Limits of Sequential Local Algorithms on the Random k-XORSAT Problem

Kingsley Yung
The Chinese University of Hong Kong, Hong Kong, China

Abstract

The random k-XORSAT problem is a random constraint satisfaction problem of n Boolean variables and $m = rn$ clauses, which a random instance can be expressed as a $G\mathbb{F}(2)$ linear system of the form $Ax = b$, where A is a random $m \times n$ matrix with k ones per row, and b is a random vector. It is known that there exist two distinct thresholds $r_{core}(k) < r_{sat}(k)$ such that as $n \to \infty$ for $r < r_{sat}(k)$ the random instance has solutions with high probability, while for $r_{core} < r < r_{sat}(k)$ the solution space shatters into an exponential number of clusters. Sequential local algorithms are a natural class of algorithms which assign values to variables one by one iteratively. In each iteration, the algorithm runs some heuristics, called local rules, to decide the value assigned, based on the local neighborhood of the selected variables under the factor graph representation of the instance.

We prove that for any $r > r_{core}(k)$ the sequential local algorithms with certain local rules fail to solve the random k-XORSAT with high probability. They include (1) the algorithm using the Unit Clause Propagation as local rule for $k \geq 9$, and (2) the algorithms using any local rule that can calculate the exact marginal probabilities of variables in instances with factor graphs that are trees, for $k \geq 13$. The well-known Belief Propagation and Survey Propagation are included in (2). Meanwhile, the best known linear-time algorithm succeeds with high probability for $r < r_{core}(k)$. Our results support the intuition that $r_{core}(k)$ is the sharp threshold for the existence of a linear-time algorithm for random k-XORSAT.

Our approach is to apply the Overlap Gap Property OGP framework to the sub-instance induced by the core of the instance, instead of the whole instance. By doing so, the sequential local algorithms can be ruled out at density as low as $r_{core}(k)$, since the sub-instance exhibits OGP at much lower clause density, compared with the whole instance.

2012 ACM Subject Classification Mathematics of computing → Combinatorial algorithms

Keywords and phrases Random k-XORSAT, Sequential local algorithms, Average-case complexity, Phase transition, Overlap gap property

Digital Object Identifier 10.4230/LIPIcs.ICALP.2024.123

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2404.17775`

Acknowledgements The author thanks Andrej Bogdanov for his guidance and many insightful discussions.

1 Introduction

1.1 Background

The k-XORSAT problem is a Boolean constraint satisfaction problem and closely related to the well-known k-SAT problem. An instance Φ of the k-XORSAT problem consists of m clauses in n Boolean variables. Each clause is a Boolean linear equation of k variables of the form $x_{j_1} \oplus x_{j_2} \oplus \cdots \oplus x_{j_k} = b_j$, where $\oplus$ is the modulo-2 addition. By convention, when we say an XORSAT instance, without the prefix "k", we mean the same except we do not require the clauses to have exactly k variables. An **assignment** σ to the n variables is a mapping from the set $\{x_i : i \in [n]\}$ of all n variables to the set $\{0, 1\}$ of the two Boolean values. By abusing

51st International Colloquium on Automata, Languages, and Programming (ICALP 2024).
Editors: Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson;
Article No. 123; pp. 123:1–123:20


Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

the notation, we can write it as the Boolean vector $\sigma = (\sigma(x_1), \sigma(x_2), \cdots, \sigma(x_n)) \in \{0,1\}^n$ containing the assigned values. The **distance** $d(\sigma, \sigma')$ between any two assignments σ and σ' is defined to be the Hamming distance $d(\sigma, \sigma') = \sum_{i=1}^{n} \mathbb{1}(\sigma(x_i) \neq \sigma'(x_i))$. A clause is **satisfied by** an assignment if the equation of the clause holds when the variables are replaced by the corresponding assigned values, and an instance of the k-XORSAT problem is **satisfied by** an assignment if all its clauses are satisfied by the assignment. An instance is **satisfiable** if it has at least one satisfying assignment, or **unsatisfiable** if it does not have any satisfying assignment. The assignment σ that satisfies the instance Φ is called a **solution** for the instance. The set of all satisfying solutions for the instance Φ is called the **solution space** of the instance, denoted by $\mathcal{S}(\Phi)$. We are interested in the complexity of finding a solution.

Since each clause is just a Boolean linear equation, an instance Φ can be viewed as a Boolean linear system $Ax = b$, where $A \in \{0,1\}^{m \times n}$ is an $m \times n$ Boolean matrix and $b \in \{0,1\}^n$ is a vector of length n. Note that each row in A contains exactly k ones, since each clause has exactly k variables. We can see that finding solutions for a k-XORSAT instance is equivalent to solving a Boolean linear system, and the solution space $\mathcal{S}(\Phi)$ is an affine space inside $\{0,1\}^n$. By abusing the notation, we can simply write $\Phi = (A, b)$, and the terms "clause" and "equation" are interchangeable here.

We are particularly interested in random instances of the k-XORSAT problem. In a random instance $\mathbf{\Phi}$, each clause is drawn over all $2\binom{n}{k}$ possibilities, independently. In particular, the left-hand side of the equation is the modulo-2 sum of k variables chosen uniformly from $\binom{n}{k}$ possibilities, and the right-hand side is either 0 or 1 with even probabilities. Therefore, a random instance $\mathbf{\Phi}$ of the k-XORSAT problem is drawn uniformly from the ensemble $\mathbf{\Phi}_k(n, m)$ of all possible instances of the k-XORSAT problem with n variables and m clauses, each clause containing exactly k variables, and we denote this by $\mathbf{\Phi} \sim \mathbf{\Phi}_k(n, m)$. We focus on the regime in which the number of variables n goes to the infinity and the number of clauses m is proportional to the number of variables n, that is, $m = rn$, where r is a constant independent of n and called the *clause density*.

Since a k-XORSAT instance can be represented by a system of linear equations in $G\mathbb{F}(2)$, given an instance, some standard linear algebra methods such as the Gaussian elimination can determine whether the instance is satisfiable, find a solution, and even count the total number of solutions, in polynomial time. However, beyond this particular algebraic structure, some variants of the k-XORSAT problem is hard to solve. Achlioptas and Molloy mentioned in their paper [6] that random instances of the k-XORSAT problem seems to be extremely difficult for both generic CSP solvers and SAT solvers, which do not take the algebraic structure into account. Guidetti and Young [22] suggested that the random k-XORSAT is the most difficult for random walk type algorithms such as WalkSAT, among many random CSPs. The difficulty of solving random k-XORSAT instances becomes more apparent when we only consider linear-time algorithms as efficient algorithms, since we do not have linear-time algebraic method to solve a linear system in general.

Many studies suggest that the difficulties of solving random CSPs are related to the *phase transition* of the solution spaces when the clause density r grows. (We will have more detailed discussion in Section 1.3.) Pittel and Sorkin [34] obtained the sharp *satisfiability threshold* $r_{sat}(k)$ of random k-XORSAT, for general $k \geq 3$: The random k-XORSAT instance is satisfiable w.h.p. when $r < r_{sat}(k)$, and it is unsatisfiable w.h.p. when $r > r_{sat}(k)$. (We say an event $\mathcal{E}_n$, depending on a number n, occurs **with high probability**, or shortened to **w.h.p.**, if the probability of the event $\mathcal{E}_n$ occurring converges to 1 as n goes to the infinity, that is, $\lim_{n\to\infty} \Pr[\mathcal{E}_n] = 1$.) Furthermore, Ibrahimi, Kanoria, Kraning and Montanari [25] obtained the sharp *clustering threshold* $r_{core}(k)$, which is less than $r_{sat}(k)$, of random k-

XORSAT for $k \geq 3$. When $r < r_{core}(k)$, w.h.p. the solution space of a random k-XORSAT instance is "well-connected". When $r_{core}(k) < r < r_{sat}(k)$, w.h.p. the solution space of a random k-XORSAT instance shatters into an exponential number of "well-separated clusters". In [25], They also provided a linear-time algorithm that can solve a random k-XORSAT instance w.h.p. for $r < r_{core}(k)$. On the other hand, no algorithm is known to be able to find a solution for a random k-XORSAT instance with non-vanishing probability in linear time, for $r_{core}(k) < r < r_{sat}(k)$ in which solutions exist with high probability.

In this work, we consider a natural class of algorithms, called *sequential local algorithms*. A sequential algorithm selects an unassigned variable randomly and assigns a value to it, iteratively, until every variable has an assigned value. In each iteration of the algorithm, to decide the assigned value, the algorithm runs some heuristic called *local rules* which return a value $p \in [0, 1]$, and decide the assigned value to be either 0 or 1 randomly, according to the Bernoulli distribution with parameter p. Ideally, if in each iteration the local rule can calculate the exact marginal probability of the selected variable over a randomly chosen solution for the instance conditioned on fixing all previously selected variables to their assigned values, the algorithm should be able to find a solution. However, we restrict the ability of the local rules by only providing *local* structure to the local rules. To explain the meaning of "local", we first construct a graphical representation for the k-XORSAT instances: the *factor graph*. The *factor graph* G of a k-XORSAT instance Φ is constructed in the following way: (1) each variable is represented by a *variable node*; (2) each equation is represented by an *equation node*; (3) add an undirected edge (v, e) if the variable v is involved in the equation e. Note that since there is a one-to-one correspondence between variables (equations) and variable nodes (equation nodes), in this paper, the terms *variables (equations)* and *variable nodes (equation nodes)* are interchangeable. The *distance* between any two nodes is the number of edges in the shortest path connecting the two nodes. For any integer $R \geq 0$, the *local neighborhood* $B_G(v, R)$ with radius R of a node v is the subgraph of G induced by all nodes with distances less than or equal to R from the node v. By "local" in the name of the algorithms, it means the local rules only takes the local neighborhood of the selected variable, of radius R, as its input.

The actual implementation of a sequential local algorithm depends on the choice for the local rules. To emphasize the choices for the local rules of the algorithms, the sequential local algorithm with the given local rule τ is called the *τ-decimation algorithm* $\texttt{DEC}_\tau$. The formal definitions of the sequential local algorithms, as well as the τ-decimation algorithms, will be given in Section 1.4. Note that if the local rule τ takes constant time, then the τ-decimation algorithm also takes linear time.

