On the Hardness of Satisfiable Random k-XORSAT for Linear-time Algorithms

YUNG, King On

A Thesis Submitted in Partial Fulfilment

of the Requirements for the Degree of

Doctor of Philosophy

in

Computer Science and Engineering

The Chinese University of Hong Kong

September 2024

Abstract

Constraint satisfaction problems (CSPs) provide a common foundation on analysing computational problems and algorithms. Consisting of m constraints on n Boolean variables, the problem is to assign values to variables such that all constraints are satisfied, or to show that no such assignment exists. Many CSPs are notoriously hard, such as k-SAT and k-coloring, which are NP-complete. Even though no efficient algorithm is known to solve all possible instances, it might be possible to have an efficient algorithm that can solve randomly generated instances with non-vanishing probability. Finding and analysing such algorithms for random CSPs are often associated with phase transitions of random CSPs in which some behaviours of a random instance change significantly around certain thresholds of clause density (clause-to-variable ratio).

Random k-XORSAT is a random CSP of rn exclusive disjunction (XOR) clauses on n Boolean variables, where each clause contains k variables. Since the XOR operator is equivalent to the module-2 addiction, an instance can be expressed as a system of linear equations over $\mathbb{F}_2$ of the form $Ax = b$, where A is an $rn \times n$ matrix with k ones per row, and b is a vector of length rn. The existence of the satisfiability threshold $r_{\text{sat}}(k)$ and the clustering threshold $r_{\text{clt}}(k)$ of random k-XORSAT are known such that as $n \to \infty$, for $r < r_{\text{sat}}(k)$ a random instance has solutions with high probability, while for $r_{\text{clt}}(k) < r < r_{\text{sat}}(k)$ the non-empty solution space shatters into an exponential number of clusters. In many random CSPs, no efficient algorithm is known to solve the problem with non-vanishing probability when the clause density is close to their satisfiability thresholds. Although k-XORSAT instances can be solved in polynomial-time for any clause density, we discover a similar algorithmic barrier in random k-XORSAT when we only consider linear-time algorithms as efficient algorithms.

We prove that for any $r_{\text{clt}}(k) < r < r_{\text{sat}}(k)$ the sequential local algorithms with certain local rules fail to solve the random k-XORSAT with high probability. 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 neighbourhood with respect to the factor graph representation of the instance. This result covers (1) the case using the Unit Clause Propagation as local rule for $k \geq 9$, and (2) the case 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). Since the best known linear-time algorithm for random k-XORSAT succeeds with high probability only when $r < r_{\text{clt}}(k)$, this result supports the intuition that $r_{\text{clt}}(k)$ is the sharp algorithmic threshold for the existence of a linear-time algorithm for random k-XORSAT.

摘要

約束滿足問題（CSP）為分析計算問題和演算法提供了共同基礎。該問題由m個約束條件和n個布林變量組成，目的是為變量分配值，使得所有約束條件都得到滿足，或者證明不存在這樣的分配。許多CSP都非常困難，例如k-滿足問題和k-著色問題，它們是NP完全的。儘管目前沒有已知的高效算法能夠解決所有可能的實例，但對於隨機生成的實例，可能存在一個能夠以非零概率解決的高效算法。尋找和分析這些隨機CSP的算法通常與隨機CSP的相變現象相關，其中隨機實例的某些行為在特定的子句密度（子句-變量比）閾值附近發生顯著變化。

隨機k-異或滿足問題是一個隨機CSP，由n個布林變量和rn個異或（XOR）子句組成，每個子句包含k個變量。由於異或運算符等價於模除2加法，一個實例可以表示為一個$\mathbb{F}_2$上的線性方程系統，形式為$Ax = b$，其中A是一個$rn \times n$矩陣，每行有k個1，b是一個長度為rn的向量。已知隨著$n \to \infty$，隨機k-異或滿足問題存在可滿足閾值$r_{\text{sat}}(k)$和聚集閾值$r_{\text{clt}}(k)$。對於$r < r_{\text{sat}}(k)$，具有高概率隨機實例有解，而對於$r_{\text{clt}}(k) < r < r_{\text{sat}}(k)$，非空解空間分裂成指數數量的簇。在許多隨機CSP中，當子句密度接近其可滿足閾值時，沒有已知的高效算法能夠以非零概率解決問題。儘管對於任何子句密度，k-異或滿足問題的實例都可以在多項式時間內解決，但我們在僅考慮線性時間算法作為高效算法時，發現了隨機k-異或滿足問題中的類似算法障礙。

我們證明了對於$r_{\text{clt}}(k) < r < r_{\text{sat}}(k)$，具有特定局部規則的順序局部演算法在解決隨機$k$-異或滿足問題時具有很高的失敗機率。順序局部演算法是一類自然的演算法，透過迭代逐一為變數分配值。在每次迭代中，演算法都會運行一些啓發式方法，稱為局部規則，以基於實例的因子圖表示相對於局部鄰域的方式來決定分配的值。這個結果涵蓋了兩種情況：(1)當$k \geq 9$時使用單子句傳播作為局部規則，以及(2) 當$k \geq 13$時使用任何能夠計算具有樹狀因子圖實例中變數精確邊際機率的局部規則。著名的信念傳播和調查傳播也包含在(2) 中。由於已知的最佳線性時間演算法僅在$r < r_{\text{clt}}(k)$時具有很高的成功機率，因此這個結果支持了$r_{\text{clt}}(k)$是隨機k-異或滿足問題存在線性時間演算法的尖銳演算法閾值。

Acknowledgement

First and foremost, I am extremely grateful to Prof. Andrej Bogdanov for his invaluable supervision and guidance throughout my PhD journey. His professional insights were instrumental in shaping my work, and his trust in giving me the freedom to steer the research direction allowed me to grow as a researcher. I am especially thankful for his patience during the times when I struggled with my research progress.

I would also like to extend my sincere thanks to Prof. Jimmy Ho Man Lee, Prof. Siu On Chan, and Prof. Farzan Farnia for their invaluable participation in my PhD study. Without their support and insights, completing this PhD would have been far more challenging.

I am deeply grateful to Prof. Shengyu Zhang for inspiring my interest in theoretical computer science during my master's degree, which laid the groundwork for my PhD research.

My appreciation also extends to my thesis assessment committees — Prof. Qiang Xu, Prof. Farzan Farnia, Prof. Jimmy Ho Man Lee, and Prof. Alon Rosen — for their thorough review of my thesis and their valuable feedback.

My heartfelt thanks also go to the Chinese University of Hong Kong, where I have spent more than a decade pursuing my bachelor's, master's, and now this PhD degree. The solid foundation I received here has been pivotal to my academic journey.

I would also like to acknowledge my peers, Christopher Williamson and Xin Huang, for their camaraderie and the enjoyable office environment we shared.

Special thanks to Carol S.Y. Chow and Lam Kwai Ming Erica for their crucial support in maintaining my well-being during this journey.

I am immensely grateful to my parents and siblings for their unwavering support and encouragement, which have been a source of strength throughout this adventure. I especially thank my siblings for being there for me during the toughest times, particularly when I struggled with my research. Their presence and understanding made all the difference in helping me push through those difficult moments.

Lastly, I would like to acknowledge myself. There were several occasions when close friends kindly suggested that I consider ending my PhD journey due to their concern for my well-being, given the high pressure and frustration that often accompanied my research. Despite being fully aware of the challenges, I was determined—perhaps even stubborn enough—to press on. And in the end, I completed this journey.

Contents

1 Introduction

Solving a system of linear equations over a finite field $\mathbb{F}_p$ is the one of the most fundamental problems in both mathematics and computer science. A system of linear equations is a constraint satisfaction problem when equations are viewed as constraints on the values assigned to variables. We can further associate it with various problems such as the decision problem in which we determine whether it has a solution, the counting problem in which we count the number of solutions, and the optimization problem in which we find a value assignment to variables so that the number of violated equations is minimized, etc. By using the standard linear algebra methods such as Gaussian elimination, we can determine whether a linear system has a solution, find solutions and count the number of solutions, in $O(n^3)$ time. Some improvements were also made to get even faster. The best known algorithm for general linear systems has the time complexity $O(n^{2.376})$ [CW90].

If we set $p = 2$ so that the underlying finite field $\mathbb{F}_p$ only contains the two Boolean values, 0 and 1, then the linear system becomes a Boolean constraint satisfaction problem, where 1 represents `TRUE` and 0 represents `FALSE`. Among the family of Boolean constraint satisfaction problems, the most studied one is the k-SAT, since it is one of the iconic NP-complete problems. The k-SAT is analogous to a Boolean linear system in which each equation contains exactly k variables, in the sense that the OR operators are replaced by the XOR operators. This type of Boolean linear systems is called k-XORSAT. Due to its analogy with k-SAT and its linear structure, the k-XORSAT provides an ideal playground for understanding k-SAT and other related problems.

Despite concentrated effort on this central problem, no polynomial-time algorithm is known to be able to solve k-SAT in the worst case. The next best thing is to have polynomial-time algorithms that are good enough to solve most but not all instances. In other words, we want to have efficient algorithms that can solve instances on *average cases*. From the probabilistic perspective, it means we look out for algorithms that can solve a randomly generated instance with probability not close to zero. In this work, we focus on random k-XORSAT in the hope that it will provide some insights into understanding random k-SAT as well as other random constraint satisfaction problems. In addition to understand the probabilistic combinatorics of random k-XORSAT itself, studying random k-XORSAT also helps us in understanding many practical use cases in different fields. For example, random k-XORSAT forms a simple but successful code, called *low-density parity-check code*, in coding theory. Random k-XORSAT is also equivalent to the *diluted k-spin model* in statistical mechanics. In the study of data structures, Dietzfelbinger et al. [DGM$^+$10] used random k-XORSAT to establish the tight threshold of performance of the load balancing scheme, *cuckoo hashing.* In cryptography, random k-XORSAT is closely related to the problem called *learning sparse parity with noise*, which provides a promising post-quantum hardness assumption.

