EPJ manuscript No.
(will be inserted by the editor)

Phase Transitions of the Typical Algorithmic Complexity of the Random Satisfiability Problem Studied with Linear Programming

Hendrik Schawe[1a], Roman Bleim[1], and Alexander K. Hartmann[1b]

Institut für Physik, Universität Oldenburg, 26111 Oldenburg, Germany

April 6, 2018

Abstract. Here we study the NP-complete Satisfiability problem for N Boolean variables, in particular K-SAT, for which the Boolean formula has the conjunctive normal form with M clauses and $K < N$ possibly negated variables per clause. Although the worst-case complexity of NP-complete problems is conjectured to be exponential, there exist parametrized random ensembles of problems where solutions can *typically* be found in polynomial time for suitable values of the parameter. In fact, random K-SAT, with $\alpha = M/N$ as control parameter, shows a phase transition between a satisfiable phase and an unsatisfiable phase. For branch and bound algorithms, which operate in the space of feasible Boolean configurations, the empirically hardest problems are located only close to this phase transition. Here we study K-SAT ($K = 3, 4$) and the related optimization problem MAX-SAT by *linear programming approach*, which is widely used for practical problems and allows for polynomial run time. In contrast to branch and bound it operates outside the space of feasible configurations. On the other hand, finding a solution within the polynomial time is not guaranteed. We investigated several variants like including artificial objective functions, cutting-plane approaches, and a mapping to the NP-complete vertex-cover problem. We observed several *easy-hard transitions*, from where the problems are typically solvable (in polynomial time) using the given algorithms, respectively, to where they are not solvable in polynomial time. For comparison, we studied random K-SAT in regard to structural *percolation transitions* of the equivalent *factor graphs* with respect to connectivity, two-edge connectivity, q-core, and two types of leaf-removal cores, respectively. These transitions were mostly not studied in the literature before, and might be of interest on their own.

PACS. 89.70.Eg Computational complexity – 02.10.Ox Combinatorics; graph theory – 64.60.-i General studies of phase transitions

1 Introduction

The *Satisfiability problem* (SAT) [1] is to decide whether some Boolean formula is satisfiable or not, i.e., whether for a given Boolean formula, there is an *assignment* of the variables such that the formula evaluates to "true". SAT is the most-prominent NP (nondeterministic-polynomial) problem. In particular it is *NP-complete* [2], which means that all problems in NP can be mapped in polynomial time to SAT and that it is hard to solve with all so-far known algorithms. All Boolean formulas can be expressed in *conjunctive normal form* (CNF) which is a disjunction of *clauses*, each being a conjunction of variables or negated variables. This preserves satisfiability with only linear increase in problem size [3], see next section for a precise definition of CNF. Therefore K-SAT, which is a Boolean formula in CNF with K distinct variables per clause, is a commonly scrutinized version of the satisfiability problem.

Random K-SAT is an ensemble of Boolean formulas, where for a set of N variables M clauses are generated randomly. Each clause contains K variables which are chosen randomly, and each variable appears negated with probability 0.5. Interestingly, this problem shows a *phase transition* at some critical value α_s of the density $\alpha = M/N$ [4]. For large problems at $\alpha < \alpha_s$ almost all problems are satisfiable (also denoted as SAT), above α_s almost all realizations are unsatisfiable (UNSAT). The occurrence of similar phase transitions has been observed frequently for random ensembles of NP-complete problems [4–7], for which 3-SAT is the prime example [2,8]. This incited strong interest in the K-SAT problem [9–12] and many other NP-complete problems [13–18] among physicists.

For the SAT-UNSAT transition it was found that the hardest realizations are located near this α_s. This can be roughly understood because it is trivial to find some solution for much lower values of α and relatively easy to prove a realization unsatisfiable at high values of α.

While this SAT-UNSAT transition is certainly the most scrutinized in the K-SAT problem, there exist more transitions. For example 3-SAT, where SAT-UNSAT oc-

[a] *Present address:* hendrik.schawe@uni-oldenburg.de
[b] *Present address:* a.hartmann@uni-oldenburg.de

curs at $\alpha_\mathrm{s} \approx 4.26$ [9], shows a transition to chaotic behavior at $\alpha_\chi \approx 3.28$ [19], i.e., using a continuous time deterministic solver [20] the trajectory will find the solution if one exists, but it will show chaotic transient behavior above this threshold resulting in increasing escape rates from attractors. This leads to a higher computational cost and can therefore be used as a measure of hardness. Furthermore, there exists a clustering transition at $\alpha_\mathrm{c} \approx 3.86$ [11,21]. This means that here the organization of the space of the exponentially many degenerate solutions changes from one big cluster ($\alpha < \alpha_\mathrm{c}$) of solutions which are connected in assignment space to a solution space ($\alpha > \alpha_\mathrm{c}$) which is fragmented into many non-connected smaller clusters.

Usually algorithms like the *branch and bound approach*, *stochastic search* or *message passing* are used in the statistical-mechanics literature. These algorithms operate in the space of feasible assignments and approach the optimum solution from above. Here "optimum" means that the number of unsatisfied clauses is minimized, i.e., eventually becomes zero if a satisfying assignment is found. Thus for general minimization problems these algorithms yield upper bounds until the true minimum solution is found. On the other hand, the operations-research literature often uses *linear programming* (LP) [22,23] techniques for real-world applications since they are versatile and efficient, which means they run typically in polynomial time. For combinatorial problems, e.g., NP-hard optimization problems, LP yields solutions which are not necessarily feasible, which here means non-integer-valued assignments to the variables, but which establish a lower bound on the objective. Thus LP somehow approaches for minimization problems the true feasible and optimum solution from below. Nevertheless, a key observation is that whenever LP gives a feasible solution, it must be the true optimum solution of the combinatorial problem. Because of their complementary properties combinations of LP with other approaches yield powerful methods, such as *branch and cut* algorithms [24]. Here *cutting planes* are employed, which are additional inequalities which decrease LP solution space to help finding feasible solutions. Therefore LP and how it behaves for ensembles of random NP-hard problems should be given more attention in the physics community.