We introduce a notion of *freeness* to the sequential local algorithms. For any iteration in which the local rule returns $1/2$, we call it a *free step*. Intuitively, a free step means the local rule cannot obtain useful information from the local structure and let the algorithms make a random guess for the assigned value. We say a τ-decimation algorithm is *δ-free* if w.h.p. it has at least δn free steps. Moreover, we say a τ-decimation algorithm is *strictly δ-free* if it is δ'-free for some $\delta' > \delta$. If the τ-decimation algorithm is δ-free with large $\delta > 0$, it means the algorithm makes a lot of random guess on the assigned values, and it is likely that the local rule τ cannot extract useful information from the local structure to guide the algorithm. This leads to our contribution described in the next section.

1.2 Main contribution

The main contribution of this work consists of two parts. The first part is to show that as $n \to \infty$ if the τ-decimation algorithm is strictly $2\mu(k, r)$-free then w.h.p. it fails to find a solution for the random k-XORSAT instance, when the clause density r is beyond the

clustering threshold $r_{core}(k)$ but below the satisfiability threshold $r_{sat}(k)$. This can be formally written as Theorem 1. The value $\mu(k, r)$ given in Theorem 1 is an upper bound of the number of variables removed from the instance in order to obtain the sub-instance called *core instance*, which is crucial in our proof. We will discuss its meaning in detail in Section 2.

▶ **Theorem 1.** *For any $k \geq 3$ and $r \in (r_{core}(k), r_{sat}(k))$, if the τ-decimation algorithm $\mathtt{DEC}_\tau$ is strictly $2\mu(k, r)$-free on the random k-XORSAT instance $\mathbf{\Phi} \sim \mathbf{\Phi}_k(n, rn)$, then w.h.p. the output assignment $\mathtt{DEC}_\tau(\mathbf{\Phi})$ from the algorithm $\mathtt{DEC}_\tau$ on input $\mathbf{\Phi}$ is not in the solution space $\mathcal{S}(\mathbf{\Phi})$, that is,*

$$\lim_{n\to\infty} \Pr\left[\, \mathtt{DEC}_\tau(\mathbf{\Phi}) \in \mathcal{S}(\mathbf{\Phi}) \,\right] = 0,$$

where Q is the largest solution of the fixed point equation $Q = 1 - \exp(-krQ^{k-1})$, and $\mu(k, r)$ is the real-valued function given by $\mu(k, r) = \exp(-krQ^{k-1}) + krQ^{k-1}\exp(-krQ^{k-1})$.

Note. This theorem can also be applied to the *non-sequential local algorithms* in which the algorithms run their local rules and decide the assigned value for each variable, *in parallel*, without depending on the other assigned values. We will briefly discuss the reason at the end of Section 1.6 where we discuss the proof technique we used.

The best known linear algorithm of finding a solution for the random k-XORSAT instance succeeds w.h.p., for $k \geq 3$ and $r < r_{core}(k)$ [25]. That means these sequential local algorithms do not outperform the best known linear algorithm. Note that $r_{core}(k)$ is where the best known linear algorithm succeeds up to, and where the sequential local algorithms starts failing. These support the intuition that $r_{core}(k)$ is the sharp threshold of the existence of a linear-time algorithm for random k-XORSAT problem.

The second part of our contribution is to verify that the "freeness" condition in Theorem 1 is satisfied by the τ-decimation algorithm with certain local rules τ. One of them is the simplest local rule, the Unit Clause Propagation `UC`, which tries to satisfy the unit clause on the selected variable if exists, or make random guess otherwise. By using the Wormald's method of differential equations to count the number of free steps run by `UC`-decimation algorithm $\mathtt{DEC}_{\mathtt{UC}}$, we can show that it is strictly $2\mu(k, r)$-free on the random k-XORSAT instance $\mathbf{\Phi}$ for $k \geq 9$, which leads to the following theorem.

▶ **Theorem 2.** *For any $k \geq 9$, $r \in (r_{core}(k), r_{sat}(k))$, given a random k-XORSAT instance $\mathbf{\Phi} \sim \mathbf{\Phi}_k(n, rn)$, we denote by $\mathtt{DEC}_{\mathtt{UC}}(\mathbf{\Phi})$ the output assignment from the `UC`-decimation algorithm $\mathtt{DEC}_{\mathtt{UC}}$ on input $\mathbf{\Phi}$. Then, we have $\lim_{n\to\infty} \Pr\left[\, \mathtt{DEC}_{\mathtt{UC}}(\mathbf{\Phi}) \in \mathcal{S}(\mathbf{\Phi}) \,\right] = 0$.*

In each iteration, the role of the local rules is to calculate the marginal probability of the selected variable in the instance conditioned on fixing all previously selected variables to their assigned values. Belief Propagation `BP` and Survey Propagation `SP` are surprisingly good at approximating marginal probabilities of variables over randomly chosen solutions in many random constraint satisfaction problems empirically [29, 21, 7, 36, 27]. In particular, it is well-known that they can calculate the exact marginal probabilities of variables when the underlying factor graph is a tree, which is proved analytically. If Belief Propagation `BP` and Survey Propagation `SP` are used as the local rule τ, it is natural to expect that the τ-decimation algorithm can find a solution. However, we prove that even the local rule τ can give the exact marginal probabilities of variables over a randomly chosen solution for any instance whose factor graph is a tree, the τ-decimation algorithm still cannot find a solution w.h.p. for $k \geq 13$. We know that w.h.p. the local neighborhood of the factor graph of the random k-XORSAT instance is a tree. Therefore, running `BP` and `SP` on the

local neighborhood actually gives the exact marginal probabilities of the selected variables, with respect to the sub-instance induced by the local neighborhood. This implies that both BP-decimation algorithm $\texttt{DEC}_{\texttt{BP}}$ and SP-decimation algorithm $\texttt{DEC}_{\texttt{SP}}$ fail to find a solution w.h.p. for $k \geq 13$.

▶ **Theorem 3.** *For any $k \geq 13$, $r \in (r_{core}(k), r_{sat}(k))$, given a random k-XORSAT instance $\mathbf{\Phi} \sim \mathbf{\Phi}_k(n, rn)$, denote by $\texttt{DEC}_\tau(\mathbf{\Phi})$ the output assignment from the τ-decimation algorithm $\texttt{DEC}_{\texttt{UC}}$ on input $\mathbf{\Phi}$. Assume the local rule τ outputs the exact marginal probability of a selected variable for any instance whose factor graph is a tree. Then, we have* $\lim_{n\to\infty} \Pr\left[\,\texttt{DEC}_\tau(\mathbf{\Phi}) \in \mathcal{S}(\mathbf{\Phi})\,\right] = 0$.

To prove Theorem 2 and Theorem 3, we only need to calculate the number of free steps in $\texttt{DEC}_{\texttt{UC}}$ and the number of free steps in $\texttt{DEC}_\tau$ with the assumption on τ described in Theorem 3. If the number of free steps is strictly greater than $2\mu(k, r)n$ with high probability, we know that they are strictly $2\mu(k, r)$-free, and the results follow immediately by applying Theorem 1. Similarly, to obtain the same results for other τ-decimation algorithms, all we need to do is to calculate the number of free steps for those algorithms. If they are strictly $2\mu(k, r)$-free, we can obtain the same results by applying Theorem 1. Note that, due to certain limitations of our calculation, our results are limited to $k \geq 9$ in Theorem 2 and $k \geq 13$ in Theorem 3. We believe that the results hold for general $k \geq 3$, and can be proved by improving some subtle calculation in our argument.

Although we only show a few of implementations of the sequential local algorithms, we believe that the results are general across many sequential local algorithms with different local rules due to Theorem 3. In the framework of sequential local algorithms, the role of the local rules is to approximate the marginal probabilities of the selected variables over a random solution for the instance induced by the local neighborhood centered at the selected variables. Therefore, we believe that for any local rule that can make a good approximation on the marginals, it shall give similar results as Theorem 3. (Note that a more general definition of "δ-free" may be useful, for example, we can say a τ-decimation algorithm is (δ, ϵ)-*free* if we have $|p - 1/2| < \epsilon$, where p is the value returned by the local rule, for at least δn iterations.)

It is worth to mention the differences between these implementations of the sequential local algorithms and their well-known variants. Firstly, the UC-decimation algorithm is slightly different from the well-known Unit Clause algorithm. Under the framework of sequential local algorithm, the variables are selected in a random order. However, in the well-studied Unit Clause algorithm the variables in unit clauses are selected first [5]. The difference in the variable order could be crucial to the effectiveness of the Unit Clause algorithm. Secondly, the BP-decimation algorithm and the SP-decimation algorithm are slightly different from the Belief Propagation Guided Decimation Algorithm [12] and Survey Propagation Decimation Algorithm [8, 7, 28]. In the framework of sequential local algorithms, we only provide the local neighborhood to BP and SP. It is equivalent to bounding the number of messages passed in each decimation step by the constant $R \geq 0$ in BP-guided Decimation Algorithm and SP-guided Decimation Algorithm. It is in contrast to many empirical studies of BP-guided Decimation Algorithm and SP-guided Decimation Algorithm which allow the message passing iteration to continue until it converges.

Moreover, this work provides a new variant of the *overlap gap property method*, which was originally introduced by Gamarnik and Sudan [20]. Instead of considering the overlap gap property of the whole instance, we utilize that property of a sub-instance of the random k-XORSAT instance. In particular, the proof of this work is inspired by [20], which uses the overlap gap property method to rule out the class of balanced sequential local algorithms

from being able to solve random NAE-k-SAT problem when the clause density is close to the satisfiability threshold. Instead of directly applying the method on the whole instance, we focus on the sub-instance induced by the 2-core of the factor graph of the instance. This modification help us obtain tight bounds of algorithmic threshold unlike [20]. If we apply the original overlap gap property method and use the first moment method to obtain the property, we are able to show that the sequential local algorithms fail to solve the random k-XORSAT problem when the clause density is lower than a certain threshold $r_1(k)$. However, that threshold $r_1(k)$ is much higher than the $r_{core}(k)$. It only tells us that the algorithms fails when the density is very close to the satisfiability threshold $r_{sat}(k)$. With our modification, we can lower that threshold to exactly $r_{core}(k)$, namely, the algorithms fail in finding a solution when the clause density is as low as the clustering threshold. This opens a new possibility to improve other results which use the overlap gap property method on other random constraint satisfaction problems.

