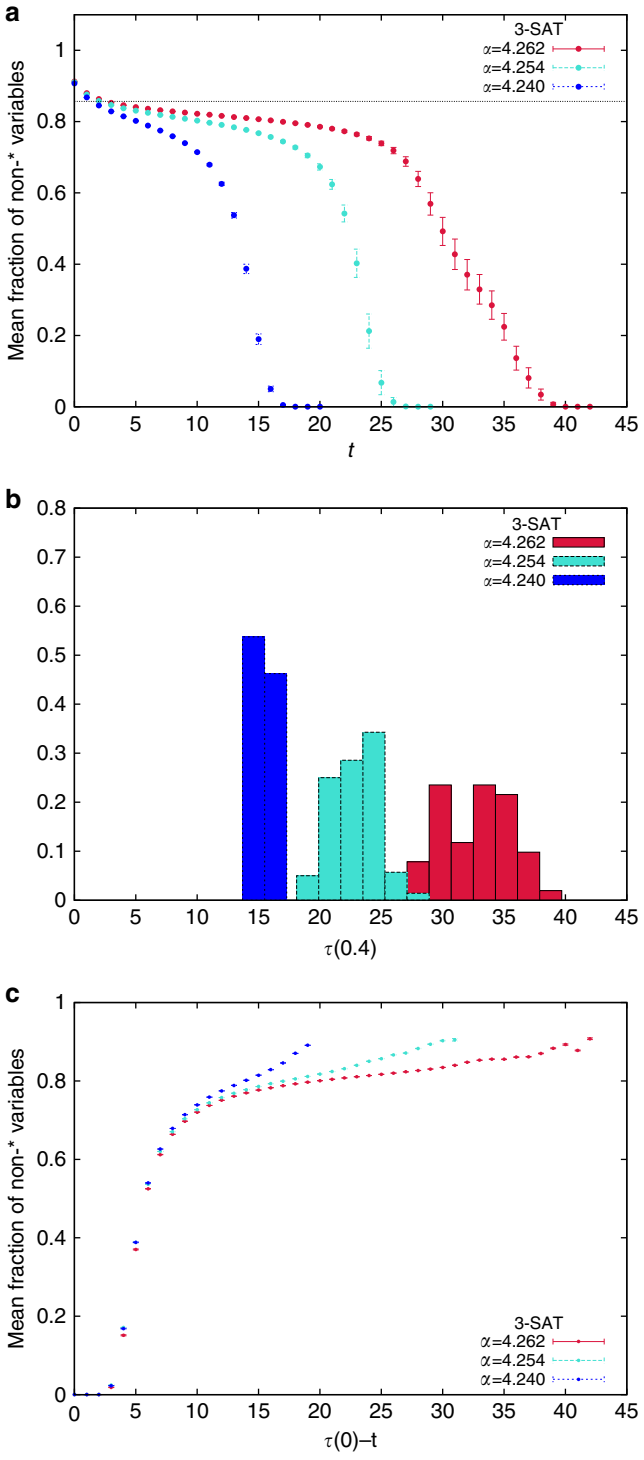

Figure 5 | Whitening random 3-SAT solutions. (**a**) The mean fraction of non-$\star$ variables during the whitening procedure applied to all solutions found by the BSP algorithm goes to zero, following a two steps process. The relaxation time grows increasing α towards the algorithmic threshold. The horizontal line is the SP prediction for the fraction of frozen variables in typical solutions at $\alpha=4.262$ and the comparison with the data shows that solutions found by BSP are atypical. (**b**) Histograms of the whitening times, defined as the number of iterations required to reach a fraction 0.4 of non-$\star$ variables. Increasing α both the mean value and the variance of the whitening times grow. (**c**) Averaging the fraction of non-$\star$ variables at fixed $\tau(0)-t$, that is, fixing the time to the all-$\star$ fixed point, we get much smaller errors than in (**a**), suggesting that the whitening procedure is practically solution-independent once the plateau is left. In all the panels, error bars are s.e.m.

all-$\star$ configuration, $\tau(0)-t$, we see that fluctuations are strongly suppressed and the relaxation is the same for each solution (see panel (**c**) in Fig. 5).