Until now, studies of the behavior of LP have only been conducted for the vertex cover (VC) [25] and the traveling salesperson problem (TSP) [26]. Indeed there exist regions in parameter space, where feasible and optimal solutions can be found in polynomial time. For VC, the LP approach yielded solutions up to the percolation transition of the underlying graph ensemble. Therefore, the problem is *easy* with respect to LP up to the percolation transition, and *hard* beyond. For LP improved with cutting planes, another easy-hard transitions occurs at the point of the onset of replica symmetry breaking [16]. Note that this coincides with the percolation threshold for the *leaf-removal core* [27]. This is the point where one would not reasonably expect easy instances anymore. For TSP the easy-hard transitions detected by the LP approach coincided with structural changes of the optimal tour which can be intuitively understood as increases in hardness. Since for TSP many more cutting planes exist, which are not yet tested, it is conceivable to use this technique to find more and more easy-hard transitions this way and understand them, leading to deeper insight into the problem.

Since K-SAT is the archetypal NP-complete problem, we wanted to extend those promising results of the mentioned previous studies. As we will show below, we found also some "easy-hard" transitions. Nevertheless, these transitions occur at much lower values of the parameter α than the clustering transition, in contrast to VC on random graphs. The reason could be that SAT is a decision problem while VC is an optimization problem. Nevertheless, although MAX-SAT, the canonical optimization formulation of K-SAT, is an optimization problem, it shows the same properties as K-SAT in our examination, so this is maybe not the explanation. This result could be related to the fact that K-SAT shows anyway a richer behavior [11], i.e., several different types of transition, in contrast to VC on random graphs. Anyway, we cannot explain this result in the moment, and think it will motivate further studies which aim at the origin of the different behavior.

Further we scrutinized some percolation properties of the factor graph representation of K-SAT. These new results should be interesting in themselves.

In the following we will first present the methods we used in Sec. 2. In Sec. 3.1 we will present the results of the LP approach, before Sec. 3.2 shows different percolation thresholds that we would assume change the hardness of the problem. Section 3.3 lists further attempts. And Sec. 4 concludes the article.

2 Methods

2.1 K-SAT

A realization of K-SAT consists of a Boolean formula over N variables x_i ($i = 1, \ldots, N$). The formula is in conjunctive normal form, i.e., it is a conjunction of M clauses c_j ($j = 1, \ldots, M$), where every clause is a disjunction of K literals l_{kj} ($k = 1, \ldots, K$). A literal is a variable x_i or its negation $\overline{x}_i$. In each clause, each variable may appear only once. As an example for $N = 4$, $M = 2$ and $K = 3$ take

$$(\overline{x}_1 \vee x_2 \vee \overline{x}_3) \wedge (x_1 \vee x_3 \vee \overline{x}_4). \qquad (1)$$

This example is solvable with, e.g., $x_1 =$ "true" $= 1$ and $x_3 =$ "false" $= 0$, and arbitrary assignments for the other variables. Note that each clause is satisfiable by $2^K - 1$ out of 2^K possible assignments to the variables. Thus, each clause restricts the space of satisfiable assignments a bit. Clearly, with more clauses per variable, i.e., a higher amount of constraints, it is more probable that a random formula is unsatisfiable. As mentioned before, for random 3-SAT with $N \to \infty$ there is a critical density $\alpha_\mathrm{s} = M/N \approx 4.26$ [9] at which a phase transition from satisfiable to unsatisfiable (SAT-UNSAT) happens.

2.2 Linear Programming

A linear program (LP) is an optimization problem, which can be expressed by a set of linear constraints and a linear objective function, which should be optimized. There are fast (polynomial-time) algorithms to solve a linear program, e.g., the ellipsoid method [28] or interior point methods [29,30]. However, in many sophisticated solvers the simplex algorithm is used, which typically terminates quickly for real-world problems, despite its exponential worst-case time complexity [22,23]. Though, as soon as some variables need to be integer valued, this problem gets computationally hard. In fact, *integer linear programming* is an NP-hard problem [8].

A K-SAT realization can be expressed as an integer linear program. Therefore every positive literal x_i is expressed as an integer variable x_i and every negative literal $\overline{x}_i$ as $(1 - x_i)$. Since one or more literals of every clause $c \in C$ need to be true for a satisfying assignment, the corresponding integer linear program contains for each clause the constraint that the sum of the expressions for the included literals must be greater or equal 1. The example from Eq. (1) generates following linear inequalities.

$$\begin{aligned}(1-x_1)+x_2+(1-x_3)&\geq 1\\ x_1+x_3+(1-x_4)&\geq 1\end{aligned}$$

Since an LP is an optimization problem but SAT is merely a decision problem, we can choose an arbitrary objective function for which to optimize. The simplest objective function is zero, i.e., no optimization.

$$\begin{aligned}&\text{min.} && 0 &&\\ &\text{s.t.} && \sum_{x_i\in c_j} x_i + \sum_{\overline{x}_i\in c_j}(1-x_i)\geq 1, && \forall 1\leq j\leq M\\ &&& x_i\in\{0,1\}, && \forall 1\leq i\leq N\end{aligned}$$

The last constraint fixes the variables to integer values. We will relax this constraint to $x_i \in [0, 1]$. This allows us to apply a fast LP algorithm to solve the relaxed problem and introduce a measure of hardness for the problem realization. If the LP relaxation yields a solution consisting of only integer variables, the solution is obtained by a polynomial time method and the corresponding realization is obviously easy to solve.