1.3 Phase transition of random k-XORSAT

Many random ensembles of constraint satisfaction problems CSPs such as random k-SAT and random NAE-k-SAT are closely related to the random k-XORSAT. For example, the well-known random k-SAT is analogous to the random k-XORSAT, in the sense that we can obtain a k-XORSAT instance from a k-SAT instance by replacing OR operators with XOR operators. We are particularly interested in the existences of some sharp thresholds on the clause density r in which the behaviors of a random instance changes sharply when the clause density r grows and passes through those thresholds. The following three thresholds are particularly of interest.

1. The *satisfiability threshold* separates the regime where w.h.p. the random instance is satisfiable and the regime where w.h.p. it is unsatisfiable.
2. The *clustering threshold* separates the regime where w.h.p. the solution space can be partitioned into well-separated subsets, each containing an exponential small fraction of solutions, and the regime where w.h.p. the solution space cannot be partitioned in this way.
3. The *algorithmic threshold* separates the regime where we have an efficient algorithm that can find a solution for a satisfiable random instance with non-vanishing probability, and the regime where no such algorithm exists.

Many random constraint satisfaction problems such as random k-SAT, random NAE-k-SAT and random graph coloring share the following phenomena related to these thresholds [3].

- There is an upper bound of the (conjectured) satisfiability threshold.
- There is a lower bound of the (conjectured) satisfiability threshold, from the non-constructive proof, and the lower bound is essentially tight.
- There are some polynomial time algorithms that can find a solution when the density is relatively low, but no polynomial time algorithm is known to succeed when the density is close to the satisfiability threshold. This leads to a conjectured algorithmic threshold, which is asymptotically below the (conjectured) satisfiability threshold.
- The clustering phenomenon takes place when the density is greater than a (conjectured) clustering threshold, and this threshold is close to or even asymptotically equal to the algorithmic threshold.

It is worth to mention that not every random constraint satisfaction problems share this set of phenomena. The most notable example is *symmetric binary perceptron* (SBP). Its satisfiability threshold $\alpha_{sat}(k) > 0$ was established by [33, 2]. They also showed that SBP exhibits clustering property and almost all clusters are singletons, for clause density $\alpha > 0$.

On the other hand, [1] gave a polynomial-time algorithm that can find solutions w.h.p., for low clause density. Therefore, there is a regime of low clause density in which SBP exhibits clustering property, and it is solvable in polynomial time, simultaneously. Its clustering phenomenon does not cause the hardness.

Being analogous to the random k-SAT problem, the random k-XORSAT problem shares those phenomena with other random constraint satisfaction problems. However, the story is slightly different in the random k-XORSAT problem. Since the k-XORSAT instances are equivalent to Boolean linear systems, their solution spaces are simply some affine subspaces in the Hamming hypercube $\{0,1\}^n$. Because of their algebraic structures, we are able to obtain the existence and the (n-independent) value of the satisfiability threshold of the random k-XORSAT problem. Dubois and Mandler [14] proved that there exists an n-independent satisfiability threshold $r_{sat}(k)$ for $k = 3$, and determined the exact value of the sharp threshold by the second moment argument. Pittel and Sorkin [34] further extended it for general $k \geq 3$. An independent work on cuckoo hashing from [13] also included an argument of the k-XORSAT satisfiability threshold for general $k \geq 3$. Those proofs consider the 2-core of the hypergraph associated with the random k-XORSAT instance. (In graph theory, a *k-core* of a (hyper)graph is a maximal subgraph in which all vertices have degree at least k.) Based on the 2-core, we can construct a sub-instance, called *2-core instance* or simply *core instance* of the original instance. One can prove that the original instance is satisfiable if and only if core instance is satisfiable. [10, 30] studied the core instance and determined the satisfiability threshold of the core instance, which can be converted to the satisfiability threshold of the random k-XORSAT instance.

Mézard, Ricci-Tersenghi and Zecchina [30] started the study of the clustering phenomenon of random k-XORSAT, and linked it to the existence of the non-empty 2-core instance. From [35, 32, 26], we know the non-empty 2-core of random hypergraphs suddenly emerges at a critical edge density $r_{core}(k)$. After that, Ibrahimi, Kanoria, Kraning and Montanari [25], and Achlioptas and Molloy [6] independently proved that there exists the clustering threshold $r_{clt}(k)$, which is equal to $r_{core}(k)$ and smaller than $r_{sat}(k)$, such that w.h.p. the solution space is a connected component for density $r < r_{core}(k)$, and w.h.p. the solution space shatters into exponentially many $\Theta(n)$-separated components for density $r > r_{core}(k)$, provided that we consider the solution space as a graph in which we add an edge between two solutions if their Hamming distance is $O(\log n)$.

As we mentioned before, the random k-XORSAT instance can be written as a random Boolean linear system, so it can be solved in polynomial time by using linear algebra method, regardless the clause density. For example, Gaussian elimination can solve it in $O(n^3)$ time. Since we do not have linear time algebraic method to solve linear system, we can still study the algorithmic phase transition if we only consider linear-time algorithms as efficient algorithms. In the proofs in [25], they provided an algorithm that can find a solution in linear time when $r < r_{core}(k)$, which implies that $r_{core}(k)$ is a lower bound of the (linear-time version of) algorithmic threshold $r_{alg}(k)$ of the random k-XORSAT problem. We conjecture that no algorithm can solve the random k-XORSAT problem in linear time with non-vanishing probability when $r > r_{core}(k)$, which implies $r_{core}(k)$ is an upper bound of $r_{alg}(k)$ and thus $r_{alg}(k) = r_{core}(k)$. This would lead to the intimate relation between the failure of linear time algorithms on random k-XORSAT and the clustering phenomenon of its solution space.

1.4 Sequential local algorithms

Sequential local algorithms are a class of algorithms parametrized by a local rule τ that specifies how values should be assigned to variables based on the "neighborhoods" of the variables. Given a local rule τ, the sequential local algorithm can be written as the following *τ-decimation algorithm*.

Given a fixed even number $R \geq 0$, we denote by $\mathcal{I}_R$ the set of all instances in which each of those instances has exactly one of its variables selected as *root*, and all nodes in its factor graph have distances from the root variable node at most R. A **local rule** is defined to be a function $\tau : \mathcal{I}_R \to [0,1] \in \mathbb{R}$, mapping from $\mathcal{I}_R$ to the interval $[0,1]$. Given an instance Φ, since the local neighborhood $B_\Phi(x^*, R)$ of a variable node x^* represents a sub-instance of Φ induced by all nodes having distance at most R from the root variable node x^*, we have $B_\Phi(x^*, R) \in \mathcal{I}_R$ and $\tau(B_\Phi(x^*, R))$ is well-defined. Then, the τ-decimation algorithm can be expressed as the followings.

Algorithm 1 τ-decimation algorithm.

Input: an instance of the k-XORSAT problem Φ,
an even number $R \geq 0$, and
a local rule $\tau : \mathcal{I}_R \to [0,1]$.
Set $\Phi_0 = \Phi$.
for $t = 0, ..., n-1$ **do**
  Select an unassigned variable x^* from Φ_t, uniformly at random.
  Set $\sigma(x^*) = \begin{cases} 1 & \text{with probability } \tau(B_{\Phi_t}(x^*, R)) \\ 0 & \text{with probability } 1 - \tau(B_{\Phi_t}(x^*, R)) \end{cases}$
  Obtain Φ_{t+1} from Φ_t by:
  (i) remove x^*;
  (ii) for any clause having x^* before (i), add $\sigma(x^*)$ to its right-hand-side value;
  (iii) remove all clauses that no longer contain any variable.
end for
Output: the assignment σ.

For any $t \in [n]$, if the value $\tau(B_{\Phi_t}(x^*, R))$ given by the local rule τ in the t-th iteration is $1/2$, then we call that iteration a **free step**. In a free step, the τ-decimation algorithm simply assigns a uniformly random Boolean value to the selected variable. On the contrary, if the value $\tau(B_{\Phi_t}(x^*, R))$ given by the local rule τ in the t-th iteration is either 0 or 1, then we call that iteration a **forced step**. In a forced step, the τ-decimation algorithm is forced to assign a particular value to the selected variable according to the value $\tau(B_{\Phi_t}(x^*, R))$. To simplify our discussion, we introduce the following definitions for those τ-decimation algorithms having certain numbers of free steps.

▶ **Definition 4.** *For any $\delta \in [0,1]$, we say a τ-decimation algorithm $\texttt{DEC}_\tau$ is **δ-free** on the random k-XORSAT instance $\mathbf{\Phi} \sim \mathbf{\Phi}_k(n, rn)$ if w.h.p the τ-decimation algorithm $\texttt{DEC}_\tau$ on input $\mathbf{\Phi}$ has at least δn free steps.*

▶ **Definition 5.** *For any $\delta \in [0,1]$, we say a τ-decimation algorithm $\texttt{DEC}_\tau$ is **strictly δ-free** on the random k-XORSAT instance $\mathbf{\Phi} \sim \mathbf{\Phi}_k(n, rn)$ if there exists $\delta' > \delta$ such that the τ-decimation algorithm $\texttt{DEC}_\tau$ is δ'-free on $\mathbf{\Phi}$.*

There are many choices for the local rules τ. The simplest one is the Unit Clause Propagation `UC`. In each iteration, after selecting the unassigned variable x^*, `UC` checks whether there exists a unit clause (clause with one variable) on the variable x^*. If yes, then `UC` sets $\tau(B_{\Phi_t}(x^*, R))$ to be the right-hand-side value of the unit clause, which can force the decimation algorithm to pick the suitable value to satisfy that clause. In this case, this iteration is a forced step. (If there are multiple unit clauses on the selected variable x^*, then only consider the one with the lowest index.) If there is no unit clause on the selected variable x^*, then `UC` sets $\tau(B_{\Phi_t}(x^*, R))$ to $1/2$, which let the algorithm choose the assigned value randomly. In this case, this iteration is a free step.

Algorithm 2 Unit Clause Propagation `UC`.