Critical exponent for the whitening time divergence. In order to quantify the increase of the whitening time approaching the algorithmic threshold, and inspired by critical phenomena, we check for a power law divergence as a function of $(\alpha_a^{BSP}-\alpha)$ or Σ_{res}, which are linearly related. In Fig. 6 we plot in a double logarithmic scale the mean whitening time $\tau(c)$ as a function of the residual complexity Σ_{res}, for different choices of the fraction c of non-$\star$ variables defining the whitening time. Data points are fitted via the power law $\tau(c)=A(c)+B(c)\Sigma_{res}^{-\nu}$, where the critical exponent ν is the same for all the c values. Joint interpolations return the following best estimates for the critical exponent: $\nu=0.281(6)$ for $K=3$ and $\nu=0.269(5)$ for $K=4$, where the uncertainties are only fitting errors. The two estimate turn out to be compatible within errors, thus suggesting a sort of universality for the critical behaviour close to the algorithmic threshold α_a^{BSP}.

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-$\star$ 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 behaviour 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 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 for example, SPR[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 constraints.

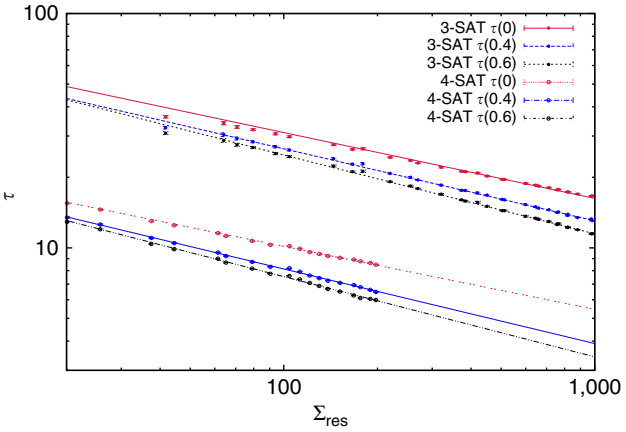

Figure 6 | Critical exponent for the whitening time divergence. The whitening time $\tau(c)$, defined as the mean time needed to reach a fraction c of non-$\star$ variables in the whitening procedure, is plotted in a double logarithmic scale as a function of Σ_{res} for random 3-SAT problems with $N=10^6$ (upper data set) and random 4-SAT problems with $N=5\times10^4$ (lower data set). The whitening time measured with different c values seems to diverge at the algorithmic threshold, where the residual complexity Σ_{res} vanishes. The lines are power law fits with exponent $\nu=0.281(6)$ for $K=3$ and $\nu=0.269(5)$ for $K=4$. Error bars are s.e.m.

For $K=3$ the BSP algorithm finds solutions practically up to the SAT-UNSAT threshold α_s, while for $K=4$ a tiny gap to the SAT-UNSAT threshold still remains, but the algorithmic threshold α_a^{BSP} seems to be located beyond the rigidity threshold α_r in the large N limit. Beating the rigidity threshold, that is, finding solutions in a region where the majority of solutions belongs to clusters with frozen variables, is hard, but not impossible (while going beyond α_f should be impossible). Indeed, 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 α_a for polynomial time algorithms solving large random K-SAT problems. Under the assumption that frozen solutions cannot be found in polynomial time, such an algorithmic threshold α_a would coincide with the freezing transition at α_f (that is, when the last unfrozen solution disappears). Unfortunately for random K-SAT the location of α_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 bicolouring problems, where the threshold values are known[44,45] 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 allows the BSP algorithm to reach solutions at large α values. Clearly we pay the price that a too frequent backtracking makes the algorithm slower, but it seems worth paying such a price to approach the SAT-UNSAT threshold closer than any other algorithm.