Now we consider a randomly generated instance of a constraint satisfaction problem, say random k-SAT. It is possible that we obtain an unsatisfiable instance. Designing an algorithm searching for solutions that do not exist is meaningless. Therefore, before seeking the desired algorithms, it is better to first study the probability that a random instance has a solution. It turns out that finding this probability becomes a long-standing open problem in probabilistic combinatorics. After the breakthrough from Friedgut [Fri99], it is believed that for many random constraint satisfaction problems of interest, there exists a critical value $r_{\text{sat}}(k) > 0$, called the *satisfiability threshold*, of the ratio of the number of constraints to the number of variables such that for the constraint-to-variable ratio below $r_{\text{sat}}(k)$ a random instance has solutions with probability converging to 1, while for the constraint-to-variable ratio above $r_{\text{sat}}(k)$ there is no solution for a random instance with probability converging to 1, as the number of variables n grows to the infinity. Meanwhile, for most of the random constraint satisfaction problems, all known algorithms for those problems do not work when the constraint-to-variable ratio is close to $r_{\text{sat}}(k)$. To be more specific, the largest constraint-to-variable ratio where we have polynomial-time algorithms for finding solutions with probability not converging to zero, called *algorithmic threshold* $r_{\text{alg}}(k)$, is well below $r_{\text{sat}}(k)$. There is a *statistical-to-computation gap* between the believed satisfiability threshold and the algorithmic threshold. Within the gap, we know that solutions are likely to exist while we do not have effective polynomial-time algorithms to find them. The 1-step Replica Symmetry Breaking (1RSB) hypothesis [KMR$^+$07] from statistical physics suggests that this average-case hardness is caused by the clustering phenomenon of the solution space, when the constraint-to-variable ratio is close to the satisfiability threshold.

Getting back to random k-XORSAT, we know random k-XORSAT exhibits the clustering phenomenon of solution space when the constraint-to-variable ratio is close to its satisfiability threshold, but we have polynomial-time algorithms that can solve any k-XORSAT instance regardless the constraint-to-variable ratio. Therefore, it is clear that random k-XORSAT is an exceptional case of the above hypothesis. However, we notice that we do not have linear-time algorithms that can solve a random k-XORSAT instance with probability not converging to zero, when the clustering phenomenon emerges. This leads us to the linear-time version of the above hypothesis.

Our primary objective is to study the average-case hardness of random k-XORSAT with respect to linear-time algorithms, instead of polynomial-time algorithms. In this work, we rule out a natural class of algorithms, *sequential local algorithms*, from being able to solve random k-XORSAT instances for constraint-to-variable ratio close to the satisfiability threshold. Together with the linear-time algorithm from [IKKM12] that can solve random k-XORSAT with high probability for low constraint-to-variable ratio, this helps us determine the sharp linear-time algorithmic threshold of random k-XORSAT. To do so, we leverage the well-known results from [IKKM12, AM15] about the clustering phenomenon on the solution space of random k-XORSAT instances, and introduce an improvement to a recently developed proof technique based on the notion of *overlap gap property*. Our result successfully links the average-case hardness, the clustering phenomenon and the overlap gap property all together for random k-XORSAT.

This thesis is based on the paper *Limits of Sequential Local Algorithms on the Random k-XORSAT Problem* by Yung (ICALP 2024), which was selected as the Best Student Paper of Track A "Algorithms, Complexity and Games" at ICALP 2024. This prize is awarded to the best paper submitted to ICALP exclusively by full-time student(s).

1.1 Random k-XORSAT

The **k-XOR satisfaction problem** (k-XORSAT) is a Boolean constraint satisfaction problem. In general, a **constraint satisfaction problem** CSP is a problem consisting of a set of variables $X = \{x_1, x_2, \ldots, x_n\}$, a set of their respective domains of values $D = \{D_1, D_2, \ldots, D_n\}$, and a set of constraints $C = \{c_1, c_2, \ldots, c_m\}$. Each variable $x_i \in X$ can be assigned a value from its respective domain of values $D_i \in D$. Each constraint $c_j \in C$ specifies a requirement on the values assigned to a subset of variables. In particular, a constraint c_j is a pair (S_j, R_j), where $S_j \subset X$ is a subset of k variables and R_j is a k-ary relation on the respective domains. The relation R_j lists out all combinations of values on the variables in S_j that satisfy the requirement. Besides listing out all satisfying combinations, a constraint can be expressed in different ways such as an equation that has to hold, or a Boolean formula that has to evaluate to `TRUE`, etc. Given a CSP instance, various tasks can be performed, including determine whether it has at least one value assignment to all variables that satisfies all constraints, finding such an assignment, finding all such assignments, etc. A Boolean CSP is a CSP in which the domain of values of each variable is the set $\{0, 1\}$, where 0 represents the `FALSE` value and 1 represents the `TRUE` value. In the context of Boolean CSP, a constraint is usually called a **clause**.

Instance of k-XORSAT An instance Φ of the k-XORSAT consists of m clauses in n Boolean variables $x_1, x_2, \ldots, x_n$. Each clause is a Boolean linear equation $x_{j_1} \oplus x_{j_2} \oplus \cdots \oplus x_{j_k} = b_j$ in k variables, where $\oplus$ is the modulo-2 addition in $\mathbb{F}(2)$. Note that the modulo-2 addition $\oplus$ is equivalent to the logical operator XOR. This explains why we have the word "XOR" in the name of this CSP. By convention, if not all clauses contain exactly k variables, then we simply call it an XORSAT instance, without the prefix k. We are interested in the task of finding a Boolean-valued assignment to all variables so that all equations hold. This task is equivalent to solving the system of linear equations.

Assignments for instances An **assignment** $\sigma : \{x_i : i \in [n]\} \to \{0, 1\}$ for the instance Φ 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 the notation, we can write it as the Boolean vector $\sigma = (\sigma(x_1), \sigma(x_2), \ldots, \sigma(x_n)) \in \{0, 1\}^n$ containing the assigned values, in the order of the indices of the variables. The **distance** $d(\sigma, \sigma')$ between any two assignments σ and σ' is defined to be the Hamming distance $d(\sigma, \sigma') = \sum_{i=1}^{n} \mathbb{1}(\sigma(x_i) \neq \sigma'(x_i))$ of the two corresponding Boolean vectors. We say a clause c is **satisfied by** the assignment σ if the clause c holds when the variables are replaced by the corresponding values assigned by σ. We also say the instance Φ is **satisfied by** the assignment σ if all its clauses are satisfied by σ, and in this case we call the assignment σ a **solution** of the instance

Φ. Moreover, we say the instance is **satisfiable** if it has at least one solution, or **unsatisfiable** otherwise. The set of all solutions of the instance Φ is called the **solution space** of the instance Φ, denoted by $\mathcal{S}(\Phi)$. Therefore, the instance Φ is satisfiable if and only if the solution space $\mathcal{S}(\Phi)$ is non-empty.

Random instances We are particularly interested in random instances of the k-XORSAT. In a random instance $\mathbf{\Phi}$, each clause is drawn independently. For each clause, 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. Note that the left-hand side and the right-hand side of the equation are drawn independently. It is equivalent to say that each clause is chosen uniformly from all $2\binom{n}{k}$ possibilities, independently. Hence, a random instance $\mathbf{\Phi}$ of the k-XORSAT problem is drawn uniformly from the ensemble $\mathbf{\Phi}_k(n, m)$ of all possible k-XORSAT instances of n variables and m clauses, where each clause contains exactly k variables. We denote this by $\mathbf{\Phi} \sim \mathbf{\Phi}_k(n, m)$. We focus on the regime in which the **clause density** (or simply **density**), defined to be the ratio $r = m/n$, converges to a constant independent of n, assuming the number of variables n goes to the infinity. Informally speaking, we focus on the large instances with number of clauses proportional to the number of variables.

Algorithmic hardness of random k-XORSAT All k-XORSAT instance can be solved in polynomial time. Since the clauses of a k-XORSAT instance Φ are nothing but Boolean linear equations over $\mathbb{F}_2$, the instance Φ is equivalent to a linear system over $\mathbb{F}_2$ with n variables and m equations, where each equation has k variables. Therefore, the 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, random k-XORSAT appears to be difficult to solve by other algorithms. Achlioptas and Molloy mentioned in their paper [AM15] that random k-XORSAT instances seem to be extremely difficult for both generic CSP solvers and SAT solvers, which usually do not take the linear algebraic structure into account. Guidetti and Young [GY11] 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 linear systems in general. This leads us to our main question:

How hard is it to solve a random k-XORSAT instance by linear-time algorithms?

Prior to diving into the investigation and analysis of linear-time algorithms for random k-XORSAT, it is essential to understand the behaviour of the random k-XORSAT itself. In order to do so, we might first get some insight from random k-SAT, which is closely analogous to random k-XORSAT.

Analogous to random k-SAT Studying analogous problems may help us understand random k-XORSAT. The most studied problem among the family of random Boolean CSPs is the **random k-satisfaction problem** (random k-SAT). In k-SAT instances, a clause is a disjunction of literals of k variables, where a literal of a variable x_i is either the variable x_i itself or its negation $\overline{x_i}$. For a random k-SAT instance, each clause is drawn independently in the following way: The k variables in the disjunction are drawn from the $\binom{n}{k}$ possibilities, uniformly at random, and each variable is replaced by its negation with probability 1/2.

Random k-SAT is closely analogous to random k-XORSAT. Replacing a variable by its negation is equivalent to applying the modulo-2 addition of 1 to the variable, so a clause c_j in a random k-SAT instance can be written as the following equation

$$(x_{j_1} \oplus b_{j_1}) \vee (x_{j_2} \oplus b_{j_2}) \vee \cdots \vee (x_{j_k} \oplus b_{j_k}) = 1, \tag{1}$$

where $b_{j_1}, b_{j_2}, \ldots, b_{j_k}$ are k i.i.d. random Boolean variables, each with even distribution over $\{0,1\}$. On the other hand, suppose we have a clause $x_{j_1} \oplus x_{j_2} \oplus \cdots \oplus x_{j_k} = b_j$ in a random k-XORSAT instance $\mathbf{\Phi}$. We can replace the right-hand side with the sum of k i.i.d random Boolean variables $b_{j_1}, b_{j_2}, \ldots, b_{j_k}$, each with even distribution over $\{0,1\}$, plus 1. We can do this because b_j and $b_{j_1} \oplus b_{j_2} \oplus \cdots \oplus b_{j_k} + 1$ have the same distribution, independent of the rest of the equation. By simple arithmetic operations, the clause can be written as

$$(x_{j_1} \oplus b_{j_1}) \oplus (x_{j_2} \oplus b_{j_2}) \oplus \cdots \oplus (x_{j_k} \oplus b_{j_k}) = 1. \tag{2}$$

We can immediately see that the equations (1) and (2) only differ in the types of operators between the random literals, where disjunction $\vee$ is used in random k-SAT while exclusive disjunction $\oplus$ is used in random k-XORSAT.