Input: the selected variable x^*, and its local neighborhood $B_{\Phi_t}(x^*, 2)$
if there exists any unit clause on the variable x^* **then**
    Pick the unit clause c on the variable x^* (with the lowest index if having multiple such clauses).
    Output: the right-hand-side value of the clause c.
else
    Output: $1/2$.
end if

1.5 Message passing algorithms

A new challenger to break the algorithmic threshold came out from statistical mechanics. In experiments [29, 21, 7, 36, 27], the *message passing algorithms* demonstrated their high efficiency on finding solutions of random k-SAT problem with the densities close to the satisfiability threshold. Those algorithms include *Belief Propagation Guided Decimation Algorithm* and *Survey Propagation Guided Decimation Algorithm*, which are based on the insightful but non-rigorous *cavity method* from statistical mechanics [7, 36]. Unfortunately, several analyses showed that they do not outperform the best known algorithms for some problems. Coja-Oghlan [12] showed that BP-guided Decimation fails to find solutions for random k-SAT w.h.p. for density above $\rho_0 2^k/k$ for a universal constant $\rho_0 > 0$, and thus does not outperform the best known algorithm from [11]. Hetterich [23] also gave the same conclusion for SP-guided Decimation by showing that it fails w.h.p. for density above $(1 + o_k(1))2^k \ln k/k$.

For random NAE-k-SAT, Gamarnik and Sudan [20] showed that the balanced sequential local algorithms fail to find solutions for density above $(1 + o_k(1))2^{k-1} \ln^2 k/k$ for sufficiently large k. This means the algorithms do not outperform the best known algorithm, Unit Clause algorithm, which can find solutions w.h.p. for density up to $\rho\, 2^{k-1}/k$ for some universal constant $\rho > 0$ for sufficiently large k [5]. The framework of balanced sequential local algorithms also covers BP-guided Decimation and SP-guided Decimation with the number of message passing iterations is bounded by $O((\ln \ln n)^{O(1)})$.

In our work, we obtain an analogous result. In Theorem 1, we show that w.h.p. strictly $2\mu(k, r)$-free sequential local algorithms fails to solve the random k-XORSAT problem when the clause density exceeds the clustering threshold. Then, in Theorem 3, we show that any sequential local algorithm with local rule that can compute the exact marginals are strictly $2\mu(k, r)$-free and thus fails to find a solution for random k-XORSAT problem. This theorem covers the sequential local algorithms with Belief Propagation `BP` and Survey Propagation `SP` as local rules.

1.6 Technique

The works from [3, 4] demonstrated the clustering phenomenon for several random CSPs, and conjectured that it could be an obstruction of solving those problems. [19, 20] and subsequent works leveraged a different notion of clustering, named *overlap gap property* (OGP) by [18], to link the clustering phenomenon to the hardness rigorously. Gamarnik gave a detailed survey on it [15].

This paper focuses on the vanilla version of the OGP. Given an instance Φ of the constraint satisfaction problem, we say it exhibits the overlap gap property with values $0 \le v_1 < v_2$ if every two solutions σ and σ' satisfy either $d(\sigma, \sigma') \le v_1$ or $d(\sigma, \sigma') \ge v_2$, where d is a

metric on its solution space. (We assume d is the Hamming distance throughout this paper.) Intuitively, it means every pair of solutions are either close to each other, or far from each other, and thus the solution space of the instance exhibits a topological discontinuity based on the proximity.

Now we illustrate how does the overlap gap property method. (Some details of the overlap gap property method is slightly different if we consider different variants of the OGP, but the overall idea of *topological barrier* stays the same.) Assume we have an algorithm $\mathcal{A}$ that takes a random instance $\mathbf{\Phi}$ as input and outputs an assignment σ for the instance. The output σ can be viewed as a random variable which depends on both the random instance $\mathbf{\Phi}$ and some internal random variables, represented by a random internal vector $\mathbf{I}$, of the algorithm. Let Φ and I be realizations of $\mathbf{\Phi}$ and $\mathbf{I}$ respectively, and denote the output assignment as $\sigma_0 = \sigma_{\Phi,I}$. We then re-randomize the components of the internal vector I *one-by-one*. After each re-randomizing a component of I, we run the algorithm again to generate a new output assignment. Then, we obtain a sequence of assignments $\sigma_0, \sigma_1, \cdots, \sigma_T$ for the instance Φ, where T is the number of components of the random vector I that we have re-randomized. Next, we show that the algorithm is *insensitive to its input* in the sense that when one of the components of the random internal vector $\mathbf{I}$ is re-randomized, the output assignment almost remains unchanged, In particular, $d(\sigma_i, \sigma_{i+1})$ is smaller than $v_2 - v_1$. We also show that the algorithm has certain *freeness* in the sense that when all components in the random internal are re-randomized the output assignment is expected to change a lot. In particular, $\mathbb{E}[d(\sigma_0, \sigma_T)]$ should be larger than v_2. These two properties together imply that the sequence of assignments cannot "jump" over the overlap gap, while two ends of the sequence probably lie in different clusters. Therefore, there should be an assignment σ_{T_0} that falls in the gap, namely, there exists $T_0 > 0$ such that $v_1 \leq d(\sigma_0, \sigma_{T_0}) \leq v_2$. If the probability that the algorithm successfully finds a solution is greater than some small value s_n slowly converging to 0, then there could be a very small probability that both σ_0 and σ_{T_0} are solutions of the instance Φ, with $v_1 \leq d(\sigma_0, \sigma_{T_0}) \leq v_2$. Even though this probability is very small, it still has the chance to violates the OGP of the instance. Then, by contradiction, we could conclude that the probability that the algorithm succeeds in finding a solution is smaller than $s_n = o(1)$, namely, w.h.p. the algorithm fails in finding a solution.

Instead of considering the overlap gap property of the entire instance $\mathbf{\Phi}$, we move our focus to the overlap gap property of a sub-instance of $\mathbf{\Phi}$. Indeed, the sub-instance we consider is the *2-core instance* $\mathbf{\Phi_c}$ induced by the 2-core of the factor graph representation of the random instance $\mathbf{\Phi}$. In [25, 6], they proved that the 2-core instance $\mathbf{\Phi_c}$ exhibits the overlap gap property with $v_1' = o(n)$ and $v_2' = \epsilon_k n$ for some constant $\epsilon_k > 0$ for clause density $r_{core}(k) < r < r_{sat}(k)$. We remove all variables not in the core instance from the sequence of assignments $\sigma_0, \sigma_1, \cdots, \sigma_T$ we obtained above, then it becomes a sequence of assignments $\sigma_0', \sigma_1', \cdots, \sigma_T'$ for the core instance. We also prove that the algorithm is *insensitive to its input* with respect to the core instance in the sense that $d(\sigma_i', \sigma_{i+1}') < v_2' - v_1'$, and has certain *freeness* so that $\mathbb{E}[d(\sigma_0', \sigma_T')] > v_2'$. By repeating the above argument of the overlap gap property method, we can conclude that w.h.p. the algorithm fails in find a solution.

Our proof can also be used for the non-sequential local algorithms. Since the local rule τ runs on the local neighborhood of each variable in parallel, the values assigned to variables do not depend on each other. Informally speaking, there is no long-range dependency among those assigned values. Therefore, re-randomizing one component of the internal vector I, say I_{i+1}, only affects the value $\sigma(x_{i+1})$ assigned to the corresponding variable x_{i+1}. So, we have $d(\sigma_i, \sigma_{i+1}) \leq 1 < v_2' - v_1'$. Hence, we can obtain the same result as Theorem 1 for non-sequential local algorithms with the same proof.

1.7 Related works

The vanilla version of OGP helps us rule out some large classes of algorithms on random CSPs for relatively high clause densities, but it is not sophisticated enough to close the *statistical-to-computational gap* in some cases such as random NAE-k-SAT discussed in [20] and random k-XORSAT discussed in this paper (more details in Section 2). There have been some recent works trying to improve the notion of OGP by developing different variants of OGP. The most notable one is *multi-OGP* [37, 9, 24, 17, 16], which succeeds in closing the statistical-to-computational gap in certain models. However, it is not clear about the relation between the clustering property and the multi-OGP.

2 Overlap Gap Property

We say that a k-XORSAT instance Φ exhibits the **overlap gap property** (or shortened as OGP) with the values $0 \le v_1 < v_2$ if for any two solutions $\sigma, \sigma' \in S(\Phi)$ we have either $d(\sigma,\sigma') \le v_1$ or $d(\sigma,\sigma') \ge v_2$. Informal speaking, any two solutions of the instance are either close to each other or far away from each other, and thus the solution space exhibits a topological discontinuity. Given a random k-XORSAT instance, we can prove that it exhibits the OGP when the clause density is greater than certain value, and obtain the following lemma.

▶ **Lemma 6.** *For any $k \ge 3$, there exists $r_1(k) > 0$ such that for $r > r_1(k)$ and any pair of solutions $\sigma, \sigma' \in S(\mathbf{\Phi})$ of the random k-XORSAT instance $\mathbf{\Phi} \sim \mathbf{\Phi}_n(k, rn)$, w.h.p. the distance $d(\sigma,\sigma')$ between the two solutions is either $\le u_1 n$ or $\ge u_2 n$ for some $0 \le u_1 < u_2$. In particular, the value of $r_1(k)$ is given by*

$$r_1(k) = \min_{0\le\alpha\le 1} \frac{1 + H(\alpha)}{2 - \log(1 + (1-2\alpha)^k)},$$

where H is the binary entropy function, that is, $H(x) = -x\log_2(x) - (1-x)\log_2(1-x)$.

Instead of considering the random k-XORSAT instance $\mathbf{\Phi}$ itself, we focus on the sub-instance, called the *core instance* (defined below), of the random k-XORSAT instance $\mathbf{\Phi}$, and show that core instance also exhibits the overlap gap property, even when the clause density is much lower. (See Table 1.)

Table 1 Compare $r_{core}(k)$ with $r_1(k)$ for different k. The numeric values in the table are rounded off to 6 decimal places.

k	3	4	5	6	7	8	9
$r_{core}(k)$	0.818470	0.772280	0.701780	0.637081	0.581775	0.534997	0.495255
$r_1(k)$	0.984516	0.943723	0.905812	0.874349	0.848314	0.826470	0.807862

We start from defining the **peeling algorithm** and the **core instances**. Given an XORSAT instance Φ, suppose there exists a variable x of degree 1, which means it is involved in exactly one clause e. We remove the variable x and the only clause c involving x, to obtain a modified instance Φ'. If we have a solution σ' for the modified instance Φ', we can always choose a suitable value for the variable x to satisfy the clause c, and extend the solution σ' to a solution σ for the original instance Φ. Similarly, we can also do the same thing if the variable x is of degree 0, since it does not involve in any equation, and we are free to choose any value for it. By doing this, solving the original instance Φ is reduced to solving the modified instance Φ'.