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 itself. 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

Survey inspired decimation. A detailed description of the survey inspired decimation (SID) algorithm can be found in refs 12,13,32. The SID algorithm is

based on the SP equations derived by the cavity method[12,13], that can be written in a compact way as

$$\hat{m}_{a\to i} = \prod_{j\in\partial_a\setminus i} m_{j\to a}, \tag{1}$$

$$m_{i\to a} = \frac{\pi^-_{ia}(1-\pi^+_{ia})}{1-\pi^+_{ia}\pi^-_{ia}}, \tag{2}$$

$$\text{with}\quad \pi^\pm_{ia} = 1-\prod_{b\in\partial^\pm_{ia}}(1-\hat{m}_{b\to i}), \tag{3}$$

where ∂_a is the set of variables in clause a, and ∂^+_{ia} (resp. ∂^-_{ia}) 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.

The interpretation of the SP equations is as follows: $\hat{m}_{a\to i}$ 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 $m_{i\to a}$ 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 $\{m_{i\to a}, \hat{m}_{a\to i}\}$ living on the edges of the factor graph[7], that are solved in an iterative way, leading to a message passing algorithm (MPA)[4], where outgoing messages from a factor graph node (variable or clause) are functions of the incoming messages. Once the MPA reaches a fixed point $\{m^*_{i\to a}, \hat{m}^*_{a\to i}\}$ that solves the SP equations, the number of clusters can be estimated via the complexity

$$\Sigma = \log N_{\rm clus} = \sum_i \Sigma_i + \sum_a (1-K_a)\Sigma_a, \tag{4}$$

$$\Sigma_a = \log(1-\prod_{j\in\partial_a} m^*_{j\to a}),\ \ \Sigma_i = \log(1-\pi^+_i\pi^-_i) \tag{5}$$

$$\text{with}\quad \pi^\pm_i = 1-\prod_{b\in\partial^\pm_i}(1-\hat{m}^*_{b\to i}), \tag{6}$$

where K_a is the length of clause a (initially $K_a=K$) and ∂^+_i (resp. ∂^-_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 ($1-w^+_i-w^-_i$)

$$w^\pm_i = \frac{\pi^\pm_i(1-\pi^\mp_i)}{1-\pi^+_i\pi^-_i}. \tag{7}$$

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

$$b_i = 1-min(w^+_i, w^-_i). \tag{8}$$

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 that 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 Σ during the SID algorithm can be very informative[35]. Indeed it is found that, if Σ 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 Σ always remains well positive, the SID algorithm reduces so much the problem, that eventually a trivial SP fixed point, $m^*_{i\to a}=\hat{m}^*_{a\to i}=0$, 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 $\alpha_a^{\rm SID}=4.2525$ (ref. 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 $O(N\log(N))$[32].

Backtracking survey propagation. Willing to improve the SID algorithm to find solutions also in the region $\alpha_a^{\rm SID}<\alpha<\alpha_s$, one has to change the way variables are assigned. The fact the SID algorithm assigns each variable only once is clearly 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 backtracking survey propagation (BSP) algorithms[36] 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 $\{\hat{m}^*_{a\to i}\}_{a\in\partial_i}$ 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 assigning other variables, then it is likely that releasing the variable x_i may be beneficial in the search for a solution. So both the variables to be fixed in the decimation step and the variables to be released in the backtracking step are chosen according to their biases b_i: the variables to be fixed have the largest biases and the 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 variables are reassigned on average $1/(1-r)$ times each before the BSP algorithm reaches the end, but its complexity remains at most $O(N\log N)$ 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 Σ 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.

Data availability statement. The numerical codes used in this study and the data that support the findings are available from the corresponding author upon request.