Phase transition of random CSPs There are many common phenomena shared by different CSPs including random k-XORSAT and random k-SAT. One of the widely studied phenomena is the *phase transition*. It studies the existences and values of sharp thresholds on the clause density r across where the behaviour of a random instance changes sharply.

Take the satisfiability as an example. When the clause density r is extremely low, we can expect that the random instance $\mathbf{\Phi}$ is likely to be satisfiable, since we only make a few restrictions on value assignments. On the other hand, when the clause density r is extremely high, we can expect that the random instance $\mathbf{\Phi}$ is likely to be unsatisfiable, since we make too many restrictions on the value assignments such that no assignment can satisfy all clauses. Interestingly, for large n, when the clause density r increases, the random instance $\mathbf{\Phi}$ changes from being likely to be satisfiable to being likely to be unsatisfiable *sharply* at a certain clause density, rather than *gradually*. Friedgut [Fri99] showed that a wide range of problems, most importantly including random k-SAT, have such a sharp n-dependent satisfiability threshold – the largest density for which a random instance has solutions with high probability. (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, $\lim_{n\to\infty} \Pr[\mathcal{E}_n] = 1$.) This suggests the existence of the **satisfiability threshold** $r_{\text{sat}}(k)$, independent of n, of a random CSP so that the random instance is satisfiable w.h.p. for clause density $r < r_{\text{sat}}(k)$, and it is unsatisfiable w.h.p. for clause density $r > r_{\text{sat}}(k)$. The result from [Fri99] makes searching for tight bounds of the satisfiability thresholds a promising target.

There is a fruitful line of research studying the phase transitions of random k-SAT. More information about it can be found in Appendix A.

Statistical-to-computation gap For many random CSPs such as random k-SAT, random graph coloring, random hypergraph 2-coloring and random Max k-SAT, the asymptotically tight bounds of their satisfiability thresholds have been determined. However, all known polynomial-time algorithms for finding solutions only succeed for clause densities much lower than those for which we know the solutions exists. We denote by the **algorithmic threshold** $r_{\text{alg}}(k)$ of a random CSP the largest clause density for which a polynomial-time algorithm for finding solutions with non-vanishing probability is known. (We say an event $\mathcal{E}_n$, depending on a number n, occurs **with non-vanishing probability**, if the probability of the event $\mathcal{E}_n$ occurring does not converge to 0 as n goes to the infinity, that is, $\lim_{n\to\infty} \Pr[\mathcal{E}_n] \neq 0$.) There exists a **statistical-to-computation gap** between the algorithmic threshold and the satisfiability threshold. For example, the satisfiability threshold of random k-SAT is $2^k \ln 2 - \frac{1+\ln 2}{2} + o_k(1)$ for $k \geq k_0$ for some constant $k_0 \geq 3$ [DSS15], but the best known polynomial-time algorithm for finding solutions of a random k-SAT instance succeeds w.h.p. for clause density $r < (1 - o_k(1))\, 2^k \,\frac{\ln(k)}{k}$ [Coj10], and seems to fail beyond. The notations $O_k(\cdot)$ and $o_k(\cdot)$ represent the standard asymptotic notation as k increases, instead of assuming that n increases in usual cases. Therefore, for clause density $(1 - o_k(1))\, 2^k \,\frac{\ln(k)}{k} < r < 2^k \ln 2 - \frac{1+\ln 2}{2} + o_k(1)$, we know a random k-SAT instance has solutions w.h.p. but we do not have an efficient algorithm to find them in polynomial-time.

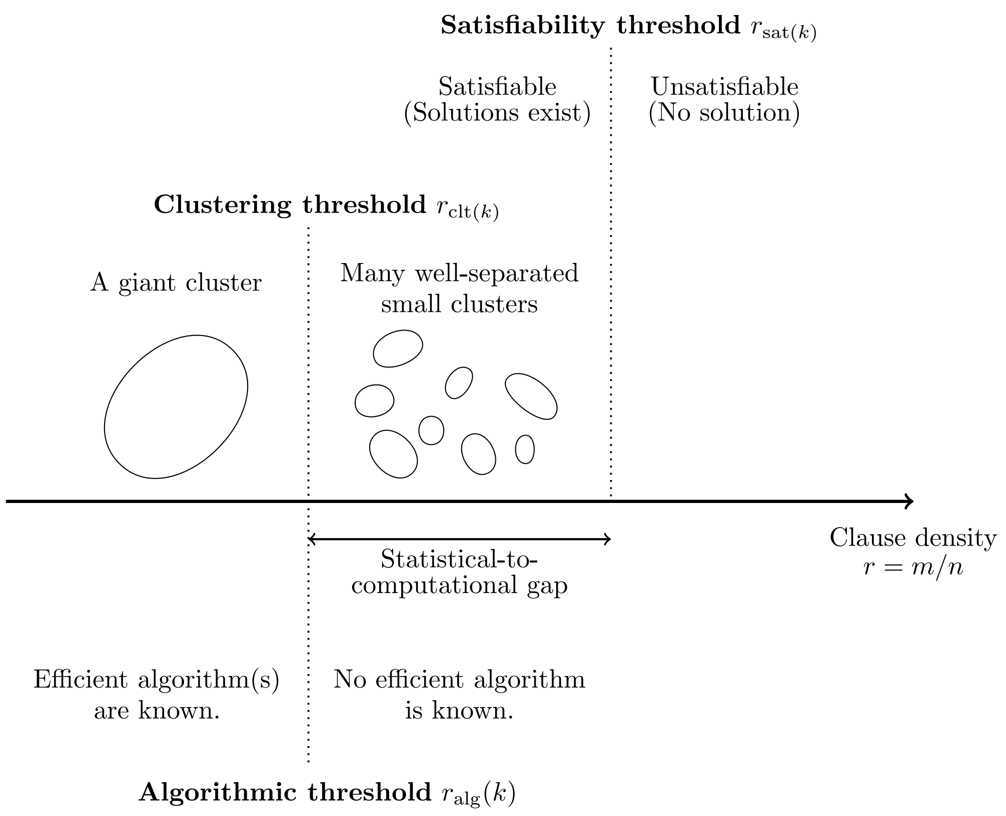
Clustering and hardness The barrier of the algorithmic hardness seems to link with the geometry of the solution spaces. Mézard, Mora and Zecchina [MMZ05], and Achlioptas and Ricci-Tersenghi [AR06] first observed that the solution space of a random k-SAT instance undergoes a dramatic change in geometric structure near the algorithmic threshold $r_{\text{alg}}(k)$. Achlioptas and Coja-Oghlan [AC08] further showed that w.h.p. the solution space of a random k-SAT instance can be partition into exponentially many well-separated subsets, each containing at most an exponentially small fraction of all solutions, for clause density $(1 + \epsilon_k)2^k \frac{\ln k}{k} < r < (1 - \epsilon_k)2^k \ln 2$, for some sequence $\epsilon_k \to 0$. This "clustering" phase starts near $r_{\text{alg}}(k)$ at which the best known algorithm from [Coj10] stops working. It seems that it is not a coincidence only on random k-SAT, since [AC08] also showed the similar results for random graph coloring and random hypergraph 2-coloring: the best algorithm stops working at which the solution space starts shattering.

In different prior works, different terminologies were used to describe the clustering phenomenon. Referencing the survey from Gamarnik [Gam21], we use the following definition in this paper. We say an instance Φ exhibits the **clustering property** if the solution space $\mathcal{S}(\Phi)$ can be partitioned into connected components, called **clusters**, each containing at most an exponentially small in n fraction of solutions, separated by distances of order $\Omega(n)$ with each other, by saying two solutions σ, σ' are adjacent if the distance $d(\sigma, \sigma')$ between them is at most $o(n)$. (In the context of physics, the term "cluster" usually requires $d(\sigma, \sigma') = 1$ to say the two solutions σ, σ' are adjacent. However, the results from [AC08] mentioned above also hold when the requirement $d(\sigma, \sigma') = 1$ is replaced by $d(\sigma, \sigma') = o(n)$.) Then, we define the **clustering threshold** $r_{\text{clt}}(k)$ to be the value at most $r_{\text{sat}}(k)$ such that a random instance exhibits cluster the clustering property w.h.p. for clause density $r_{\text{clt}}(k) < r < r_{\text{sat}}(k)$, and a random instance does not exhibit the clustering property w.h.p. for clause density $r < r_{\text{clt}}(k)$.

Common pattern in phase transition The similar pattern of phase transitions can be found across many random CSPs. With random k-SAT, random graph coloring and random hypergraph 2-coloring as concrete examples, Achlioptas and Coja-Oghlan [AC08] rounded up the pattern for most random CSPs of interest, into the list shown below, and believed that it is a universal, problem-independent phenomenon for random CSPs. We also visualize the pattern in Figure 1.

1. There is an upper bound on the largest clause density for which the solutions exist (usually proved via the first moment method).
2. The bound from (1) is essentially tight, with a non-constructive proof (usually via the second moment method). Combining with (1), we can determine the *satisfiability threshold*.
3. Some simple algorithms can find solutions in polynomial time for clause densities much lower than the one from (2).
4. No polynomial-time algorithm is known to be able to find solutions for clause density greater than the one from (3). This leads to the *algorithmic threshold*, and a *statistical-to-computation gap* between the satisfiability threshold and the algorithmic threshold.
5. The solution space undergoes a structural change sharply, from a "giant ball" at low density to many well-separated "small clusters" at high density, at the *clustering threshold*, which coincides the algorithmic threshold.