We can repeat this process until there is no variable of degree at most 1. This process is named the **peeling algorithm** on an instance as its input (Algorithm 3). We call the resultant instance the **2-core instance** (or simply the **core instance**) of the instance Φ, denoted by Φ_c. This name is borrowed from the graph theory. In graph theory, the *k-core* of a graph is the maximal subgraph with minimum degree at least k. It is known that the factor graph of the core instance Φ_c is exactly the maximal subgraph of the factor graph G_Φ of the instance Φ, with minimum variable degree at least 2.

Algorithm 3 Peeling algorithm.

Input: an instance Φ.
while There exists ≥ 1 variable of degree ≤ 1. **do**
    Select a variable x of degree ≤ 1.
    (Pick x with the lowest index if there are > 1 such variables.)
    Update Φ by removing the variable x_i and its only involved clause (if exists).
end while
Output: the resultant instance Φ.

Mézard and Montanari [31] gave a detailed description on the structure of the core instance. Reader can find more details about the core instance in their book. The following theorem is a short summary of some known facts about the core instances we needed in the paper.

▶ **Theorem 7.** *For any $k \geq 3$, there exists $r_{core}(k) > 0$ given by*

$$r_{core}(k) \equiv \sup\{r \in [0,1] : Q > 1 - e^{-krQ^{k-1}} \ \forall Q \in (0,1)\}$$

such that the factor graph G_c of the core instance $\mathbf{\Phi}_c$ of the random k-XORSAT instance $\mathbf{\Phi} \sim \mathbf{\Phi}_n(k, rn)$ have the following properties.

1. *For $r < r_{core}(k)$, w.h.p. the factor graph G_c of the core instance $\mathbf{\Phi}_c$ is an empty graph.*
2. *For $r > r_{core}(k)$, w.h.p. the factor graph G_c of the core instance $\mathbf{\Phi}_c$ have $V(k,r)n + o(n)$ variable nodes, where*

$$V(k,r) = 1 - \exp(-krQ^{k-1}) - krQ^{k-1}\exp(-krQ^{k-1})$$

and Q is the largest solution of the fixed point equation $Q = 1 - \exp(-krQ^{k-1})$. In particular, the fraction of variable nodes of degree l is between $\widehat{\Lambda}_l - \epsilon$ and $\widehat{\Lambda}_l + \epsilon$ with probability greater than $1 - e^{-\Theta(n)}$, where $\widehat{\Lambda}_0 = \widehat{\Lambda}_1 = 0$ and

$$\widehat{\Lambda}_l = \frac{1}{e^{krQ^{k-1}} - 1 - krQ^{k-1}} \frac{1}{l!} (krQ^{k-1})^l \quad \text{for } l \geq 2.$$

3. *Conditioning on the number of variable nodes $V(k,r)n + o(n)$ and the degree profile $\widehat{\Lambda}$, the factor graph G_c of the core instance $\mathbf{\Phi}_c$ is distributed according to the ensemble containing all possible factor graphs of k-XORSAT instances of $V(k,r)n + o(n)$ variables and variable degree distribution Λ.*

Theorem 7 shows that there exists a threshold $r_{core}(k)$ below the satisfiability threshold $r_{sat}(k)$ of random k-XORSAT problem. When the clause density r is below the threshold $r_{core}(k)$, w.h.p. the instance $\mathbf{\Phi}$ does not have a core instance. When the clause density r is above the threshold $r_{core}(k)$, w.h.p. the core instance $\mathbf{\Phi}_c$ emerges. In particular, the variable degree distribution is a Poisson distribution with mean krQ^{k-1} conditioning on $\Lambda_0 = \Lambda_1 = 0$. [31] also showed that the core instance exhibits the OGP.

▶ **Lemma 8.** *For $k \geq 3$ and $r_{core}(k) < r < r_{sat}(k)$, there exists $\epsilon(k,r) > 0$ such that w.h.p. the distance between any two solutions for the core instance $\mathbf{\Phi}_c$ of a random k-XORSAT instance $\mathbf{\Phi} \sim \mathbf{\Phi}_n(k, rn)$ is either $o(n)$ or greater than $\epsilon(k,r)n$.*

Now, we know that w.h.p. the core instance of a random k-XORSAT instance has the overlap gap property with the values $v_1 = o(n)$ and $v_2 = \epsilon(k,r)n$. With OGP, we can partition the solution space of the core instance into multiple groups, each called a **core cluster**, such that the distance between any pair of core solutions in the same core cluster is at most $o(n)$, and the distance between any pair of core solutions in different core clusters is at least $\epsilon(k,r)n$.

Suppose we have a k-XORSAT instance Φ. We first define a binary relation on the solution space of a core instance Φ_c by: for $\sigma_c, \sigma_c' \in \mathcal{S}(\Phi_c)$, we write $\sigma_c \simeq \sigma_c'$ if and only if $d(\sigma_c, \sigma_c') = o(n)$. It is easy to see it is an equivalence relation. Then, we can partition the solution space by the equivalence classes of $\simeq$. We can denote those equivalence classes by $\mathcal{S}_{c,1}, \mathcal{S}_{c,2}, ..., \mathcal{S}_{c,n_c}$. Thus, we have $\mathcal{S}_{c,1} \sqcup \mathcal{S}_{c,2} \sqcup ... \sqcup \mathcal{S}_{c,n_c} = \mathcal{S}(\Phi_c)$ where $\sqcup$ is the disjoint union. Then, we have

$$\begin{aligned} &d(\sigma_c, \sigma_c') = o(n) && \text{if } \ \sigma_c, \sigma_c' \in \mathcal{S}_{c,i}, \quad \text{and} \\ &d(\sigma_c, \sigma_c') \geq \epsilon(k,r)n && \text{if } \ \sigma_c \in \mathcal{S}_{c,i},\ \sigma_c' \in \mathcal{S}_{c,j} \text{ and } \mathcal{S}_{c,i} \neq \mathcal{S}_{c,j} \end{aligned}$$

Now we can partition the solution space $\mathcal{S}(\Phi)$ of the original instance Φ into clusters based on the partition of the solution space of core instance. We set

$$\mathcal{S}(\Phi) = \bigsqcup_{i=1}^{n_c} \mathcal{S}_i \quad \text{and} \quad \mathcal{S}_i = \{\sigma \in \mathcal{S}(\Phi) : \pi(\sigma) \in \mathcal{S}_{c,i}\} \quad \text{for } i = 1, 2, ..., n_c$$

where π is defined to be the **projection** mapping assignments for the instance Φ to assignments for the core instance Φ_c by removing all variables not in the core instance Φ_c. Each $\mathcal{S}_i$ is called a **cluster** in the solutions space $\mathcal{S}(\Phi)$. We can then prove that these clusters are well-separated from each other.

▶ **Lemma 9.** *Let $k \geq 3$ and $r_{core}(k) < r < r_{sat}(k)$. Suppose $\mathbf{\Phi} \sim \mathbf{\Phi}_n(k, rn)$ is a random k-XORSAT instance. Then, w.h.p. there exists a partition $\mathcal{S}(\mathbf{\Phi}) = \mathcal{S}_1 \sqcup \mathcal{S}_1 \sqcup ... \sqcup \mathcal{S}_{n_c}$ for the solutions space $S(\mathbf{\Phi})$ of the random instance $\mathbf{\Phi}$ such that the following statements hold.*

1. *If $\sigma, \sigma' \in \mathcal{S}_i$ for some $i \in [n_c]$, then we have $d(\sigma, \sigma') \leq \mu(k,r)n + o(n)$, where the real-valued function $\mu(k,r)$ is given by $\mu(k,r) = \exp(-krQ^{k-1}) + krQ^{k-1}\exp(-krQ^{k-1})$ and Q is the largest solution of the fixed point equation $Q = 1 - \exp(-krQ^{k-1})$.*
2. *If $\sigma \in \mathcal{S}_i, \sigma' \in \mathcal{S}_j$ and $\mathcal{S}_i \neq \mathcal{S}_j$ for some $i, j \in [n_c]$, then we have $d(\sigma, \sigma') \geq \epsilon(k,r)n$.*

Proof. Assume the instance $\mathbf{\Phi}$ has a non-empty core instance $\mathbf{\Phi}_c$, which exists with high probability according to Theorem 7. We also assume the core instance $\mathbf{\Phi}_c$ exhibits the OGP with $v_1 = o(n)$ and $v_2 = \epsilon(k,r)n$, which occurs with high probability according to Lemma 8. Let σ and σ' be two solutions of the random k-XORSAT instance $\mathbf{\Phi}$, and let $\sigma_c = \pi(\sigma)$ and $\sigma_c' = \pi(\sigma')$ be the projection of σ and σ' on the core solution space $\mathcal{S}(\mathbf{\Phi})$, respectively.

To prove the first part of the lemma, we assume that σ and σ' are in the same cluster, that is, $\sigma, \sigma' \in \mathcal{S}_i$ for some $i \in [n_c]$. By the definition of cluster, we have $d(\sigma_c, \sigma_c') = o(n)$. Therefore, $d(\sigma, \tau)$ is upper bounded by the number of variables not in the core instance, plus $o(n)$. By Theorem 7, the number of variables outside the core instance is given by $(1 - V(k,r))n + o(n)$. Hence, we have $d(\sigma, \tau) \leq (1 - V(k,r))n + o(n) = \left(\exp(-krQ^{k-1}) + krQ^{k-1}\exp(-krQ^{k-1})\right)n + o(n)$.

To prove the second part of the lemma, we assume that σ and τ are in the different clusters, that is, $\sigma \in \mathcal{S}_i$, $\sigma' \in \mathcal{S}_j$ and $\mathcal{S}_i \neq \mathcal{S}_j$ for some $i, j \in [n_c]$. By the definition of cluster and Lemma 8, we have $d(\sigma_c, \sigma_c') \geq \epsilon(k,r)n$ Therefore, we have $d(\sigma, \sigma') \geq d(\pi(\sigma), \pi(\sigma')) = d(\sigma_c, \sigma_c') \geq \epsilon(k,r)n$. ◀

3 Preparation of OGP method

In this section, we introduce some notions and obtain some preliminary results needed by the overlap gap property method to prove the main results.