References

1. Cook, S. A. *The complexity of theorem proving procedures*, Proc. 3rd Ann. 151–158 (ACM Symp. on Theory of Computing, Assoc. Comput. Mach., New York, 1971).
2. Garey, M. & Johnson, D. S. *Computers and Intractability; A guide to the theory of NP-completeness* (Freeman, 1979).
3. Papadimitriou, C. H. *Computational Complexity* (Addison-Wesley, 1994).
4. Mézard, M. & Montanari, A. *Information, Physics, and Computation* (Oxford University Press, 2009).
5. Moore, C. & Mertens, S. *The Nature of Computation* (Oxford University Press, 2011).
6. Cook, S. A. & Mitchell, D. G. in *Discrete Mathematics and Theoretical Computer Science*, Vol. 35 (eds Du, J., Gu, D. & Pardalos, P.)American Mathematical Society, 1997).
7. Kschischang, F. R., Frey, B. J. & Loeliger, H.-A. Factor graphs and the sum-product algorithm. *IEEE Trans. Infor. Theory* **47,** 498–519 (2001).
8. Kirkpatrick, S. & Selman, B. Critical behaviour in the satisfiability of random Boolean expressions. *Science* **264,** 1297–1301 (1994).
9. Dubois, O., Boufkhad, Y. & Mandler, J. *Typical random 3-SAT formulae and the satisfiability threshold*, in Proc. 11th ACM-SIAM Symp. on Discrete Algorithms, 126–127 (San Francisco, CA, USA, 2000).
10. Dubois, O., Monasson, R., Selman, B. & Zecchina, R. (eds.)Phase transitions in combinatorial problems. *Theoret. Comp. Sci.* **265,** 1–2 (2001).
11. Ding, J., Sly, A. & Sun, N. *Proof of the satisfiability conjecture for large k*, Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing (Portland, OR, USA, 2015).
12. Mézard, M., Parisi, G. & Zecchina, R. Analytic and algorithmic solution of random satisfiability problems. *Science* **297,** 812–815 (2002).
13. Mézard, M. & Zecchina, R. The random K-satisfiability problem: from an analytic solution to an efficient algorithm. *Phys. Rev. E* **66,** 056126 (2002).
14. Mertens, S., Mézard, M. & Zecchina, R. Threshold values of random K-SAT from the cavity method. *Random Struct. Alg.* **28,** 340–373 (2006).
15. Montanari, A., Ricci-Tersenghi, F. & Semerjian, G. Clusters of solutions and replica symmetry breaking in random k-satisfiability. *J. Stat. Mech.* P04004 (2008).
16. Mézard, M., Ricci-Tersenghi, F. & Zecchina, R. Two solutions to diluted p-spin models and XORSAT problems. *J. Stat. Phys.* **111,** 505–533 (2003).