A drawback is that additionally to the principal degeneracy of the problem, i.e., there are possibly many assignments that satisfy the formula, the relaxation leads to a much higher degeneracy. For example, the assignment of all $x_i = 0.5$ is always a solution of the relaxation. After we performed some simulations for SAT in this way, it became evident that this degeneracy is a major problem for this decision problem, which is not present in the optimization problems studied with this method [25,26]. This degeneracy leads to different behavior for slight changes in the algorithm. Also, primal and dual simplex often lead to different behavior such that for many instances one version will result in an integer solution while the other does not. We observed a similar behavior when considering different pricing strategies or a presolve stage to tighten the LP. This analysis would therefore only yield information about these technical details and not about the fundamental problem of K-SAT. For example, the presolver of both Gurobi and CPLEX can solve easy instances up to an critical $\alpha_{\text{pre}} = 1.640(1)$, which is the same threshold up to which the *pure-literal* rule (explained later) can solve K-SAT realizations, while without presolve the easy-hard transition occurs at lower α – dependent on technical details of the method. Therefore, we do not present results of LP with zero objective in is study.

Instead, we will introduce artificial objective functions to reduce the degeneracy drastically. Further, the choice of the objective function has an influence on the prevalence of integer solutions. Though note, only linear objective functions enable the efficient linear programming techniques. Therefore, non-linear objectives like $\sum_i x_i(1 - x_i)$, which are minimal if all variables x_i are either 1 or 0, are not admissible and in fact generally NP-hard [31].

One simple way to replace the zero objective is maximizing the sum over all variables (MV)

$$\text{max.} \qquad \sum_{i=1}^{N} x_i,$$

which will on average lift variables which are 0 in the integer solution to larger values like 0.5 and thus suppresses integer solutions typically.

Note that this and other additional objective functions have no influence on whether a formula is satisfiable or not, they are just meant as a tool to reduce the degeneracy of the problem to make it less dependent on details of the algorithm and to facilitate finding integer solutions. Both works out as we will see below. As a third objective we tried maximizing the number of fulfilled literals per clause, which we will call *Satisfaction Multiplicity Maximization* (SMM). This can be achieved with a slightly modified linear program by introducing one new variable z_i per clause counting the number of fulfilled literals of its clause and maximizing the sum over all z_i.

$$\begin{aligned}&\text{max.} && \sum_{i=1}^{M} z_i &&\\ &\text{s.t.} && \sum_{x_i\in c_j} x_i + \sum_{\overline{x}_i\in c_j}(1-x_i)\geq z_j, && \forall 1\leq j\leq M\\ &&& x_i\in\{0,1\}, && \forall 1\leq i\leq N\\ &&& z_i\geq 1, && \forall 1\leq j\leq M\end{aligned}$$

The new kind of constraint ensures that $z_i \geq 1$, i.e., that every clause contains at least one fulfilled literal, such that the solution assignment satisfies the Boolean formula. This type of additional optimization is similar to MAX-SAT, where one tries to maximize the number of satisfied clauses. For MAX-SAT one would instead enforce $0 \leq z_i \leq 1\ \forall i$. We also tried this MAX-SAT approach to solve the K-SAT decision problem. Since this formulation does not mitigate the degeneracy problem, and because we did not observe any better performance than by using the other approaches, we do not show results for this approach here.

The SMM objective is an example for a linear objective with a slight preference for integer valued variables, since assignment of all variables of a clause to 1 or 0 according to their polarity contributes more to the objective function than assignments of non-integer values. For example a variable which appears more often unnegated will on average be assigned more often to value 1. This strongly reduces the degeneracy of the solution of the optimization problem, i.e., many of the solutions where the majority of variables are non-integer, are not optimal under this new objective function. Of course this may still yield non-integer values for some variables, which occur in conflicting clauses.

Note that when using LP, finding integer solutions may be facilitated in principle by adding so called *cutting planes*. This was previously observed also for ensembles of random instances for the vertex cover [25] and the TSP [26]. Nevertheless, for the present study this yields no improvement, i.e., no additional phase transitions could be observed, see below. Thus, we do not describe the cutting-plane approach in this section in detail.

2.3 Graph Representation of K-SAT Realizations

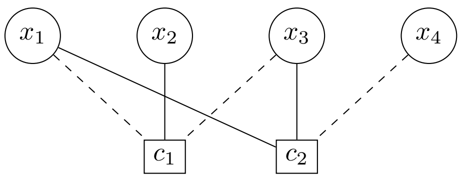

Fig. 1. The factor graph of the example from Eq. (1). The negation of a variable is marked by a dashed line.

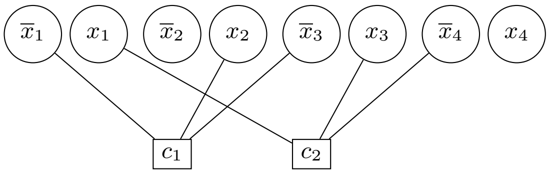

Fig. 2. The literal factor graph of the example from Eq. (1).

We introduce a graph representation of the satisfiability problem, we will call the *literal factor graph* (LFG). While one of the most scrutinized graph representations of SAT is its *factor graph* (FG) [32,6], especially useful for belief or survey propagation [33,34], the authors do not know of any study about the LFG. Both are conceptually similar. In the FG, which is a bipartite graph exhibiting a node for each variable and each clause, i.e., containing $N+M$ nodes, each variable is connected to the clauses it occurs in as shown in Fig. 1 for the example Eq. (1). For the LFG, which contains $2N+M$ nodes for *literals* and clauses, each literal is connected to the clauses it occurs in. The LFG representation of the example from Eq. (1) is given in Fig. 2. Note that the ensemble of LFG for some α is statistically the same as the factor graph ensemble at 2α. We will use these graph representations as a tool to detect changes in the problem structure and compare them to the solvability by LP techniques, since previous work on vertex cover and TSP showed [25,26] that some LP easy-hard transitions coincide with changes of the problem or solution structure.