Even through we can show the solution space of a random CSP undergoes a structural change at the largest clause density where polynomial-time algorithms are known to solve the problem, we cannot confirm whether the solution space geometry is responsible to the algorithmic hardness of the random CSP, or it is just a coincidence. We still need a more sophisticated theory to explain the linkage between the solution space geometry and the algorithmic hardness for the random CSPs. One possible way is to rule out different classes of efficient algorithms by the clustering property.



Satisfiability threshold $r_{\text{sat}(k)}$
Satisfiable (Solutions exist)
Unsatisfiable (No solution)
Clustering threshold $r_{\text{clt}(k)}$
A giant cluster
Many well-separated small clusters
Statistical-to-computational gap
Clause density $r = m/n$
Efficient algorithm(s) are known.
No efficient algorithm is known.
Algorithmic threshold $r_{\text{alg}}(k)$

Figure 1: Visualization of the relative positions of the satisfiability threshold, the algorithmic threshold and the clustering threshold, for a random CSP, in general case.

It is worth to mention that not every random constraint satisfaction problems share this set of phenomena. A notable example is *symmetric binary perceptron* (SBP). Its satisfiability threshold $r_{\text{sat}}(k) > 0$ was established by [PX21, ALS22b]. They showed that for clause density $r > 0$ w.hp. SBP exhibits clustering property, and surprisingly almost all clusters are singleton. On the other hand, [ALS22a] gave a polynomial-time algorithm that can find solutions w.h.p., for some low clause densities. Therefore, there is a regime of low clause densities in which SBP exhibits clustering property, and it is solvable in polynomial time, simultaneously. Its clustering property does not cause the polynomial-time algorithmic hardness.

Phase transition of random k-XORSAT Random k-XORSAT is another example of the fact that the clustering property does not cause the polynomial-time algorithmic hardness since we can solve all k-XORSAT instances regardless their clause densities. But still, the values of its satisfiability threshold and clustering threshold have been precisely determined, thanks to the orderly algebraic structure.

For the satisfiability threshold, Dubois and Mandler [DM02] first proved the existence of the satisfiability threshold $r_{\text{sat}}(k)$ for $k = 3$, and determined the exact value of $r_{\text{sat}}(3)$ by a second moment argument. They believed that their argument could be extended for general $k > 3$. Mézard, Ricci-Tersenghi and Zecchina [MRZ03] gave a heuristic argument of the satisfiability threshold of random k-XORSAT for $k \geq 3$, by using the non-rigorous cavity method and the second moment method. A work on cuckoo hashing from [DGM$^+$10] also included an argument of the satisfiability threshold of random k-XORSAT for general $k \geq 3$. Finally, Pittel and Sorkin [PS16] gave a complete proof of the existence of the satisfiability threshold of random k-XORSAT for $k \geq 3$, by using the critical-set approach of Kolčin [Kol99], instead of using the second moment method. From [PS16], we know, for any $k \geq 3$, the value of the satisfiability threshold of random k-XORSAT is given by

$$r_{\text{sat}}(k) = \frac{\lambda_k}{k(1-e^{-\lambda_k})^{k-1}}$$

where λ_k is the root of the equation

$$\frac{x(e^x-1)}{e^x-1-x} = k.$$

For the clustering threshold, Mézard, Ricci-Tersenghi and Zecchina [MRZ03] started the studying of the clustering phenomenon of random k-XORSAT, and gave a non-rigorous argument about it. After that, Ibrahimi et al. [IKKM12] and Achlioptas and Molloy [AM15] successfully determined the clustering threshold $r_{\text{clt}}(k)$, independently. The value of $r_{\text{clt}}(k)$ is given by

$$r_{\text{clt}}(k) = \min_{x>0} \frac{x}{k(1-e^{-x})^{k-1}} \tag{3}$$

Similar to the symmetric binary perceptron, the clustering property of random k-XORSAT does not cause the polynomial-time hardness, since we have linear algebra methods that can solve any k-XORSAT instance in polynomial time, regardless the clause density. However, we do not have linear-time algorithms for k-XORSAT instances in general. Therefore, we are interested in studying the algorithmic hardness of random k-XORSAT with respect to linear-time algorithms, where only linear-time algorithms are considered as efficient algorithms. In [IKKM12], they provided an algorithm that can find solutions in linear time w.h.p. for density $r < r_{\text{clt}}(k)$. Meanwhile, no linear-time algorithm is known to find solutions for clause density $r > r_{\text{clt}}(k)$ with non-vanishing probability. We believe that the clustering threshold $r_{\text{clt}}(k)$ could be related to the algorithmic hardness of random k-XORSAT, just like the other random CSPs, except the polynomial-time algorithmic hardness are replaced by linear-time algorithmic hardness.

1.2 Sequential local algorithms

In this work, we consider a natural class of algorithms, called *sequential local algorithms*. A sequential local algorithm selects an unassigned variable randomly, assigns a value to it, and simply the instance, 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, and the output assignment is uniformly distributed over all solutions.

However, we restrict the power of the local rules by only providing the *local structure* closely related the selected variable to the local rules. To distinguish the local structure closely related to the selected variables from the rest, we consider the graphical representation, called *factor graph*, of instances. We will discuss the factor graphs with more details in Section 2.2. In short, each variable is represented by a variable node, each equation is represented by equation node, and a variable node is connected to an equation node by an edge if the corresponding variable is involved in the corresponding equation. The **distance** between any two nodes is the number of edges in the shortest path connecting the two nodes. The **local neighborhood** $B_\Phi(v, R)$ with radius $R \geq 0$ of a node v is the sub-instance induced by the subgraph of G containing all nodes with distances at most R from the node v and edges among those nodes. In each iteration, we only provide the local neighborhood of the residue instance to the local rule to calculate the value p.

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* $\mathtt{DEC}_\tau$. Note that if the local rule τ takes constant time, then the τ-decimation algorithm also takes linear time.

τ-decimation algorithms 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

```
1: Input:
       Instance of the $k$-XORSAT problem $\Phi$
       Even number $R \geq 0$
       Local rule $\tau : \mathcal{I}_R \to [0,1]$
2: Set $\Phi_0 = \Phi$.
3: for $t = 0, ..., n-1$ do
4:     Select an unassigned variable $x^*$ from $\Phi_t$, uniformly at random.
5:     Set $\sigma(x^*) = \begin{cases} 1 & \text{with probability } \tau(B_{\Phi_t}(x^*, R)) \\ 0 & \text{with probability } 1 - \tau(B_{\Phi_t}(x^*, R)) \end{cases}$
6:     Obtain $\Phi_{t+1}$ from $\Phi_t$ by:
           (i) Remove $x^*$.
           (ii) For clauses having $x^*$ before (i), add $\sigma(x^*)$ to its right-hand-side value.
           (iii) Remove all clauses that no longer contain any variable.
7: end for
8: Output: the assignment $\sigma$.
```

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. At the other extreme, 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 1. 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 2. 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 (a clause with one variable only) 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`

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

Message passing algorithms A new challenger to break the algorithmic threshold came out from statistical mechanics. In experiments [MPZ02, GS02, BMZ05, KSS09, RS09], 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 *Belief Propagation* and *Survey Propagation* inspired by the insightful but non-rigorous *cavity method* from statistical mechanics [BMZ05, RS09]. Belief Propagation `BP` and Survey Propagation `SP` showed notable performance in approximating the marginal probabilities of variables over a randomly chosen solution, which is ideal as local rules for sequential local algorithms.

Nevertheless, we only have a few rigorous analyses of the performance of those algorithms on random constraint satisfaction problems. Unfortunately those analyses conclude that these algorithms cannot overcome the topological barrier from the clustering phenomenon, and do not outperform the existing algorithms to push the algorithmic threshold toward the satisfiability threshold. Coja-Oghlan [Coj11] gave the first rigorous analysis of the `BP`-guided Decimation on the random k-SAT problem, and proved that w.h.p. the algorithm fails to solve the problem when the density exceeds $\rho_0 \cdot 2^k/k$ for some universal constant $\rho_0 > 0$. It does not outperform the best known algorithm from [Coj10], which can find a solution w.h.p. for the clause density $r < (1-o_k(1))2^k \ln(k)/k$. Building upon this work, Hetterich [Het16] also proved that w.h.p. the `SP`-guided Decimation fails to solve the random k-SAT problem when the density exceeds $(1 + o_k(1))2^k \ln k/k$. The `SP`-guided decimation also does not outperform the best known algorithm from [Coj10]. Still, we are interested in the performance of sequential local algorithms aimed with Belief Propagation and Survey Propagation as local rules.

1.3 Our contributions

The main results of our work consist 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 $\mathbf{\Phi}$, when the clause density r is above the clustering threshold $r_{\text{clt}}(k)$ and below the satisfiability threshold $r_{\text{sat}}(k)$. This can be formally written as Theorem 1, and the proof will be given in Section 3. Note that the value $\mu(k,r)$ stated in the theorem is an upper bound of the normalized diameter of clusters in the solution space $\mathcal{S}(\mathbf{\Phi})$, which will be discussed in Lemma 4 of Section 2.1.

Theorem 1. *For any $k \geq 3$ and $r \in (r_{\text{clt}}(k), r_{\text{sat}}(k))$, 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 w.h.p. the output assignment $\texttt{DEC}_\tau(\mathbf{\Phi})$ from the algorithm $\texttt{DEC}_\tau$ on input $\mathbf{\Phi}$ is not in the solution space $\mathcal{S}(\mathbf{\Phi})$, that is,*

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

where $\mu(k,r)$ is the real-valued function 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 $Q = 1 - \exp(-krQ^{k-1})$.

The best known linear-time algorithm of finding a solution for the random k-XORSAT instance succeeds w.h.p., for $k \geq 3$ and $r < r_{\text{clt}}(k)$ [IKKM12]. That means the (strictly $2\mu(k,r)$-free) sequential local algorithms do not outperform the best known linear-time algorithm. Now we know that $r_{\text{clt}}(k)$ is where the best known linear-time algorithm succeeds up to, and where the sequential local algorithms starts failing. These support the intuition that $r_{\text{clt}}(k)$ is the sharp algorithmic threshold of linear-time algorithms for random k-XORSAT.