3.1 Sequence of output assignments

The random k-XORSAT instance $\mathbf{\Phi}$ is a random variable, and the τ-decimation algorithm $\mathtt{DEC}_\tau$ is a randomized algorithm. Therefore, the assignment output by the τ-decimation algorithm $\mathtt{DEC}_\tau$ on input $\mathbf{\Phi}$ is also a random variable. The outcomes of the output assignment depend on the random instance $\mathbf{\Phi}$, the order of variables being chosen, and the value selection based on the output from the local rule τ. Now we introduce two random variables to explicitly represent the order of variables and the value selection so that we can have a more concrete language to discuss how the randomness from both the instance and the algorithm affects the output assignment. We adopt the notation from [20] in the following discussion.

The order of variables can be represented by a random vector $\mathbf{Z} = (\mathbf{Z}_1, \mathbf{Z}_2, \cdots, \mathbf{Z}_n)$ whose entries are n i.i.d. random variables with uniform distribution over the interval $[0,1] \subset \mathbb{R}$, independent of the random instance $\mathbf{\Phi}$. We call $\mathbf{Z}$ the **ordering vector** of the algorithm. For all $i \in [n]$, the variable x_i in the instance $\mathbf{\Phi}$ is associated with the random variable $\mathbf{Z}_i$. In each iteration of the algorithm, the unassigned variable x_i with the largest value $\mathbf{Z}_i$, among all other unassigned variables, is selected. In the other words, we can construct the permutation $s : [n] \to [n]$ such that $\mathbf{Z}_{s(1)} > \mathbf{Z}_{s(2)} > \cdots > \mathbf{Z}_{s(n)}$, and for all $t \in [n]$ the variable $x_{s(t)}$ is selected in the t-th iteration. The value selection based the output from the local rule τ can be represented by a random vector $\mathbf{U} = (\mathbf{U}_1, \mathbf{U}_2, ..., \mathbf{U}_n)$ whose entries are n i.i.d. random variables with uniform distribution over the interval $[0,1] \subset \mathbb{R}$. We call $\mathbf{U}$ the **internal vector** of the algorithm. In the t-th iteration of the algorithm, the value $\sigma(x_{s(t)})$ assigned to the selected variable $x_{s(t)}$ is set to be 1 if $\mathbf{U}_t < \tau(B_{\mathbf{\Phi}_t}(x_{s(t)}, R))$, and 0 otherwise. Conditioning on $\mathbf{\Phi}$, $\mathbf{Z}$ and $\mathbf{U}$, the output assignment σ can be uniquely determined. Therefore, we can view the τ-decimation algorithm $\mathtt{DEC}_\tau$ as a deterministic algorithm on random input $(\mathbf{\Phi}, \mathbf{Z}, \mathbf{U})$, and denote by $\sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{U}}$ the output of the algorithm.

With this notion of the deterministic algorithm, we can construct a sequence of output assignments which will be used in the argument of the overlap gap property method. The sequence of output assignments is generated by applying the τ-decimation algorithm $\mathtt{DEC}_\tau$ on a random k-XORSAT instance $\mathbf{\Phi}$ multiple times in the following way: First, given a random k-XORSAT instance $\mathbf{\Phi}$, we sample an ordering vector $\mathbf{Z}$ and an internal vector $\mathbf{U}$. Then, we run the τ-decimation algorithm $\mathtt{DEC}_\tau$ on input $\mathbf{\Phi}$ with the ordering vector $\mathbf{Z}$ and the internal vector $\mathbf{U}$ to get the first output assignment σ_0. After that, we re-randomize (i.e. sample again) the entries of the internal vector $\mathbf{U}$ one by one from $\mathbf{U}_1$ to $\mathbf{U}_n$. Right after each re-randomization we run the algorithm again to get a new output assignment. By doing this, we obtain a sequence of $n+1$ output assignments for the instance $\mathbf{\Phi}$ in total. We denote by σ_i the output assignment generated after re-randomizing the first i entries of $\mathbf{U}$, for $i = 0, 1, 2, ..., n$. Precisely speaking, let $\mathbf{V} = (\mathbf{V}_1, \mathbf{V}_2, ..., \mathbf{V}_n)$ and $\mathbf{W} = (\mathbf{W}_1, \mathbf{W}_2, ..., \mathbf{W}_n)$ be two independent random internal vectors with the uniform distribution over $[0,1]^n$, and set $\mathbf{U^i} = (\mathbf{W}_1, ..., \mathbf{W}_i, \mathbf{V}_{i+1}, ..., \mathbf{V}_n)$ for $i = 0, 1, 2, ..., n$. Note that $\mathbf{U^0} = \mathbf{V}$ and $\mathbf{U^n} = \mathbf{W}$.

Then, the sequence of output assignments $\{\sigma_i\}_{i=0}^n$ can be written as $\{\sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{U^i}}\}_{i=0}^n$, which is equivalent to the sequence of output assignment obtained by running the τ-decimation algorithm $\mathtt{DEC}_\tau$ (for $n+1$ times in total) on input $(\mathbf{\Phi},\mathbf{Z},\mathbf{U^i})$ for all $i=0,1,...,n$.

Recall the projection π mapping assignments for the instance $\mathbf{\Phi}$ to assignments for the core instance $\mathbf{\Phi}_c$, by removing all variables not in the core instance. We can further obtain a sequence of assignments for the core instance $\mathbf{\Phi}_c$ by applying the projection on the output assignments σ_i, that is, we set $\{\sigma_i' = \pi(\sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{U^i}})\}_{i=0}^n$.

3.2 Insensitive to internal vector

In this section, we show that the τ-decimation algorithm $\mathtt{DEC}_\tau$ is *insensitive* to its internal vector. By *insensitive*, it means when the value of an entry in the internal vector $\mathbf{U}$ is changed, only a small portion of the assigned values in the output assignment $\sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{U}}$ change accordingly. If so, every two consecutive output assignments in the sequence $\{\sigma_i = \sigma_{\Phi,\mathbf{Z},\mathbf{U^i}}\}_{i=0}^n$ should only differ from each other in only a small portion of assigned values.

Consider the sequence of output assignment $\{\sigma_i = \sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{U^i}}\}_{i=0}^n$ from Section 3.1. Note that the i-th output assignment σ_i in the sequence is the output of the algorithm on input $(\mathbf{\Phi},\mathbf{Z},\mathbf{U^i})$. For any $i \in [n]$, the only difference between the input $(\mathbf{\Phi},\mathbf{Z},\mathbf{U^{i-1}})$ and the input $(\mathbf{\Phi},\mathbf{Z},\mathbf{U^i})$ is the i-th entries of the internal vectors $\mathbf{U^{i-1}}$ and $\mathbf{U^i}$. We can immediately see that the insensitivity of the algorithm implies that every two consecutive output assignments in the sequence are close to each other. Gamarnik and Sudan [20] proved the insensitivity of the τ-decimation algorithm in their works, using the notion of *influence range*. Although their works [20] focused on the random NAE-k-SAT problem, the proof for the insensitivity of the τ-decimation algorithm is independent of the type of clauses in the random constraint satisfaction framework. So, we can directly use the result here.

▶ **Definition 10.** *Given a random instance $\mathbf{\Phi}$ and a random ordering vector $\mathbf{Z}$, we say that x_i **influences** x_j if either $x_i = x_j$ or in the variable-to-variable graph of the instance $\mathbf{\Phi}$ there exists a sequence of variable nodes $y_0, y_1, ..., y_t \in \{x_1, x_2, ..., x_n\}$ such that the following statements hold.*

1. *$y_0 = x_i$ and $y_t = x_j$.*
2. *There exists a path from y_l to y_{l+1}, of length at most r, in the variable-to-variable graph G, for $l = 0, 1, ..., t-1$.*
3. *$\mathbf{Z}_{y_{l-1}} > \mathbf{Z}_{y_l}$ for $l = 1, 2, ..., t$. In particular, $\mathbf{Z}_{x_i} > \mathbf{Z}_{x_j}$.*

*We define the **influence range** of x_i to be the set of all variables x_j influenced by x_i, denoted by $\mathcal{IR}_{x_i}$.*

▶ **Lemma 11.** *Given an instance Φ, a vector $Z \in [0,1]^n$, and two vectors $U, U' \in [0,1]^n$, we assume there exists $i \in \{1,2,...,n\}$ such that $U_i \neq U_i'$ and $U_j = U_j'$ for all $j \neq i$. Then, $\sigma_{\Phi,Z,U}(x) = \sigma_{\Phi,Z,U'}(x)$ for all variables $x_j \notin \mathcal{IR}_{x_i}$.*

▶ **Lemma 12.** *For any $\xi \in (0,1)$ and sufficiently large n,*

$$\Pr\left[\max_{1\le i\le n} |\mathcal{IR}_{x_i}| \ge n^{1/6}\right] \le \exp\left(-\ln n (\ln\ln n)^{\xi/4}\right).$$

They first showed that changing the value of only one entry, say U_i, in the internal vector U only affects the values assigned to the variables in the influence range of the variable x_i (Lemma 11). They further showed that w.h.p. the size of the influence range of variables is sublinear for all variables (Lemma 12). Note that in the original statement of Lemma 12 in [20], the index 1/6 in the inequality above can be any real number between 0 and 1/5.

Here, we pick a fixed value $1/6$ for simplicity. Combining these two lemmas, we can show that w.h.p. the differences between $\sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{U^{i-1}}}$ and $\sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{U^{i}}}$ is upper bounded by $n^{1/6}$ for all $i \in [n]$.