2.4 Mapping to Vertex Cover

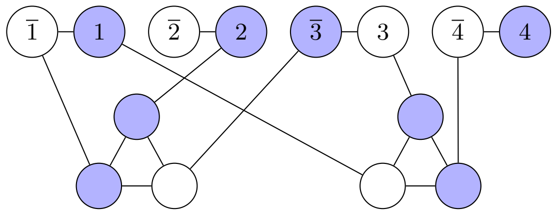

Fig. 3. The graph for the vertex cover problem which is equivalent to the formula shown in Eq. (1). Shown is a vertex cover of size $N+(K-1)M = 4+2\times 2 = 8$, which corresponds to a satisfying assignment $x_1 = 1$, $x_2 = 1$, $x_3 = 0$, $x_4 = 1$.

All NP-complete problems, by definition, can be mapped onto each other in polynomial time. Thus, it is reasonable to ask, whether transforming SAT instances to instances of another problem and applying algorithms specifically suited for the other problem changes the performance, as measured by the location of the easy-hard transition. Here we used a classical mapping [1] of SAT to the vertex cover problem. For each K-SAT instance an equivalent graph $G=(V,E)$ is constructed in the following way: The set V of nodes contains one pair of nodes $i, \overline{i}$ for each variable x_i ($i=1,\ldots,N$), which represents the variable and its negation. Furthermore, V contains one node (kj) for each literal l_{kj} in each clause c_j, respectively. Therefore V contains $2N+KM$ nodes. For the set E of edges, for each clause c_j a complete subgraph of size K is formed by connecting all pairs of "literal nodes" (kj) pairwise which correspond to this clause. Also, each "variable node" i is connected with its corresponding "negated variable node" $\overline{i}$. Finally, for each literal l_{kj}, if the literal represents a non-negated variable x_i, an edge connecting (kj) with i is included, while if the literal represents a negated variable $\overline{x}_i$ the corresponding literal node (kj) is connected with $\overline{i}$. Thus, E contains $MK(K-1)/2+N+MK$ edges. Now a minimum vertex cover is obtained. This is a subset $V' \subset V$ of nodes such that for each edge of E at least one of the two endpoints is in V'. By construction [1], G contains a vertex cover of size $N+(K-1)M$ if and only if the corresponding formula is satisfiable. In Fig. 3 the graph corresponding to the formula from Eq. (1) is shown.

Thus, one approach to SAT is to transform each formula into the equivalent graph and use an existing algorithm for VC to solve it. We applied an LP formulation with cycle cutting planes, see Ref. [25] for details. For the previous work, this algorithm was able to solve VC instances in the parameter-space region, where the solutions were contained basically in one cluster, corresponding to the replica-symmetric region [35].

3 Results

We sample random 3-SAT instances, where each clause may contain any variable at most once. For up to 14 system sizes $N \in [128, 524288]$ we simulated $n = 5000$ realizations for 30 to 100 different values of the density α. All error estimates are obtained by bootstrap resampling [36–38], except for errors of fit parameters shown in the plots, which are *gnuplot*'s asymptotic standard errors corrected according to Ref. [37]. To solve the LP realizations, the implementation of the dual-simplex algorithms of the commercial optimization library *CPLEX* [39] is used. During the research additionally the primal- and dual-simplex implementations of *Gurobi* [40] and *lp_solve* [41] with multiple pricing strategies were used to ensure that the results are independent from the algorithm and the details of the implementation.

3.1 LP-Transitions with Objective Function

As mentioned before, we observed that non-trivial objective functions can be used to obtain a result independent from the details of the LP-solver implementation. Though the objective function itself will have an influence on the position of the transition point. An objective which prefers variables to be integer will result in more integer solutions at the same α, i.e., a transition at a larger value of α.

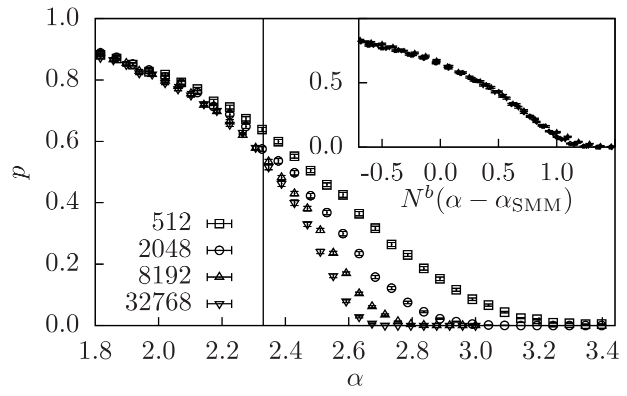

Fig. 4. Probability p that SMM yields an integer solution. The smaller system sizes show visible deviations from the common curves, which is visible in the main plot, where $N = 512$ does not lie on the other curves at α_{SMM} marked by the vertical line. Inset: Collapse with $b = 0.118(1)$, $\alpha_{\mathrm{SMM}} = 2.36(1)$ for $N \geq 4096$.

Plotting the probability that the SMM solution of a realization is integer, i.e., the actual solution, as a function of α in Fig. 4 shows a decrease at some apparent critical value α_{SMM}. This decrease is steeper for larger system sizes N, which is a behavior typical for phase transitions. The transition depicted here is an algorithmic transition from *easy*, since all realizations are solvable by LP techniques, i.e., in polynomial time, to some *harder* phase, where the LP does not yield solutions.

To estimate the point of this transition α_{SMM}, we use finite-size scaling, i.e., rescaling the α axis according to $(\alpha - \alpha_{\mathrm{SMM}})N^b$, should result in a *collapse* of the data points on one curve. To find values α_{SMM} and b generating a good collapse the algorithm and implementation from Ref. [42] is used. It yields $\alpha_{\mathrm{SMM}} = 2.361(7)$ and $b = 0.118(1)$. The collapse obtained when rescaling with those values is shown in the inset of Fig. 4.