17. Krzakala, F., Montanari, A., Ricci-Tersenghi, F., Semerjian, G. & Zdeborova, L. Gibbs states and the set of solutions of random constraint satisfaction problems. *PNAS* **104,** 10318–10323 (2007).
18. Zdeborova, L. & Krzakala, F. Phase transitions in the coloring of random graphs. *Phys. Rev. E* **76,** 031131 (2007).
19. Achlioptas, D. & Coja-Oghlan, A. *Algorithmic Barriers from Phase Transitions*, in FOCS '08: Proceedings of the 49th annual IEEE symposium on Foundations of Computer Science, 793–802 (Philadelphia, PA, USA, 2008).
20. Achlioptas, D., Coja-Oghlan, A. & Ricci-Tersenghi, F. On the solution-space geometry of random constraint satisfaction problems. *Random Struct. Alg.* **38,** 251–268 (2011).
21. Ricci-Tersenghi, F. & Semerjian, G. 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. Sakari, S., Alava, M. & Orponen, P. Focused local search for random 3-satisfiability. *J. Stat. Mech.* P06006 (2005).
23. Ardelius, J. & Aurell, E. Behavior of heuristics on large and hard satisfiability problems. *Phys. Rev. E* **74,** 037702 (2006).
24. Monasson, R., Zecchina, R., Kirkpatrick, S., Selman, B. & Troyansky, L. Determining computational complexity from characteristic phase transitions. *Nature* **400,** 133–137 (1999).
25. Semerjian, G. On the freezing of variables in random constraint satisfaction problem. *J. Stat. Phys.* **130,** 251–293 (2008).
26. Maneva, E., Mossel, E. & Wainwright, M. J. A new look at survey propagation and its generalizations. *JACM* **54,** 17 (2007).
27. Braunstein, A., Dall'Asta, L., Semerjian, G. & Zdeborova, L. The large deviations of the whitening process in random constraint satisfaction problems. *J. Stat. Mech.* P053401 (2016).
28. Achlioptas, D. & Ricci-Tersenghi, F. *On the Solution-Space Geometry of Random Constraint Satisfaction Problems*, STOC '06: Proceedings of the 38th annual ACM symposium on Theory of Computing 130–139, (Seattle, 2006).
29. Achlioptas, D. & Ricci-Tersenghi, F. Random formulas have frozen variables. *SIAM J. Comput.* **39,** 260–280 (2009).
30. Ardelius, J. & Zdeborova, L. Exhaustive enumeration unveils clustering and freezing in the random 3-satisfiability problem. *Phys. Rev. E* **78,** 040101 (2008).
31. Selman, B., Kautz, H. A. & Cohen, B. Noise strategies for improving local search. *in Proc. AAAI-94* 337–343 (Seattle, WA, USA, 1994).
32. Braunstein, A., Mézard, M. & Zecchina, R. Survey propagation: an algorithm for satisfiability. *Random Struct. Alg.* **27,** 201–226 (2005).
33. Frieze, A. & Suen, S. Analysis of two simple heuristics on a random instance of k-SAT. *J. Algor.* **20,** 312–355 (1996).
34. Mulet, R., Pagnani, A., Weigt, M. & Zecchina, R. Coloring random graphs. *Phys. Rev. Lett.* **89,** 268701 (2002).
35. Parisi, G. Some remarks on the survey decimation algorithm for K- satisfiability, preprint at http://arxiv.org/abs/cs/0301015 (2003).
36. Parisi, G. A backtracking survey propagation algorithm for K-satisfiability, preprint at http://arxiv.org/abs/cond-mat/0308510 (2003).
37. Chavas, J., Furtlehner, C., Mézard, M. & Zecchina, R. Survey-propagation decimation through distributed local computations. *J. Stat. Mech.* P11016 (2005).
38. Dall'Asta, L., Ramezanpour, A. & Zecchina, R. Entropy landscape and non-Gibbs solutions in constraint satisfaction problems. *Phys. Rev. E* **77,** 031118 (2008).
39. Zdeborová, L. & Mézard, M. Constraint satisfaction problems with isolated solutions are hard. *J. Stat. Mech.* P12004 (2008).
40. Chambers, J. M. Algorithm 410: partial sorting. *Commun. ACM* **14,** 357–358 (1971).
41. Parisi, G. *On local equilibrium equations for clustering states*, preprint at http://arxiv:cs/0212047 (2002).
42. Parisi, G. On the survey-propagation equations in random constraint satisfiability problems. *J. Math Phys* **49,** 125216 (2008).
43. Braunstein, A. & Zecchina, R. Survey Propagation as local equilibrium equations. *J. Stat Mech.* P06007 (2004).
44. Castellani, T., Napolano, V., Ricci-Tersenghi, F. & Zecchina, R. Bicolouring random hypergraphs. *J. Phys. A* **36,** 11037 (2003).
45. Coja-Oghlan, A. & Zdeborová, L. *The condensation transition in random hypergraph 2-coloring*, Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms (Kyoto, Japan, 2012).

Acknowledgements

We thank K. Freese, R. Eichhorn and E. Aurell for useful discussions. This research has been supported by the Swedish Science Council through grant 621-2012-2982 and by the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme (grant agreement No [694925]).

Author contributions

All authors contributed to all aspects of this work.

Additional information

Competing financial interests: The authors declare no competing financial interests.

Reprints and permission information is available online at http://npg.nature.com/reprintsandpermissions/

How to cite this article: Marino, R. *et al.* The backtracking survey propagation algorithm for solving random K-SAT problems. *Nat. Commun.* **7**,12996 doi: 10.1038/ncomms12996 (2016).