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 for finite k. By doing so, we can give the sharp algorithmic threshold for finite k, rather than providing asymptotic algorithmic threshold. One of them is the simplest local rule, the Unit Clause Propagation $\texttt{UC}$, 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 $\texttt{UC}$-decimation algorithm $\texttt{DEC}_{\texttt{UC}}$, 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 $k \geq 9$, $r \in (r_{\text{clt}}(k), r_{\text{sat}}(k))$, given a random k-XORSAT instance $\mathbf{\Phi} \sim \mathbf{\Phi}_k(n, rn)$, we denote by $\texttt{DEC}_{\texttt{UC}}(\mathbf{\Phi})$ the output assignment from the $\texttt{UC}$-decimation algorithm $\texttt{DEC}_{\texttt{UC}}$ on input $\mathbf{\Phi}$. Then, we have*

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

In each iteration, the role of the local rules is to approximate 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 [MPZ02, GS02, BMZ05, RS09, KSS09]. In particular, it is well-known that they can calculate the exact marginal probabilities of variables when the underlying factor graph is a tree. Furthermore, we know that w.h.p. the local neighborhood of the factor graph of the random k-XORSAT instance is a tree. 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. Unfortunately, the information we provide to the local rules is limited to local neighborhoods of the selected variables, which may not contain much information about the whole instance. We show that the τ-decimation algorithm armed with local rules as powerful as `BP` and `SP` still cannot find a solution with high probability. Precisely speaking, we prove that even the local rule τ outputs 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 $k \geq 13$. This result is formally written in Theorem 3. Both `BP`-decimation algorithm $\mathtt{DEC}_{\mathtt{BP}}$ and `SP`-decimation algorithm $\mathtt{DEC}_{\mathtt{SP}}$ are included in Theorem 3.

Theorem 3. *For any $k \geq 13$, $r \in (r_{\text{core}}(k), r_{\text{sat}}(k))$, given a random k-XORSAT instance $\mathbf{\Phi} \sim \mathbf{\Phi}_k(n, rn)$, denote by $\mathtt{DEC}_\tau(\mathbf{\Phi})$ the output assignment from the τ-decimation algorithm $\mathtt{DEC}_{\mathtt{UC}}$ 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*

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

The proofs of Theorem 2 and Theorem 3 will be given in Section 4. To prove Theorem 2 and Theorem 3, we only need to calculate the number of free steps in $\mathtt{DEC}_{\mathtt{UC}}$ and the number of free steps in $\mathtt{DEC}_\tau$ 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 centred 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 $\texttt{DEC}_{\texttt{UC}}$ is slightly different from the well-known Unit Clause algorithm `UCA`, which runs in the following way: Recursively, assign a suitable value to the variable in a unit clause to satisfy the unit clause if exists, or assign a randomly chosen value to a randomly chosen unassigned variable otherwise. Under the framework of sequential local algorithm, the variables are selected in a random order. However, in the Unit Clause algorithm `UCA` the variables in unit clauses are always selected first if they exist [AKKT02]. 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 [Coj17] and Survey Propagation Decimation Algorithm [BZ04, BMZ05, MMW07]. 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.

1.4 Related works

We observe that symmetric binary perceptron SBP mentioned in Section 1.1 has some similar behaviours as random k-XORSAT. The first one is that the clustering property does not cause its hardness (with respect to polynomial-time algorithms). For low clause density, SBP exhibits clustering [PX21, ALS22b], but it is polynomial-time solvable by the algorithm from [ALS22a]; for density $r_{\text{clt}}(k) < r < r_{\text{sat}}(k)$, random k-XORSAT exhibits clustering [IKKM12, AM15], but it is polynomial-time solvable by some standard linear algebra methods. The second common behaviour is that both the solutions of SBP instances and the solutions of the 2-core instances (not the whole instances) of random k-XORSAT are *highly isolated*. Our study on the hardness with respect to linear-time algorithms may bring some insight to the dramatic statistical-to-computational gap of SBP. Furthermore, the m-OGP (a variant of OGP) of SBP has been established recently by [GKPX23]. It might be interesting if we can obtain similar result for random k-XORSAT.

2 Proof technique

Let recall some key concepts discussed in previous sections. For many random CSPs, we have the satisfiability threshold, which is the largest density for which w.h.p. a random instance has solutions. We also have the algorithmic threshold, which is the largest density for which time-efficient (polynomial-time for most of the random CSPs, or linear-time for random k-XORSAT) algorithms for finding solutions with non-vanishing probability are known. However, the algorithmic threshold is much below the satisfiability threshold so that there is a statistical-to-computational gap. To explain this mysterious statistical-to-computational gap, one of the ways is to rule out all time-efficient algorithms (or at least some natural classes of algorithms) for the clause densities within the gap. Meanwhile, we can observe that the clustering threshold, which separate the regime of exhibiting the clustering property and the regime of not exhibiting it, coincides the algorithmic threshold. It is natural to ask whether we can link the clustering phenomenon to the algorithmic hardness, and rule out some natural classes of algorithms from being able to find solutions, by using the clustering property.

The works from [GS17a, GS17b] and subsequent works leveraged a different notion of clustering, *overlap gap property* OGP, and attempted to link the clustering property to the algorithmic hardness with rigorous proofs. The approach based on OGP successfully rules out some large classes of algorithms such as stable algorithms and algorithms based on low-degree polynomials, for the regime where the clustering property emerges, on some random constraint satisfaction problems and optimization problems on random structures. Starting from [GS17a, GS17b], various classical algorithms has been ruled out for some problems well below the point where solution exists, for example, balanced sequential local algorithms for random NAE-k-SAT problem [GS17b], low-degree polynomial algorithms for maximum independent set problem [Wei22] and random k-SAT problem [BH22], sufficiently stable algorithms for random number partitioning problem [GK23].

2.1 Overlap gap property OGP

The *overlap gap property* OGP was first discovered by Mézard, Mora and Zecchina in [MMZ05], and Achlioptas and Ricci-Tersenghi in [AR06], and named by Gamarnik in [GL18]. Gamarnik gave a detailed survey about OGP and its variants [Gam21]. Intuitively, this property states that every two solutions are either close to each other or far from each other. In the other words, the distance between every two solutions cannot be in a middle range. This notion can be applied to both satisfaction problems and optimization problems, with slightly different definitions. In this work, we stick to the definition based on constraint satisfaction problems.

Definition 3. Given an instance Φ of a CSP, we say it exhibits **overlap gap property OGP** with values v_1 and v_2, where $0 \leq v_1 < v_2$, if every pair of solutions $\sigma, \sigma' \in \mathcal{S}(\Phi)$ satisfy either $d(\sigma, \sigma') \leq v_1$ or $d(\sigma, \sigma') \geq v_2$.

Relation with clustering property This property immediately implies that we can split the solution space into well-separated subsets. Suppose we have an instance Φ that exhibits OGP with values v_1 and v_2, where $0 \leq v_1 < v_2$. We can pick an arbitrary solution σ and split the solution space into at least two subsets: one contains all solutions at distance $\leq v_1$ from σ, and another one contains all solutions at distance $\geq v_2$ from σ. By splitting the solution space in this way, the distance between these two subsets is at least $v_2 - v_1$.

If $2v_1 < v_2$ additionally, there exists a unique partition $\bigsqcup_{i=1}^{L} C_i$ of the solution space $\mathcal{S}(\Phi)$ constructed as follows. We first define a relation $\sim$ on $\mathcal{S}(\Phi)$ by saying $\sigma \sim \sigma'$ if and only if $d(\sigma, \sigma') \leq v_1$. It is obvious that this relation is reflexive and symmetric. If $\sigma_1 \sim \sigma_2$ and $\sigma_2 \sim \sigma_3$ for some $\sigma_1, \sigma_2, \sigma_3 \in \mathcal{S}(\Phi)$, then $d(\sigma_1, \sigma_3) \leq d(\sigma_1, \sigma_2) + d(\sigma_2, \sigma_3) \leq 2v_1 < v_2$, which implies that $d(\sigma_1, \sigma_3) \leq v_1$ due to the OGP of Φ. Thus, this relation is transitive. Therefore, it is an equivalence relation, and the equivalence classes $C_1, C_2, \ldots, C_L$ form a partition of $\mathcal{S}(\Phi)$. By doing so, the diameter of each class is at most v_1, and the distance between every pair of classes is at least v_2.

It is worth to mention that it is not clear whether the converse holds. That means we do not know whether the clustering property and the overlap gap property are equivalent to each other, or not. If one can find an example that the CSP exhibits clustering property without overlap gap property, we will be able to rule out the implication of OGP from the clustering property. However, we do not have such an example so far.

Outline of OGP-based approach Now we outline the OGP-based approach with the basic form of OGP, defined in Definition 3, on a generic random CSP. Some details of this approach are slightly different if we consider different variants of OGP. However, the overall idea of *topological barriers* stays the same, and it is not hard to extend the approach we described below.

Assume we have a random CSP whose solution space exhibits OGP with high probability, and we have a randomized algorithm $\mathcal{A}$ that takes a random instance $\mathbf{\Phi}$ as input and outputs an assignment σ for $\mathbf{\Phi}$. The output σ can be viewed as a random variable which depends on the randomness from the instance $\mathbf{\Phi}$ and the algorithm $\mathcal{A}$. The randomness from the algorithm $\mathcal{A}$ can be represented by a random vector $\mathbf{I}$, named **random internal vector**. Let Φ and I be realizations of $\mathbf{\Phi}$ and $\mathbf{I}$ respectively, and denote the output assignment as $\sigma_0 = \sigma_{\Phi, I}$. Then, we re-randomize the components of the internal vector I *one-by-one*, iteratively. After each re-randomization of a component of I, we run the algorithm again to generate a new output assignment. By doing so, we obtain a sequence of assignments $\sigma_0, \sigma_1, \ldots, \sigma_T$ for the instance Φ, where T is the number of components of the vector I that we have re-randomized.