As a cross-check, we determine the position of the peaks of the variance $\mathrm{Var}(p)$ of the solution probability like in previous work [25,26], see Fig. 5. Since the single measurements can only take the values 1 or 0, the maximum of the variance is $\max \mathrm{Var}(p) = 0.25$ at $p = 0.5$. Therefore, this method effectively extrapolates the position of the $p = 0.5$ point, which should be at the transition point for $N \to \infty$. The positions are extrapolated with a power law $\alpha_{\max}(N) = \alpha_{\mathrm{SMM}} - aN^{-\tilde{b}}$ as typical for second order phase transitions. Since the data points are almost on a horizontal line and the exponent $\tilde{b}$ is therefore very small, we can not get a reliable fit for this exponent, but using the exponent from the collapse, leads to a compatible critical density $\alpha_{\mathrm{SMM}} = 2.35(1)$ and a reasonable fit, with $\chi^2_{\mathrm{red}} = 1.9$, as shown in in Fig. 5.

For 4-SAT the exponent seems to be larger and a fit through the positions of the maxima yields $\alpha_{\mathrm{SMM}} = 3.60(8)$, $\tilde{b} = 0.5(1)$ with $\chi^2_{\mathrm{red}} = 0.5$ (not shown, simulations used smaller system sizes).

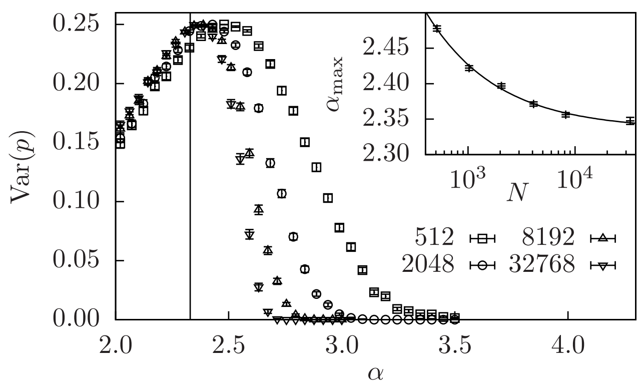

Fig. 5. Variance of the solution probability p for the SMM objective. Inset: The power-law fit $\alpha_{\max} = aN^{-\tilde{b}} + \alpha_{\mathrm{SMM}}$ to the position of the peaks $\alpha_{\max}$. For fixed $\tilde{b} = 0.1176$ the fit yields $\alpha_{\mathrm{SMM}} = 2.35(1)$ with $\chi^2_{\mathrm{red}} = 1.9$.

The other optimization function of this study, MV, i.e., maximizing the sum of all variables, leads to a qualitatively similar behavior as SMM but a transition at lower $\alpha_{\mathrm{MV}} = 1.25(2)$ (not pictured, simulations used smaller system sizes). The lower transition point is plausible, since this maximization prefers variables to be larger than zero instead of 0. For this reason, we have not analyzed this algorithm for 4-SAT. The best estimates for the transition points are collected in Tab. 1.

Table 1. Values of critical points. α_{VC} denotes the critical point when mapping SAT to VC and applying an LP + cutting plane solver used for VC. α_{MV} is the easy-hard transition for LP+MV. α_{SMM} is the easy-hard transition for LP+SMM. α_{c} denotes the clustering transition [11] and α_{s} the SAT-UNSAT transition [9,43].

K	α_{VC}	α_{MV}	α_{SMM}	α_{c}	α_{s}
3	0.90(3)	1.25(2)	2.361(7)	3.86	4.26
4	–	–	3.60(8)	9.547	9.93

3.2 Structural Analysis of the Factor Graph

The previous work on vertex cover and TSP showed [25, 26] that some LP easy-hard transitions coincide with changes of the problem or solution structure. Therefore we study the factor-graph representation of SAT with respect to percolation transitions. We found two percolation transitions that do coincide with each other. Though we did not find one coinciding with the transitions of the optimizing LP formulations. We will nevertheless show them, since these percolation transitions of this factor graph could be interesting on their own.

The standard percolation (SP) threshold is known to be $p_c = \frac{1}{K(K-1)}$ [6]. We look additionally at the transitions, where the *two-edge-connected component* (2EC) [44] percolates, i.e., a component such that every node can reach every other node by two distinct paths. Third, we investigate the percolation of the *q-core* [45] for $q \in \{2, 3\}$. The q-core is the remainder of a graph after all nodes with degree $q-1$ or lower are removed iteratively. Note that the 2-core of a realization is a superset of the 2EC, since 2-core can not remove nodes from a two-edge-connected component thus a 2-core consists of two-edge-connected components connected by single edges.

We determined the percolation-transition points similar to the method used above. This means, we extrapolated the position of the maximum of the variance of the size of the core or the 2EC. For the extrapolation to large values of N we use a power law again. As an example the inset of Fig. 6 shows this for 3-core and apparently the power-law to extrapolate fits nicely.

We do not include graphs for the other cases, since they look very similar. The resulting percolation points are shown in Tab. 2. Note that the percolation thresholds for 2EC and $q = 2$ are within statistical errors compatible with each other. Also note that the 4-core is always empty since the clause-nodes have at most $K = 3$ neighbors and are thus directly removed. Since the factor graph is bipartite, this leaves only isolated nodes, i.e., an empty 4-core.

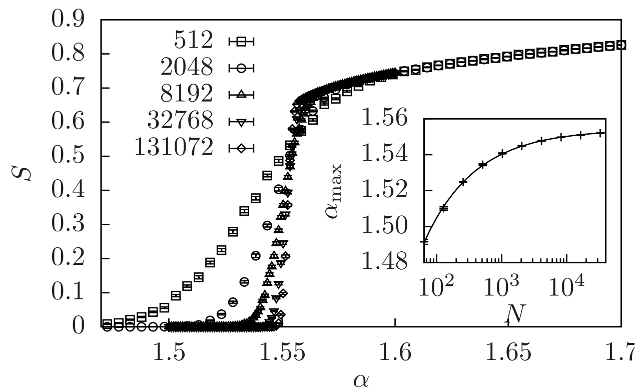

Fig. 6. Size of the 3-core at different values of α and N. The inset shows the extrapolation of the position of maximum variance similar to Fig. 5. The power law $\alpha_{\max} = c - aN^{-b}$ with $c = 1.554(1)$ and $b = 0.55(1)$ fits.