▶ **Lemma 13.** *For any $\xi \in (0,1)$ and sufficiently large n,*

$$\Pr\left[\, d(\sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{U^{i-1}}}, \sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{U^{i}}}) \geq n^{1/6} \text{ for some } i \in [n] \,\right] \leq \exp\left(-\ln n (\ln\ln n)^{\xi/4}\right).$$

Proof. Fix an arbitrary $i \in [n]$. We know that $\mathbf{U_j^{i-1}} = \mathbf{U_j^{i}}$ for all $j \neq i$, and $\mathbf{U_i^{i-1}} \neq \mathbf{U_i^{i}}$. By Lemma 11, we have $\sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{U^{i-1}}}(x_j) = \sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{U^{i}}}(x_j)$ for all variables $x_j \notin \mathcal{IR}_{x_i}$. If $d(\sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{U^{i-1}}}, \sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{U^{i}}}) \geq n^{1/6}$ for some $i \in [n]$, we have $|\mathcal{IR}_{x_i}| \geq n^{1/6}$. Hence, by Lemma 12, the result follows. ◀

3.3 Freeness

Recall the definition of *free steps*. An iteration of the τ-decimation algorithm $\mathtt{DEC}_\tau$ is called a free step if the local rule τ gives the value $1/2$ in that iteration. In this case, the value chosen by the τ-decimation algorithm for the selected variable is either 0 or 1 with even probability. Intuitively, it means that the local rule τ cannot capture useful information from the local structure to guide the τ-decimation algorithm choosing value for the selected variable, and thus the τ-decimation algorithm simply make a random guess for the assigned value. We also recall the definition of a τ-decimation algorithm being δ-free. A τ-decimation algorithm $\mathtt{DEC}_\tau$ is δ-free on the random k-XORSAT instance $\mathbf{\Phi}$ if w.h.p. the algorithm has at least δn free steps, on input $\mathbf{\Phi}$. Informal speaking, the more free the τ-decimation algorithm, the less the information captured by the local rule.

By using the Wormald's method of differential equations, we can calculate the degree profile of the remaining factor graph after t steps of the τ-decimation algorithm, for all $0 \leq t \leq n$. With the degree profiles, we can calculate the probability of each step being free, and thus approximate how free the τ-decimation algorithm is. The probability of having free steps depends on the choice of the local rules. Lemma 14 shows the freeness of the UC-decimation algorithm.

▶ **Lemma 14.** *For $k \geq 3$ and $r > 0$, the UC-decimation algorithm $\mathtt{DEC}_{\mathtt{UC}}$ is $w_1(k,r)$-free on the random k-XORSAT instance $\mathbf{\Phi} \sim \mathbf{\Phi}_k(n, rn)$, where*

$$w_1(k,r) = \frac{(kr)^{\frac{1}{1-k}}}{k-1}\,\gamma\left(\frac{1}{k-1}, kr\right)$$

and γ is the lower incomplete gamma function given by $\gamma(a,x) \equiv \int_0^x t^{a-1}e^{-t}dt$.

The role of the local rules is to approximate the marginal probability of the selected variable over a randomly chosen solution for the sub-instance induced by the local neighborhood of the selected variable. Interestingly, even we have a local rule τ that is capable to give the exact marginals when the factor graph is a tree, it still cannot provide enough useful information to guide the τ-decimation algorithm making good decision for the assigned value. With such a local rule, the τ-decimation algorithm still has a certain level of freeness.

▶ **Lemma 15.** *Assume the local rule τ can give the exact marginal probabilities of variables on any factor graph that is a tree. For $k \geq 3$ and $r > 0$, the τ-decimation algorithm $\mathtt{DEC}_\tau$ is $w_e(k,r)$-free on the random k-XORSAT instance $\mathbf{\Phi} \sim \mathbf{\Phi}_k(n, rn)$, where*

$$w_e(k,r) = \int_0^1 S_R(x)dx,$$

$S_0(x) = 1$ and $S_l(x) = \exp\left(-kr[(1-x)(1-S_{l-1}(x)) + x]^{k-1}\right)$ for any $l \geq 1$ and $x \in \mathbb{R}$.

4 Proof of main theorems

We denote by α_n the *success probability* of the τ-decimation algorithm $\texttt{DEC}_\tau$, namely, α_n is the probability that the assignment output by the τ-decimation algorithm $\texttt{DEC}_\tau$ on the random k-XORSAT instance $\mathbf{\Phi} \sim \mathbf{\Phi}_k(n, rn)$ with n variables and rn clauses is a solution for $\mathbf{\Phi}$. Formally, we define α_n by the following expression $\alpha_n \equiv \Pr\left[\, \sigma_{\mathbf{\Phi}\sim\mathbf{\Phi}_k(n,rn),\mathbf{Z},\mathbf{U}} \in \mathcal{S}(\mathbf{\Phi}) \,\right]$, where $\mathbf{\Phi} \sim \mathbf{\Phi}_k(n, rn)$ is the random k-XORSAT instance, $\mathbf{Z}$ is the random ordering vector, and $\mathbf{U}$ is the random internal vector, as mentioned in Section 3.1. Now, we consider the sequence of output assignments $\{\sigma_i = \sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{U^i}}\}_{i=0}^n$ generated by the procedure in Section 3.1. We first prove that if the algorithm $\texttt{DEC}_\tau$ is δ-free, then the expected distance $\mathbb{E}\left[d(\sigma_0, \sigma_n)\right]$ between the first and the last assignments in the sequence is at least $(\delta/2)n + o(n)$.

▶ **Lemma 16.** *If the τ-decimation algorithm $\texttt{DEC}_\tau$ is δ-free on the random k-XORSAT instance $\mathbf{\Phi} \sim \mathbf{\Phi}_k(n, rn)$ for some $\delta > 0$, then we have $\mathbb{E}\left[d(\sigma_0, \sigma_n)\right] \geq (\delta/2)n + o(n)$.*

Next, we will show that, if the τ-decimation algorithm is "free enough", namely, strictly $2\mu(k, r)$-free, then we can pick a pair of output assignments and project them to the core instance $\mathbf{\Phi}_c$ so that the distance between the two corresponding core assignments falls in the *forbidden range* from the overlap gap property of the core instance $\mathbf{\Phi}_c$.

▶ **Lemma 17.** *For any $k \geq 3$ and $r \in (r_{core}(k), r_{sat}(k))$, if the τ-decimation algorithm $\texttt{DEC}_\tau$ is strictly $2\mu(k, r)$-free on the random k-XORSAT instance $\mathbf{\Phi} \sim \mathbf{\Phi}_k(n, rn)$, then there exist $0 \leq i_0 \leq n$ and $0 < \epsilon' < \epsilon(k, r)$ such that w.h.p. we have $\left|d(\pi(\sigma_0), \pi(\sigma_{i_0})) - \frac{1}{2}\epsilon' n\right| < \frac{1}{4}\epsilon' n$, where $\epsilon(k, r)$ is given in Lemma 8.*

The following lemma shows that the probability of both the output assignments $\sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{U^o}}$ and $\sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{U^{i_o}}}$ being solutions for the instance $\mathbf{\Phi}$ is lower bounded by α_n^2.

▶ **Lemma 18.** *For any $i \in [n]$, we have $\Pr\left[\, \sigma_0 \in \mathcal{S}(\Phi) \text{ and } \sigma_i \in \mathcal{S}(\Phi) \,\right] \geq \alpha_n^2$.*

Finally, we can combine all above lemmas in this section to give the proof of Theorem 1.

Proof of Theorem 1. We denote by $\mathcal{A}$ the event of $\left|d(\pi(\sigma_0), \pi(\sigma_{i_0})) - \frac{1}{2}\epsilon' n\right| < \frac{1}{4}\epsilon' n$, and we have $\Pr\left[\,\mathcal{A}\,\right] = 1 - o(1)$ by Lemma 17. On the other hand, we pick $i = i_0$ for the inequality in Lemma 18. We denote by $\mathcal{B}$ the event of $\sigma_0 \in \mathcal{S}(\Phi)$ and $\sigma_{i_0} \in \mathcal{S}(\Phi)$, and $\Pr\left[\,\mathcal{B}\,\right] \geq \alpha_n^2$. Note that we have $\Pr\left[\,\mathcal{A} \cap \mathcal{B}\,\right] \geq 1 - \Pr\left[\,\text{Not } \mathcal{A}\,\right] - \Pr\left[\,\text{Not } \mathcal{B}\,\right] \geq 1 - o(1) - (1 - \alpha_n^2) = \alpha_n^2 - o(1)$. Thus, we have $\alpha_n \leq \Pr\left[\,\mathcal{A} \cap \mathcal{B}\,\right]^{1/2} + o(1)$.

Now assume both $\mathcal{A}$ and $\mathcal{B}$ take places. Since both σ_0 and σ_{i_0} are solutions for the random instance $\mathbf{\Phi}$, both $\pi(\sigma_0)$ and $\pi(\sigma_{i_0})$ are solutions for the core instance $\mathbf{\Phi}_c$. Moreover, the distance $d(\pi(\sigma_0), \pi(\sigma_{i_0}))$ falls in the interval $((1/4)\epsilon' n, (3/4)\epsilon' n) \subsetneq (o(n), \epsilon n)$, which takes place with probability at most $o(1)$ by Lemma 8. So, we have $\Pr\left[\,\mathcal{A} \cap \mathcal{B}\,\right] \leq o(1)$, and thus $\alpha_n \leq o(1)$. ◀

To prove Theorem 2 and 3, all we need to do is to show that $\texttt{DEC}_{\texttt{UC}}$ and $\texttt{DEC}_\tau$ with the exact marginal assumption are strictly $2\mu(k, r)$-free. The results immediately follow by applying Theorem 1. From Lemma 14 and 15, we know that $\texttt{DEC}_{\texttt{UC}}$ and $\texttt{DEC}_\tau$ are $w_1(k, r)$-free and $w_e(k, r)$-free, respectively. So, we only need to show that $w_1(k, r) > 2\mu(k, r)$ and $w_e(k, r) > 2\mu(k, r)$. It can be done with the following lemmas, which give an upper bound of $\mu(k, r)$ in Lemma 19, a lower bound of $w_1(k, r)$ in Lemma 20, and a lower bound of and $w_e(k, r)$ in Lemma 21.