Next, we verify that the algorithm $\mathcal{A}$ has the following two properties.

1. The algorithm $\mathcal{A}$ is *insensitive to its internal randomness*, in the sense that when one component of the internal vector $\mathbf{I}$ is re-randomized, the output assignment almost remains unchanged. In particular, every two consecutive assignments in the sequence $\sigma_0, \sigma_1, \ldots, \sigma_T$ are close to each other, with distance at most $v_2 - v_1$.

2. The algorithm $\mathcal{A}$ is *dependent on its internal randomness*, in the sense that when all component of the internal vector $\mathbf{I}$ is re-randomized, the output assignment changes a lot. In particular, the first assignment σ_0 and the last assignment σ_T in the sequence are far from each other, with distance greater than v_1.

Intuitively, the first property implies that each assignment in the sequence should be in the same cluster as σ_0 if both σ_0 and that assignment are solutions, since the solutions cannot "jump" over the overlap gap within the solution space. Meanwhile, the second property implies that two ends of the sequence should lie in different clusters if they are solutions, since the distance between them is larger than the diameter of clusters.

Combining these two properties, we are able to pick an assignment σ_{i_0} that falls in the gap, namely, there exists $0 < i_0 \leq T$ such that $v_1 \leq d(\sigma_0, \sigma_{i_0}) \leq v_2$. We denote by α_n the probability that the algorithm $\mathcal{A}$ successfully finds a solution on input $\mathbf{\Phi}$, named **success probability** of the algorithm $\mathcal{A}$ on the random CSP. Then, the probability that both σ_0 and σ_{i_0} are solutions is lower bounded by α_n^2. On the other hand, since the distance $d(\sigma_0, \sigma_{i_0})$ is forbidden due to OGP of the instance, the probability that both σ_0 and σ_{i_0} are solutions is upper bounded by $o(1)$. Therefore, the success probability α_n is upper bounded by $o(1)$.

OGP of random k-XORSAT The natural way to reveal the overlap gap property of the random k-XORSAT problem is to apply the first moment method. Given a random k-XORSAT instance $\mathbf{\Phi} \sim \mathbf{\Phi}_k(n, rn)$, we denote by $Z(d)$ the number of pairs of solutions with distance $d \geq 0$ between them. The expected number of pairs of solutions with distance αn between them, $\mathbb{E}[Z(\alpha n)]$, for any $\alpha \in [0,1]$, is given by the following lemma. The proof is given in Appendix B.

Lemma 1. *Let $k \geq 3$ and $r > 0$. For any $\alpha \in [0,1]$, the expected value of $Z(\alpha n)$ is given by*

$$\mathbb{E}[Z(\alpha n)] = \frac{1}{\sqrt{2\pi n}} \frac{1}{\sqrt{\alpha(1-\alpha)}} f(k,r,\alpha)^n + o(n),$$

where the real-valued function f is defined by

$$f(k,r,\alpha) \equiv \frac{2}{\alpha^\alpha (1-\alpha)^{1-\alpha}} \left(\frac{1+(1-2\alpha)^k}{4} \right)^r .$$

For convenience, we simply assume $\alpha^\alpha(1-\alpha)^{1-\alpha} = 1$ when $\alpha = 0$ or $\alpha = 1$.

If there exists an interval $(u_1, u_2) \subset [0,1]$ such that the expected value $\mathbb{E}[Z(\alpha n)]$ converges to 0 as $n \to \infty$ for any $\alpha \in (u_1, u_2)$, then by Markov's inequality we know that

$$\Pr[\, Z(\alpha n) > 0 \,] \leq \mathbb{E}[Z(\alpha n)] \to 0 \quad \text{as } n \to \infty$$

for $\alpha \in (u_1, u_2)$. It means w.h.p. there is no pair of solutions with distance αn between them, for any $\alpha \in (u_1, u_2)$. Equivalently, we can say that w.h.p. the distance between any pair of solutions is either at most $u_1 n$ or at least $u_2 n$, and conclude that w.h.p. the random k-XORSAT instance $\mathbf{\Phi}$ exhibits OGP with values $v_1 = u_1 n$ and $v_2 = u_2 n$. We can further show the minimal clause density for which such interval exists. The proof is given in Appendix B.

Lemma 2. *For any $k \geq 3$, 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 $\leq u_1 n$ or $\geq u_2 n$ for some $0 \leq u_1 < u_2$. In particular, the value of $r_1(k)$ is given by*

$$r_1(k) = \min_{0 \leq \alpha \leq \frac{1}{2}} \frac{1 + H(\alpha)}{2 - \log_2(1 + (1 - 2\alpha)^k)},$$

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

k	$r_{\text{clt}}(k)$	$r_1(k)$
3	0.818470	0.984516
4	0.772280	0.943723
5	0.701780	0.905812
6	0.637081	0.874349
7	0.581775	0.848314
8	0.534997	0.826470
9	0.495255	0.807862
10	0.461197	0.791788

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

By comparing $r_1(k)$ with the clustering threshold $r_{\text{clt}}(k)$ of random k-XORSAT (Table 1), we can see that $r_1(k)$ is much higher than the clustering threshold $r_{\text{clt}}(k)$, which is also the believed algorithmic threshold. Lemma 2 only guarantees that random k-XORSAT exhibits OGP when the density is high and close to the satisfiability threshold. It is not unique for random k-XORSAT. In some random CSPs, this basic form of OGP starts appearing for clause density much higher than the clustering threshold. Consequently, the approach based on this basic form of OGP can only rule out algorithms for the regimes of relatively high clause densities. For example, random k-SAT exhibits the basic OGP for density $r > (1 - \epsilon)\, 2^k \frac{\ln 2}{2}$ for all $\epsilon > 0$ [DMMZ08], while its clustering threshold is upper bounded by $(1 + o_k(1))\, 2^k \frac{\ln k}{k}$ [AC08].

Variants of OGP One way to improve the OGP-based approach is to consider different variants of OGP. If we can show that a random CSP exhibits a variant of OGP for lower densities, it may open the possibility of ruling out algorithms for lower densities. The most notable variant is *multi-OGP* (m-OGP), which differs from the basic form of OGP in two features. Firstly, m-OGP considers a collection of m solutions, instead of a pair of solutions, and requires that at least a pair in the collection are either close to each other or far from each other. Secondly, it considers an ensemble of instances, rather than a single instance, and the solutions are picked from different instances in the ensemble. The format definition m-OGP is given as follows.

Definition 4. Given a set of instances Ξ of a CSP, we say it exhibits **multi-OGP** (m-OGP) with values v_1 and v_2, where $0 \leq v_1 < v_2$, if for any solutions $\sigma_1, \sigma_2, \ldots, \sigma_m$ of instances

$\Phi_1, \Phi_2, \dots, \Phi_m \in \Xi$ respectively, at least one pair σ_i, σ_j satisfies either $d(\sigma_i, \sigma_j) \le v_1$ or $d(\sigma_i, \sigma_j) \ge v_2$.

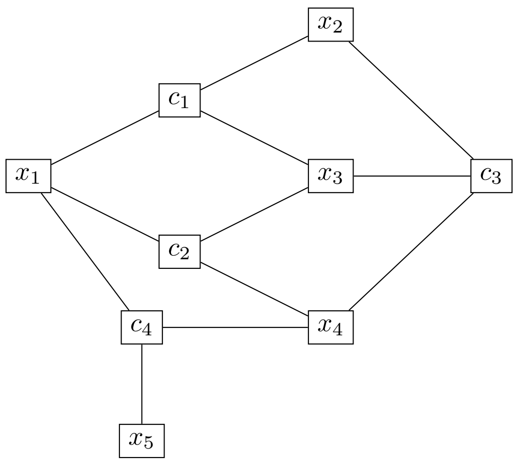
By showing the problem exhibits m-OGP, some recent works successfully ruled out algorithms from several problems for relatively low densities. Here are some examples. [GS17b] showed that random NAE-k-SAT exhibits m-OGP for densities above $(1+o_k(1))2^{k-1}\frac{\ln^2 k}{k}$ for sufficiently large k, which is well below its satisfiability threshold $2^{k-1}\ln 2 - \ln 2/2 - 1/4 - O_k(1)$ [CP12], and thus balanced sequential local algorithms are ruled out for densities. [Wei22] showed that maximum independent set problem exhibits m-OGP for independent sets larger than half-optimal size, and thus low-degree polynomial algorithms are ruled out from maximum independent set problem for larger than half-optimal size, while the best known polynomial-time algorithms from can find an independent set of half-optimal size [Kar76]. [BH22] showed that random k-SAT exhibits m-OGP for density $r > (1 + o_k(1))\kappa^* 2^k \log k/k$ for a universal constant $\kappa^* \approx 4.911$, and therefore low-degree polynomial algorithms are ruled out for those densities. That density threshold matches the largest density $(1-o_k(1))2^k \log k/k$ that the best known algorithm can find solutions [Coj10], up to a constant.

Despite the nice results from the OGP-based approaches with m-OGP, the OGP-based approach still faces a significant challenge. We do not entirely understand the relation between the overlap gap property and the clustering property. It becomes even more complicated with the m-OGP, since it considers an ensemble of instances rather than a single instance. Therefore, it seems that these OGP-based approaches do not guarantee they can rule out algorithms for clause density as low as the clustering threshold.

2.2 Cores of k-XORSAT

Instead of seeking m-OGP of random k-XORSAT instances, we notice that a random k-XORSAT instance contains a sub-instance that also exhibits OGP. We believe we can leverage this fact to improvement the OGP-based approach on random k-XORSAT, in order to rule out some classes of algorithms for clause density as low as the clustering threshold. This sub-instance is called *core instance*, which is the key for the proofs of the satisfiability threshold and the clustering threshold of random k-XORSAT mentioned in Section 1.1. Now, we have a closer look to this sub-instance.