Further, we looked at *leaf removal* [27], where iteratively neighbors of leaf-nodes are removed. Leaf removal is a simplification rule for vertex cover, but also suited to simplify XORSAT realizations, where it is mostly studied. Since K-XORSAT uses the same CNF structure, it shares the same factor-graph representation with K-SAT. For leaf removal of the factor graph an exact first order transition point at $\alpha_{\mathrm{lr}} = 0.818469..$ is known ($\alpha_{\mathrm{lr}} = 0.772278..$ for 4-SAT) [46], which are also included in Tab. 2. For XORSAT this is the clustering transition, where replica symmetry breaking occurs. K-SAT realizations can also be simplified with leaf removal by setting the leafs of the factor graph, i.e., variables occurring only in one clause, such that they satisfy this clause and removing the satisfied clause and all isolated nodes from the problem. This can be iterated until a graph without leafs, the *leaf-removal core*, remains. If the core is empty, the realization is fully solved by leaf removal. In the example in Fig. 1 node 2 and node 4 are leafs, such that the problem is solved after one iteration.

Table 2. Percolation thresholds for different observables explained in the text. Note that the percolation transitions of PL and LR coincide. From the two values of FG and LFG only one is measured, while the other is calculated from the measured one.

width=1

K		SP	2EC	2-core	3-core	PL	LR
3	FG	1/6	0.219(1)	0.223(3)	1.554(1)	0.818(1)	0.8184
	LFG	1/3	0.438(2)	0.446(6)	3.108(2)	1.636(2)	1.6369
4	FG	1/12				0.772(2)	0.7722
	LFG	1/6				1.544(3)	1.5445

Further, we studied the *pure-literal rule* (PL), also called *affirmative-negative rule*, which is an integral part of the DPLL [47,48] search algorithm. A *pure literal* is a variable which occurs only in one polarity, either positive (also called affirmative) or negative in the formula. In the graph this means, if only one polarity of a variable (x_i or $\overline{x}_i$) is connected to one or more clauses, this literal can be set to true, thus fulfilling all adjacent clauses, which can be removed from the graph without affecting the satisfiability of the realization. Also, all isolated nodes are removed from the graph. This procedure can be iterated until no pairs of literals, of which only one is connected to any clause, are left. The remainder of the graph is called *pure-literal core*. If the pure-literal core is satisfiable, the realization is satisfiable. Especially, an empty pure-literal core is trivially satisfiable.

The example Fig. 2 would find that $\overline{x}_2$ is isolated and thus remove all neighbors of x_2, in this case c_1. In the next step, since after removal of c_1 the node $\overline{x}_1$ is isolated, c_2, the neighbor of x_1, is removed. The remaining pure-literal core is empty and this realization satisfiable.

In Fig. 7 the size of the pure-literal core S, i.e., the fraction of literal nodes still remaining, is plotted as a function of α. It is clearly visible that there happens a transition from "no core" to "core". We use the intersection positions $\alpha_\times$ of all pairs of curves with the system sizes $N_1 = 2N_2$ with $N_1 \geq 4096$ and extrapolate the intersections to the large N limit, similar as shown before, with the power law $\alpha_\times = aN_1^{-b} + \alpha_{\text{pl}}$. This way, we estimate $\alpha_{\text{pl}} = 1.636(2)$ and $b = 0.5(3)$, with $\chi^2_{\text{red}} = 0.5$ for 3-SAT and $\alpha_{\text{pl}} = 1.544(3)$ with $\chi^2_{\text{red}} = 0.4$ for 4-SAT (cf. inset of Fig. 7, smaller system sizes were used, the uncertainty of b is too large to make any meaningful statement about b). Other values of K were not examined.

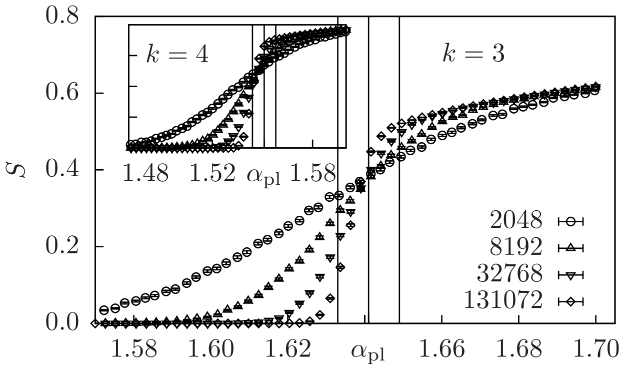

Fig. 7. Size of the pure-literal core after simplification according to the pure-literal rule for 3-SAT in the main plot and 4-SAT in the inset. The vertical lines are values for the critical α_{pl} obtained by the power-law extrapolation of the intersections mentioned in the text.

Interestingly, this seems to be the same threshold of the core/no core transition for leaf removal on the LFG, although leaf removal on the LFG is not a valid simplification rule for K-SAT. Note however that these two rules are quite different and the cores, for $\alpha > \alpha_{\text{pl}}$ for both, are substantially different from each other. Also note that there are also instances which have a pure literal core but no leaf-removal core. Despite of this, the previous analysis shows that both methods show for the large N limit no core below α_{pl} but some above α_{pl}.

For the easy-hard transitions we found, as shown in Tab. 1, we were not able to identify structural transitions (see Tab. 2) of the ensemble of SAT instances or of the solution space structure, which coincide with the observed easy-hard transitions. This is in contrast to the previous results for VC and TSP.

On the other hand, the clustering transition point α_{c} and for sure the SAT-UNSAT transition point α_{s} are well above the easy-hard transitions found here. So it remains to be determined in which way the easy-hard transition for LP+SMM corresponds to a (possible hard to determine) structural change of the problem.