▶ **Lemma 19.** *For any $k \geq 4$ and $r \in (r_{core}(k), r_{sat}(k))$, we have $\mu(k, r) < \mu_u(k)$, where*

$$\mu_u(k) = (1 - e^{-1/k}) - (1 - e^{-1/k}) \ln(1 - e^{-1/k}).$$

▶ **Lemma 20.** *For any $k \geq k_0 \geq 3$ and $r \in (r_{core}(k), r_{sat}(k))$, $w_1(k,r) \geq w_1^*(k_0)$, where*

$$w_1^*(k) = \frac{k^{\frac{1}{1-k}}}{k-1}\gamma\left(\frac{1}{k-1}, k\left(\frac{k}{k+1}\right)^{k-1}\right)$$

▶ **Lemma 21.** *For any $k \geq k_0 \geq 3$ and $r \in (r_{core}(k), r_{sat}(k))$, we have $w_e(k,r) \geq w_e^*(k_0, r_{sat}(k_0))$, where $w_e^*(k,r) = x^-(k,r) - kr^2(x^-(k,r))^k$ and*

$$x^{\pm}(k,r) = \left(\frac{1 \pm \sqrt{1 - 4(kr)^{-2}[(kr)^{\frac{1}{k-1}} - 1]}}{2}\right)^{\frac{1}{k-2}}. \tag{1}$$

Proof of Theorem 2. Let $k \geq 9$ and $r \in (r_{core}(k), r_{sat}(k))$. By Lemma 19 and 20, we have $2\mu(k,r) < 2\mu_u(9) \leq 0.3420 < 0.3575 \leq w_1^*(9) \leq w_1(k,r)$. Then, by Lemma 14, $\texttt{DEC}_{\texttt{UC}}$ is strictly $2\mu(k,r)$-free. The result follows. ◀

Proof of Theorem 3. Let $k \geq 13$ and $r \in (r_{core}(k), r_{sat}(k))$. By Lemma 19 and 21, we have $2\mu(k,r) < 2\mu_u(13) \leq 0.2668 < 0.2725 \leq w_e^*(13) \leq w_e(k,r)$. Then, by Lemma 15, $\texttt{DEC}_{\tau}$ is strictly $2\mu(k,r)$-free. The result follows. ◀

References

1. Emmanuel Abbe, Shuangping Li, and Allan Sly. Binary perceptron: Efficient algorithms can find solutions in a rare well-connected cluster. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, pages 860–873, New York, NY, USA, June 2022. Association for Computing Machinery. doi:10.1145/3519935.3519975.
2. Emmanuel Abbe, Shuangping Li, and Allan Sly. Proof of the Contiguity Conjecture and Lognormal Limit for the Symmetric Perceptron. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 327–338, February 2022. doi:10.1109/FOCS52979.2021.00041.
3. Dimitris Achlioptas and Amin Coja-Oghlan. Algorithmic Barriers from Phase Transitions. In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 793–802, Philadelphia, PA, USA, October 2008. IEEE. doi:10.1109/FOCS.2008.11.
4. Dimitris Achlioptas, Amin Coja-Oghlan, and Federico Ricci-Tersenghi. On the solution-space geometry of random constraint satisfaction problems. *Random Structures & Algorithms*, 38(3):251–268, May 2011. doi:10.1002/rsa.20323.
5. Dimitris Achlioptas, Jeong Han Kim, Michael Krivelevich, and Prasad Tetali. Two-coloring random hypergraphs. *Random Structures and Algorithms*, 20(2):249–259, March 2002. doi:10.1002/rsa.997.
6. Dimitris Achlioptas and Michael Molloy. The solution space geometry of random linear equations. *Random Structures & Algorithms*, 46(2):197–231, March 2015. doi:10.1002/rsa.20494.
7. A. Braunstein, M. Mézard, and R. Zecchina. Survey propagation: An algorithm for satisfiability. *Random Structures and Algorithms*, 27(2):201–226, September 2005. doi:10.1002/rsa.20057.
8. Alfredo Braunstein and Riccardo Zecchina. Survey propagation as local equilibrium equations. *Journal of Statistical Mechanics: Theory and Experiment*, 2004(06):P06007, June 2004. doi:10.1088/1742-5468/2004/06/P06007.
9. Guy Bresler and Brice Huang. The Algorithmic Phase Transition of Random k-SAT for Low Degree Polynomials. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 298–309, Denver, CO, USA, February 2022. IEEE. doi:10.1109/FOCS52979.2021.00038.

10 S. Cocco, O. Dubois, J. Mandler, and R. Monasson. Rigorous Decimation-Based Construction of Ground Pure States for Spin-Glass Models on Random Lattices. *Physical Review Letters*, 90(4):047205, January 2003. doi:10.1103/PhysRevLett.90.047205.

11 Amin Coja-Oghlan. A Better Algorithm for Random k -SAT. *SIAM Journal on Computing*, 39(7):2823–2864, January 2010. doi:10.1137/09076516X.

12 Amin Coja-Oghlan. Belief Propagation Guided Decimation Fails on Random Formulas. *Journal of the ACM*, 63(6):1–55, February 2017. doi:10.1145/3005398.

13 Martin Dietzfelbinger, Andreas Goerdt, Michael Mitzenmacher, Andrea Montanari, Rasmus Pagh, and Michael Rink. Tight Thresholds for Cuckoo Hashing via XORSAT. In *Automata, Languages and Programming*, volume 6198, pages 213–225. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. doi:10.1007/978-3-642-14165-2_19.

14 Olivier Dubois and Jacques Mandler. The 3-XORSAT threshold. *Comptes Rendus Mathematique*, 335(11):963–966, December 2002. doi:10.1016/S1631-073X(02)02563-3.

15 David Gamarnik. The overlap gap property: A topological barrier to optimizing over random structures. *Proceedings of the National Academy of Sciences*, 118(41):e2108492118, October 2021. doi:10.1073/pnas.2108492118.

16 David Gamarnik and Eren C. Kızıldağ. Algorithmic obstructions in the random number partitioning problem. *The Annals of Applied Probability*, 33(6B), December 2023. doi:10.1214/23-AAP1953.

17 David Gamarnik, Eren C. Kizildağ, Will Perkins, and Changji Xu. Geometric Barriers for Stable and Online Algorithms for Discrepancy Minimization. In *Proceedings of Thirty Sixth Conference on Learning Theory*, pages 3231–3263. PMLR, July 2023.

18 David Gamarnik and Quan Li. Finding a large submatrix of a Gaussian random matrix. *The Annals of Statistics*, 46(6A), December 2018. doi:10.1214/17-AOS1628.

19 David Gamarnik and Madhu Sudan. Limits of local algorithms over sparse random graphs. *The Annals of Probability*, 45(4), July 2017. doi:10.1214/16-AOP1114.

20 David Gamarnik and Madhu Sudan. Performance of Sequential Local Algorithms for the Random NAE-K-SAT Problem. *SIAM Journal on Computing*, 46(2):590–619, January 2017. doi:10.1137/140989728.

21 Carla P. Gomes and Bart Selman. Satisfied with Physics. *Science*, 297(5582):784–785, August 2002. doi:10.1126/science.1074599.

22 Marco Guidetti and A. P. Young. Complexity of several constraint-satisfaction problems using the heuristic classical algorithm WalkSAT. *Physical Review E*, 84(1):011102, July 2011. doi:10.1103/PhysRevE.84.011102.

23 Samuel Hetterich. Analysing Survey Propagation Guided Decimation on Random Formulas, February 2016. arXiv:1602.08519.

24 Brice Huang and Mark Sellke. Tight Lipschitz Hardness for optimizing Mean Field Spin Glasses. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 312–322, Denver, CO, USA, October 2022. IEEE. doi:10.1109/FOCS54457.2022.00037.

25 Morteza Ibrahimi, Yashodhan Kanoria, Matt Kraning, and Andrea Montanari. The Set of Solutions of Random XORSAT Formulae. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 760–779. Society for Industrial and Applied Mathematics, January 2012. doi:10.1137/1.9781611973099.62.

26 Jeong Han Kim. The Poisson Cloning Model for Random Graphs, Random Directed Graphs and Random k-SAT Problems. In *Computing and Combinatorics*, volume 3106, pages 2–2. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. doi:10.1007/978-3-540-27798-9_2.

27 Lukas Kroc, Ashish Sabharwal, and Bart Selman. Message-passing and local heuristics as decimation strategies for satisfiability. In *Proceedings of the 2009 ACM Symposium on Applied Computing*, pages 1408–1414, Honolulu Hawaii, March 2009. ACM. doi:10.1145/1529282.1529596.

28 Elitza Maneva, Elchanan Mossel, and Martin J. Wainwright. A new look at survey propagation and its generalizations. *Journal of the ACM*, 54(4):17, July 2007. doi:10.1145/1255443.1255445.

29 M. Mézard, G. Parisi, and R. Zecchina. Analytic and Algorithmic Solution of Random Satisfiability Problems. *Science*, 297(5582):812–815, August 2002. doi:10.1126/science.1073287.

30 M. Mézard, F. Ricci-Tersenghi, and R. Zecchina. Two Solutions to Diluted p-Spin Models and XORSAT Problems. *Journal of Statistical Physics*, 111(3/4):505–533, 2003. doi:10.1023/A:1022886412117.

31 Marc Mézard and Andrea Montanari. *Information, Physics, and Computation*. Oxford Graduate Texts. Oxford University Press, Oxford ; New York, 2009.

32 Michael Molloy. Cores in random hypergraphs and Boolean formulas. *Random Structures and Algorithms*, 27(1):124–135, August 2005. doi:10.1002/rsa.20061.

33 Will Perkins and Changji Xu. Frozen 1-RSB structure of the symmetric Ising perceptron. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, pages 1579–1588, New York, NY, USA, June 2021. Association for Computing Machinery. doi:10.1145/3406325.3451119.

34 Boris Pittel and Gregory B. Sorkin. The Satisfiability Threshold for k -XORSAT. *Combinatorics, Probability and Computing*, 25(2):236–268, March 2016. doi:10.1017/S0963548315000097.

35 Boris Pittel, Joel Spencer, and Nicholas Wormald. Sudden Emergence of a Giant k-Core in a Random Graph. *Journal of Combinatorial Theory, Series B*, 67(1):111–151, May 1996. doi:10.1006/jctb.1996.0036.

36 Federico Ricci-Tersenghi and Guilhem Semerjian. On the cavity method for decimated random constraint satisfaction problems and the analysis of belief propagation guided decimation algorithms. *Journal of Statistical Mechanics: Theory and Experiment*, 2009(09):P09001, September 2009. doi:10.1088/1742-5468/2009/09/P09001.

37 Alexander S. Wein. Optimal low-degree hardness of maximum independent set. *Mathematical Statistics and Learning*, 4(3):221–251, January 2022. doi:10.4171/msl/25.