Reducing to sub-instances One can observe that the variables involved in zero or one clause only contribute *extraneous degrees of freedom* to k-XORSAT instances. Suppose we have a variable x_i of degree 1 in an instance Φ. If we remove from Φ the variable x_i and the only clause c_j involving it to obtain a sub-instance Φ' of which we have a solution σ', then we must be able to choose a suitable value for the variable x_i to satisfy the clause c_j and extend σ' to a solution σ of the original instance Φ. Similarly, suppose we have a variable x_i of degree 0 in an instance Φ. If we remove from Φ the variable x_i to obtain a sub-instance Φ' of which we have a solution σ', then we, we can assign whatever value we want to the variable x_i to extend



$$\begin{aligned}
c_1 &: \quad x_1 + x_2 + x_3 = 1\\
c_2 &: \quad x_1 + x_3 + x_4 = 0\\
c_3 &: \quad x_2 + x_3 + x_4 = 0\\
c_4 &: \quad x_1 + x_4 + x_5 = 1
\end{aligned}$$

Figure 2: An example of the factor graph of the random k-XORSAT instance of 5 variables and 4 equations, where $k = 3$. Variable nodes are in circular shape, and equation nodes are in rectangular shape.

σ' to a solution σ of the original instance Φ. Hence, solving the instance Φ can be reduced to solving the sub-instance Φ' obtained by removing variables of degree at most 1 and the only clause involving if exists.

Removing these extraneous degrees of freedom and studying the residual sub-instances could help us understand the combinatorial behaviours of the k-XORSAT instances. In fact, the existing results about the phase transitions of random k-XORSAT rely on analysing the sub-instances called *core instances*, obtained by recursively removing those extraneous degrees of freedom. The concrete definition of *core instances* can be given via the notion of *factor graph.*

Factor graphs A k-XORSAT instance can be represented as a *factor graph.* For a k-XORSAT instance Φ with n variables and m clauses (equations), the **factor graph** G_Φ of Φ is an undirected graph containing two types of nodes: **variable nodes** and **equation nodes**. (The equation nodes are also called **function nodes** and **check nodes** in different contents.) Each variable of the instance Φ is represented by a variable node, and each equation of the instance Φ is represented by an equation node with a tag of its right-hand-side value. For any variable node v and any equation node e, there is an undirected edge (v, e) if and only if the variable corresponding to v is involved in the equation corresponding to e. There is a one-to-one correspondence between variables (equations) and variable nodes (equation nodes). By abuse of notation, the terms *variables (equations)* and *variable nodes (equation nodes)* are interchangeable. For example, we can say a variable has degree d if the corresponding variable node has degree d.

A factor graph is **k-uniform** if all equation nodes have degree k. So, it is obvious that the factor graph of a k-XORSAT instance is k-uniform. For a random k-XORSAT instance $\mathbf{\Phi} \sim \mathbf{\Phi}_k(n, m)$, its factor graph $G_{\mathbf{\Phi}}$ is uniformly distributed over the **ensemble of k-uniform factor graph** $\mathbb{G}_k(n, m)$, which is the set of all possible factor graphs with n variable nodes and m function nodes of degree k.

Cores The **2-core** of a factor graph is the largest subgraph with minimum variable degree at least r. We then define the **core instance** Φ_c (or simply called the **core**) of a k-XORSAT instance Φ to be the sub-instance induced by the 2-core of the factor graph G_Φ. Given an instance Φ, we can obtain its core instance Φ_c by the **peeling algorithm**, which recursively removes variables of degree at most 1 until no more are left [Mol05]. With this algorithm, one can easily prove that the instance Φ is satisfiable if and only if the core instance Φ_c is satisfiable. Therefore, the value of the satisfiability threshold of the whole instance Φ can be determined by that of this core instance Φ_c.

Algorithm 3 Peeling algorithm

```
1: Input: an instance $\Phi$.
2: while There exists one or more variables of degree at most 1. do
3:     Remove from $\Phi$ all variables of degree at most 1 and all clauses containing such variables.
4: end while
5: Output: the resultant instance $\Phi$.
```

It is well known that a non-empty 2-core of random hypergraphs emerges suddenly at a critical threshold of edge density [Coo04, Mol05, Kim04]. As a result, the core instance of the random k-XORSAT instance $\mathbf{\Phi} \sim \mathbf{\Phi}_k(n, rn)$ emerges suddenly at a critical clause density. To be specific, for any $k \geq 3$, there exists a value

$$r_{\text{core}}(k) = \min_{x>0} \frac{x}{k(1-e^{-x})^{k-1}} \tag{4}$$

such that the core instance $\mathbf{\Phi}_c$ of a random k-XORSAT instance $\mathbf{\Phi} \sim \mathbf{\Phi}_k(n, rn)$

- does not contain any variables or clauses w.h.p. for clause density $r < r_{\text{core}}(k)$;
- contains some variables and clauses w.h.p. for clause density $r > r_{\text{core}}(k)$.

This phenomenon plays a pivotal role in the analysis of the clustering threshold of random k-XORSAT. One may notice that the formula of $r_{\text{core}}(k)$ in (4) is exactly same as the formula of r_{clt} in (3). In [IKKM12, AM15], they showed that the random k-XORSAT does not exhibit the clustering property when the core instance $\mathbf{\Phi}_c$ is empty in the regime of clause density $r < r_{\text{core}}(k)$, and it exhibits the clustering property when the core instance $\mathbf{\Phi}_c$ is non-empty in the regime of clause density $r > r_{\text{core}}(k)$. Therefore, we have $r_{\text{clt}}(k) = r_{\text{core}}(k)$. In short, the clustering phenomenon of random k-XORSAT is a consequence of the existence of non-empty core instance for clause density $r > r_{\text{core}}(k)$.

Properties of core instances We selectively capture some properties of the core instance from [MM09], in the following theorem, to provide a rough idea on how the core instance looks like. The following theorem includes the number of variables and the degree distribution of the variables in the core instance.

Theorem 4 ([MM09]). *For any $k \geq 3$, there exists $r_{\text{core}}(k) > 0$ given by*

$$r_{\text{core}}(k) = \min_{x>0} \frac{x}{k(1-e^{-x})^{k-1}}$$

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

(a) When $r < r_{\text{core}}(k)$, w.h.p. the core instance $\mathbf{\Phi}_c$ is empty.

(b) When $r > r_{\text{core}}(k)$, w.h.p. the core instance $\mathbf{\Phi}_c$ have $V(k,r)n + o(n)$ variables, 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 variables 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

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

(c) Conditioning on the number of variables $V(k,r)n+o(n)$ and the variable degree distribution $\widehat{\Lambda} = (\widehat{\Lambda}_0, \widehat{\Lambda}_1, \ldots, \widehat{\Lambda}_m)$, the core instance $\mathbf{\Phi}_c$ is distributed according to the ensemble containing all possible k-XORSAT instances of $V(k,r)n + o(n)$ variables and variable degree distribution $\widehat{\Lambda}$.

Surprisingly, the proof in [AM15] actually tells us that the w.h.p. the core instance of a random k-XORSAT instance exhibit the overlap gap property.

Lemma 3 ([AM15]). *For $k \geq 3$ and $r_{\text{core}}(k) < r < r_{\text{sat}}(k)$, 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 $\epsilon(k,r)n$.*

Furthermore, the OGP of the core instance $\mathbf{\Phi}_c$ with the value $v_1 = o(n)$ and $v_2 = \epsilon(k,r)n$ precisely describes the clustering structure of the solution space $\mathcal{S}(\mathbf{\Phi})$ of the entire instance $\mathbf{\Phi}$, since it tells us how to partition the solution space $\mathcal{S}(\mathbf{\Phi})$. This partitioning of the solution space $\mathcal{S}(\mathbf{\Phi})$ is exactly the same one used to determine the clustering threshold of random k-XORSAT in the proofs in [IKKM12, AM15]. To obtain the partitioning of the solution space $\mathcal{S}(\mathbf{\Phi})$, we can first partition the solution space $\mathcal{S}(\mathbf{\Phi}_c)$ of the core instance $\mathbf{\Phi}_c$ into some subsets, called **core clusters**, 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 $\epsilon(k,r)n$. To be specific, we first define an equivalence relation $\sim$ on the solution space $\mathcal{S}(\mathbf{\Phi}_c)$ by writing $\sigma_c \simeq \sigma_c'$ if and only if $d(\sigma_c, \sigma_c') = o(n)$, for any $\sigma_c, \sigma_c' \in \mathcal{S}(\mathbf{\Phi}_c)$. Since $2 \cdot o(n) < \epsilon(k,r)n$ for sufficiently large n, we can partition the solution space $\mathcal{S}(\mathbf{\Phi}_c)$ by the n_c equivalence classes of the relation $\sim$, that is, $\mathcal{S}_{c,1} \sqcup \mathcal{S}_{c,2} \sqcup ... \sqcup \mathcal{S}_{c,n_c} = \mathcal{S}(\Phi_c)$, where $\sqcup$ is the disjoint union and

$\mathcal{S}_{c,1}, \mathcal{S}_{c,2}, ..., \mathcal{S}_{c,n_c}$ are the equivalence classes. Moreover, we have

$$d(\sigma_c, \sigma_c') = o(n) \qquad \text{if } \sigma_c, \sigma_c' \in \mathcal{S}_{c,i}, \quad \text{and} \tag{5}$$

$$d(\sigma_c, \sigma_c') \geq \epsilon(k,r)n \qquad \text{if } \sigma_c \in \mathcal{S}_{c,i},\ \sigma_c' \in \mathcal{S}_{c,j} \text{ and } \mathcal{S}_{c,i} \neq \mathcal{S}_{c,j}. \tag{6}$$

Based on this partitioning of $\mathcal{S}(\mathbf{\Phi}_c)$, we can partition the solution space $\mathcal{S}(\mathbf{\Phi})$ of the entire instance $\mathbf{\Phi}$ into clusters, by setting

$$\mathcal{S}(\Phi) = \bigsqcup_{i=1}^{n_c} \mathcal{S}_i \quad \text{and} \quad \mathcal{S}_i = \{\sigma \in \mathcal{S}(\Phi) : \pi(\sigma) \in \mathcal{S}_{c,i}\} \quad \text{for } i = 1, 2, ..., n_c, \tag{7}$$

where π is defined to be the **projection** mapping assignments for $\mathbf{\Phi}$ to assignments for $\mathbf{\Phi}_c$ by removing all variables not in the core instance. Then, we can prove that these subsets are well-separated from each other, by the following lemma.