Nevertheless the detected easy-hard transitions are far beyond the percolation transition. In the case of VC the pure LP method did only yield solutions up to the standard percolation transition of the underlying graph ensemble and more sophisticated techniques were required to overcome this threshold.

Thus, in summary, the behavior of random K-SAT appears from the linear-programming perspective to be richer as compared to VC and TSP. This is parallel to the richer behavior observed when treating these problems analytically.

3.3 Further results

In similar studies on vertex cover [25] and the TSP [26], the introduction of cutting planes yielded substantially better results. For vertex cover the introduction of (potentially exponentially many but actually few) cutting planes (CP) even lead to an LP+CP transition at the point where in the analytic solution replica symmetry breaking, i.e., clustering of solutions appeared [35]. This efficiency of CP was not observable here. It seems that the cutting planes we tried, namely *resolution cuts* [49] and *clique cuts* [50], were too weak at the low values of α examined here. Another cutting plane for the SAT problem, the *odd cycle inequalities* [51] are not directly applicable for K-SAT with $K \geq 3$, since they need clauses with 2 variables to be constructed. While they are useful as local cuts in a branch and cut procedure, they are never violated in the beginning for $K \geq 3$ and thus not applicable for this study.

Finally, we also used for $K = 3$ the mapping of K-SAT to the vertex cover problem for formulas up to $N = 10000$. Using LP and cycle cutting planes [25] the corresponding equivalent instances were solved. Again we measured the probability that an instance was solved by an integer solution as a function of α, for different system sizes. Using an analysis (not shown) as for the previous approaches, we were able to extrapolate an easy-hard transition for this point. We obtained a critical value of $\alpha_{\text{VC}} = 0.90(3)$, which is well below the easy-hard transitions obtained using the LP-based approaches presented above. Therefore, apparently it does not pay off using a mapping to another problem, at least for this pair of problems. A mapping to TSP will lead to TSP instances which grow quadratic in N for fixed α, therefore we did not test this.

4 Conclusions

We study the solvability of random K-SAT realizations at different values of the clause-to-variable density α using linear programming. A realization is solved if the LP yields an integer solution. Since there are LP algorithms that run in polynomial time, this means such a realization is "easy".

Since SAT is a decision problem, the application of pure LP actually does not involve optimization, which seems to make the problem less well-behaved. Therefore we included artificial objective functions, which makes the problem numerically better behaved. Note that we also observed that usually more realizations can be solved when including an artificial objective function as compared to the trivial objective function.

We were able to identify easy-hard transitions. The inclusion of artificial objective functions lead to higher values of the control parameter α, where solutions can still be found. In contrast to previous work on VC on random graphs, non of these transitions coincided with the onset of a clustered solution landscape. The reason might be that K-SAT, or "natural" decision problems in general behave differently. On the other hand, random K-SAT exhibits several transitions for the structure of the solution landscape anyway, in contrast to VC on random graphs. It would be interesting to investigate the relation between LP and solution landscape more thoroughly in the future.

Anyway, from the practical point of view, we wonder if carefully crafted objective functions could be used to improve the efficiency of solving decision problems with an LP approach, such as branch and cut.

We identified different structural transitions based on the factor and literal factor graphs, like for standard percolation, q-core, 2-edge connected component, leaf-removal and pure literal rule. Most of these transitions have not been described in the literature so far. We were not able to identify a structural transition coinciding with an easy-hard transition detected by any of the tested LP formulations. Nevertheless, we believe that such a coincidence should exist, since the properties of a problem should be coded in the graph structure. Therefore, this unknown structural property seems to be of a more complex type. The fact that the relation between LP-based easy-hard transitions and structural properties appears to be more complex might also be related to the mentioned richer behavior of the solution landscape as a function of the density α (in comparison with VC). Still, we also did not observe any structural transition corresponding to these clustering transitions of the solution space. Thus, here is still some work to be done.

Furthermore, in contrast to the previously studied vertex cover problem and the traveling salesperson problem, we did not observe any improvement by applying cutting planes. This could be due to the type of cutting planes used. Finally, we applied a mapping of SAT to VC and used specific VC LP+cutting plane algorithm, which is able to solve standard Erdős-Rényi VC instances just up to the clustering threshold. Nevertheless, for SAT this did not pay off, the easy-hard transition appears to be at even smaller values of α compared to the MV objective.

5 Acknowledgments

This work was supported by the German Science Foundation (DFG) through the grant HA 3169/8-1. The simulations were performed at the HPC Cluster HERO and CARL, both located at the University of Oldenburg (Germany) and funded by the DFG through its Major Research Instrumentation Programme (INST 184/108-1 FUGG and INST 184/157-1 FUGG) and the Ministry of Science and Culture (MWK) of the Lower Saxony State.

6 Authors contributions

All authors contributed ideas, simulation data and analysis to this study. All the authors were involved in the preparation of the manuscript.

References

1. M.R. Garey, D.S. Johnson, *Computers and intractability* (W.H. Freemann, San Francisco, 1979)
2. S.A. Cook, *The Complexity of Theorem-proving Procedures*, in *Proceedings of the Third Annual ACM Symposium on Theory of Computing* (ACM, New York, NY, USA, 1971), STOC '71, pp. 151–158
3. G.S. Tseitin, Zapiski Nauchnykh Seminarov POMI **8**, 234 (1968)
4. P. Cheeseman, B. Kanefsky, W.M. Taylor, *Where the really hard problems are.*, in *IJCAI* (1991), Vol. 91, pp. 331–337
5. A.K. Hartmann, M. Weigt, *Phase Transitions in Combinatorial Optimization Problems* (Wiley-VCH, Weinheim, 2005)
6. M. Mézard, A. Montanari, *Information, Physics and Computation* (Oxford University Press, Oxford, 2009)
7. C. Moore, S. Mertens, *The Nature of Computation* (Oxford University Press, Oxford, 2011)
8. R.M. Karp, *Reducibility among combinatorial problems* (Springer, 1972)
9. M. Mézard, G. Parisi, R. Zecchina, Science **297**, 812 (2002)
10. G. Biroli, S. Cocco, R. Monasson, Physica A: Statistical Mechanics and its Applications **306**, 381 (2002), invited Papers from the 21th IUPAP International Conference on Statistical Physics
11. F. Krzakała, A. Montanari, F. Ricci-Tersenghi, G. Semerjian, L. Zdeborová, Proceedings of the National Academy of Sciences **104**, 10318 (2007), see papercore summary `http://www.papercore.org/Krzakala2007`
12. S. Cocco, R. Monasson, Phys. Rev. Lett. **86**, 1654 (2001)

13. O.C. Martin, R. Monasson, R. Zecchina, Theoretical computer science **265**, 3 (2001)
14. I.P. Gent, T. Walsh, Artificial Intelligence **88**, 349 (1996)
15. L. Zdeborová, F. Krzakała, Phys. Rev. E **76**, 031131 (2007)
16. M. Weigt, A.K. Hartmann, Phys. Rev. Lett. **84**, 6118 (2000), see papercore summary `http://www.papercore.org/Weigt2000`
17. A.K. Hartmann, M. Weigt, Journal of Physics A: Mathematical and General **36**, 11069 (2003)
18. M. Weigt, A.K. Hartmann, Phys. Rev. Lett. **86**, 1658 (2001)
19. M. Varga, R. Sumi, Z. Toroczkai, M. Ercsey-Ravasz, Physical Review E **93**, 052211 (2016)
20. M. Ercsey-Ravasz, Z. Toroczkai, Nature Physics **7**, 966 (2011)
21. A. Mann, A.K. Hartmann, Physical Review E **82**, 056702 (2010)
22. C. Papadimitriou, K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Dover Books on Computer Science Series (Dover Publications, 1998), ISBN 9780486402581
23. T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, *Introduction to algorithms* (MIT press Cambridge, 2009)
24. M. Padberg, G. Rinaldi, SIAM Review **33**, 60 (1991)
25. T. Dewenter, A.K. Hartmann, Physical Review E **86**, 041128 (2012), see papercore summary `http://www.papercore.org/Dewenter2012`
26. H. Schawe, A.K. Hartmann, Eur. Phys. Lett. **113**, 30004 (2016)
27. M. Bauer, O. Golinelli, The European Physical Journal B-Condensed Matter and Complex Systems **24**, 339 (2001)
28. L.G. Khachiyan, USSR Computational Mathematics and Mathematical Physics **20**, 53 (1980)
29. N. Karmarkar, *A new polynomial-time algorithm for linear programming*, in *Proceedings of the sixteenth annual ACM symposium on Theory of computing* (ACM, 1984), pp. 302–311
30. Y. Nesterov, A. Nemirovskii, *Interior-point polynomial algorithms in convex programming*, Vol. 13 (Siam, 1994)
31. S. Sahni, SIAM Journal on Computing **3**, 262 (1974)
32. A. Montanari, G. Parisi, F. Ricci-Tersenghi, Journal of Physics A: Mathematical and General **37**, 2073 (2004)
33. M. Mézard, R. Zecchina, Phys. Rev. E **66**, 056126 (2002)
34. A. Braunstein, M. Mézard, R. Zecchina, Random Structures & Algorithms **27**, 201 (2005)
35. M. Weigt, A.K. Hartmann, Phys. Rev. Lett. **84**, 6118 (2000)
36. B. Efron, Ann. Statist. **7**, 1 (1979)
37. A.P. Young, *Everything You Wanted to Know About Data Analysis and Fitting but Were Afraid to Ask*, SpringerBriefs in Physics (Springer International Publishing, 2015), ISBN 978-3-319-19050-1
38. A.K. Hartmann, *Big Practical Guide to Computer Simulations* (World Scientific, 2015)
39. IBM, *IBM ILOG CPLEX Optimization Studio* (2013), `https://www.ibm.com/support/knowledgecenter/en/SSSA5P_12.6.0`, `https://www.ibm.com/support/knowledgecenter/en/SSSA5P_12.6.0`
40. Gurobi Optimization, Inc., *Gurobi Optimizer Reference Manual* (2016), `http://www.gurobi.com`, `http://www.gurobi.com`
41. M. Berkelaar, K. Eikland, P. Notebaert, *lp_solve reference guide* (2016), `http://lpsolve.sourceforge.net/5.5/`, `http://lpsolve.sourceforge.net/5.5/`
42. O. Melchert, *autoscale.py - a program for automatic finite-size scaling analyses: A user's guide* (2009), arXiv:0910.5403 [physics.comp-ph], `http://arxiv.org/abs/0910.5403`
43. S. Mertens, M. Mézard, R. Zecchina, Random Structures & Algorithms **28**, 340 (2006)
44. R. Tarjan, SIAM Journal on Computing **1**, 146 (1972)
45. B. Pittel, J. Spencer, N. Wormald, Journal of Combinatorial Theory, Series B **67**, 111 (1996)
46. M. Mézard, F. Ricci-Tersenghi, R. Zecchina, Journal of Statistical Physics **111**, 505 (2003)
47. M. Davis, H. Putnam, J. ACM **7**, 201 (1960)
48. M. Davis, G. Logemann, D. Loveland, Commun. ACM **5**, 394 (1962)
49. J.N. Hooker, Operations Research Letters **7**, 1 (1988)
50. A. Atamtürk, G.L. Nemhauser, M.W. Savelsbergh, European Journal of Operational Research **121**, 40 (2000)
51. S. Joy, J. Mitchell, B. Borchers, *A branch and cut algorithm for MAX-SAT and weighted MAX-SAT*, in *Satisfiability Problem: Theory and Applications* (1997), Vol. 35 of DIMACS Series on Discrete Mathematics and Theoretical Computer Science