Lemma 4. *Let $k \geq 3$ and $r_{\text{core}}(k) < r < r_{\text{sat}}(k)$. 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.*

(a) 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 $Q = 1 - \exp(-krQ^{k-1})$.

(b) 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 4. 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 3. 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 (7) and (5), 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 4, the number of variables outside the core instance is given by $(1 - V(k,r))n + o(n)$. Hence, we have

$$\begin{aligned} d(\sigma,\tau) &\le (1-V(k,r))n + o(n) \\ &= \left(\exp(-krQ^{k-1}) + krQ^{k-1}\exp(-krQ^{k-1})\right) n + o(n) \end{aligned}$$

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 (7) and (6), we have $d(\sigma_c, \sigma'_c) \ge \epsilon(k,r)n$. Therefore, we have $d(\sigma,\sigma') \ge d(\pi(\sigma),\pi(\sigma')) = d(\sigma_c,\sigma'_c) \ge \epsilon(k,r)n$. □

2.3 Our improvement on OGP-based approach

We introduce a new improvement of the OGP-based approach, in order to reach our main results. This new approach is similar to the one described in Section 2.1, except that we utilize the overlap gap property of a sub-instance, instead of the overlap gap property of the entire instance. In particular, for random k-XORSAT, we utilize the overlap gap property of the non-empty core instance $\mathbf{\Phi}_c$ for density $r_{\text{core}}(k) < r < r_{\text{sat}}(k)$. Recall from Lemma 3, we know that the core instance $\mathbf{\Phi}_c$ exhibits OGP with $v'_1 = o(n)$ and $v'_2 = \epsilon(k,r)n$ for some constant $\epsilon(k,r) > 0$, for clause density $r_{\text{core}}(k) < r < r_{\text{sat}}(k)$.

Same as the approach described in Section 2.1, we first sample the random k-XORSAT instance $\mathbf{\Phi}$ and the random internal vector $\mathbf{I}$. With the realizations Φ and I of them, we run the algorithm to give the first output assignment σ_0. Then, we re-randomize the components of the internal vector I one-by-one iteratively. After each re-randomization of a component of I, we run the algorithm again to generate a new assignment. This gives us a sequence of assignments $\sigma_0, \sigma_1, \ldots, \sigma_T$. Now, we remove all variables not in the core instance of Φ from each assignment in the sequence, and this results in a sequence of assignments $\sigma'_0, \sigma'_1, \ldots, \sigma'_T$ for the core instance Φ_c. From Theorem 4, the number of variables removed is upper bounded by $n - V(k,r)n + o(n) = \mu(k,r)n + o(n)$.

We then apply the same argument as before to the sequence of assignments $\sigma'_0, \sigma'_1, \ldots, \sigma'_T$ for the core instance Φ_c, instead of the sequence of assignments $\sigma_0, \sigma_1, \ldots, \sigma_T$ for the whole instance Φ. Assume the algorithm is *insensitive to its internal randomness* with respect to the core instance in the sense that $d(\sigma'_i, \sigma'_{i+1}) < v'_2 - v'_1$ for all i. We further assume the algorithm is strictly $2\mu(k,r)$-free so that $\mathbb{E}\left[d(\sigma'_0, \sigma'_T)\right] > v'_1 = o(n)$, which means the algorithm is *dependent on its internal randomness*. Then, we can pick a core assignment σ'_{i_0} from the sequence such that $v'_1 < d(\sigma'_0, \sigma'_{i_0}) < v'_2$. Then, the probability that both σ'_0 and σ'_{i_0} are core solutions is lower bounded by the probability that both σ_0 and σ_{i_0} are solutions, which is further lower bounded by α_n^2, where α_n is the success probability of the algorithm. On the other hand, since the distance $d(\sigma'_0, \sigma'_{i_0})$ is forbidden due to the OGP of the core instance, the probability that both σ'_0 and σ'_{i_0} are solutions is upper bounded by $o(1)$. Hence, the success probability α_n is upper bounded by $o(1)$.

If we apply the original OGP-based approach with Lemma 2, we are able to show that the sequential local algorithms fail to solve the random k-XORSAT problem for clause density above $r_1(k)$. Unfortunately, the threshold $r_1(k)$ is much higher than the believed algorithmic threshold $r_{\text{clt}}(k)$. With our new variant of OGP-based approach, we can obtain the same result for clause densities as low as $r_{\text{core}}(k)$. Since $r_{\text{core}}(k) = r_{\text{clt}}(k)$ for random k-XORSAT, we can conclude that the algorithms fail in finding a solution for clause densities as low as the clustering threshold $r_{\text{clt}}(k)$. This opens a new possibility to improve other results which use the OGP-based approaches on other random constraint satisfaction problems.

3 Ruling out sequential local algorithms

In this section, we will give the proof of Theorem 1. In particular, we will show that for any $k \geq 3$ and $r \in (r_{\text{clt}}(k), r_{\text{sat}}(k))$, if the τ-decimation algorithm $\texttt{DEC}_\tau$ is strictly $2\mu(k,r)$-free, then w.h.p. $\texttt{DEC}_\tau$ can find solutions of a random k-XORSAT instance.

3.1 Preparation of OGP-based approach

In this section, we introduce some notions and obtain some preliminary results needed by our OGP-based approach to prove the main results. Recall the outline of OGP-based approach in Section 2.1. We have to construct a sequence of output assignments by running the τ-decimation algorithm $\texttt{DEC}_\tau$ for multiple times. Then, we have to show that $\texttt{DEC}_\tau$ is *insensitive to its internal randomness* and *dependent on its internal randomness.* The former can be done by reusing the result from [GS17b]. The latter can be done by introducing the notion of *freeness.*

3.1.1 Sequence of output assignments

The random k-XORSAT instance $\mathbf{\Phi}$ is a random variable, and the τ-decimation algorithm $\texttt{DEC}_\tau$ is a randomized algorithm. Therefore, the assignment output by the τ-decimation algorithm $\texttt{DEC}_\tau$ 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 [GS17b] in the following discussion.

The order of variables can be represented by a random vector $\mathbf{Z} = (\mathbf{Z}_1, \mathbf{Z}_2, \ldots, \mathbf{Z}_n)$ 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 OGP-based approach. 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 is a mapping that maps 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 $\{\sigma'_i = \pi(\sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{U^i}})\}_{i=0}^n$.

3.1.2 Insensitive to internal randomness

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 3.1.1. 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 [GS17b] proved the insensitivity of the τ-decimation algorithm in their works, using the notion of *influence range*. Although their works [GS17b] 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 5. 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 5 (From [GS17b]). *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 6 (From [GS17b]). *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 5). They further showed that w.h.p. the size of the influence range of variables is sublinear for all variables (Lemma 6). Note that in the original statement of Lemma 6 in [GS17b], 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 distance 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 7. *For any $\xi \in (0,1)$ and sufficiently large n,*

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

Proof. Fix an arbitrary $i \in [n]$. We know that $\mathbf{U}^{\mathbf{i-1}}_j = \mathbf{U}^{\mathbf{i}}_j$ for all $j \neq i$, and $\mathbf{U}^{\mathbf{i-1}}_i \neq \mathbf{U}^{\mathbf{i}}_i$. By Lemma 5, 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

$d(\sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{U}^{\mathbf{i-1}}}, \sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{U}^{\mathbf{i}}}) \geq n^{1/6}$ for some $i \in [n]$, we have $|\mathcal{IR}_{x_i}| \geq n^{1/6}$. Hence, by Lemma 6 we have

$$\begin{aligned}\Pr\left[d(\sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{U}^{\mathbf{i-1}}}, \sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{U}^{\mathbf{i}}}) \geq n^{1/6} \text{ for some } i \in [n]\right] &\leq \Pr\left[|\mathcal{IR}_{x_i}| \geq n^{1/6} \text{ for some } i \in [n]\right] \\ &\leq \Pr\left[\max_{1\leq i\leq n} |\mathcal{IR}_{x_i}| \geq n^{1/6}\right] \\ &\leq \exp\left(-\ln n(\ln\ln n)^{\xi/4}\right).\end{aligned}$$

□

3.1.3 Dependent on internal randomness, and notion of freeness

Recall the definition of *free steps*. An iteration of the τ-decimation algorithm $\texttt{DEC}_\tau$ is called a free step if the local rule τ gives the value 1/2 in that iteration. In such 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. So, the value chosen by the algorithm is completely dependent on the corresponding entry in the random internal vector $\mathbf{I}$. The more the number of free steps is, the more dependent on the internal randomness the algorithm is. Therefore, the number of free steps executed by $\texttt{DEC}_\tau$ provides a lower bound that measures $\texttt{DEC}_\tau$'s dependency on its internal randomness.

Now, we recall the definition of a τ-decimation algorithm being δ-free. A τ-decimation algorithm $\texttt{DEC}_\tau$ 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 $\mathbf{\Phi}$. Then, we can say that the more free the τ-decimation algorithm $\texttt{DEC}_\tau$ is, the more dependent on the internal randomness it is.

3.2 Proof of Theorem 1

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 of $\mathbf{\Phi}$. 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 3.1.1. Now, we consider the sequence of output assignments $\{\sigma_i = \sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{U}^{\mathbf{i}}}\}_{i=0}^n$ generated by the procedure in Section 3.1.1. We first prove that if the algorithm $\texttt{DEC}_\tau$ is strictly δ-free, then the expected distance $d(\sigma_0, \sigma_n)$ between the first and the last assignments in the sequence is at least $\delta n/2$.

Lemma 8. *If the τ-decimation algorithm $\mathtt{DEC}_\tau$ is δ-free on the random k-XORSAT instance $\mathbf{\Phi} \sim \mathbf{\Phi}_k(n, rn)$ for some $\delta > 0$, then we have $\mathbb{E}[d(\sigma_0, \sigma_n)] \geq (\delta/2)n + o(n)$.*

Proof. Suppose $D \subseteq [n]$ is the subset of indices of iterations that are free steps, namely,

$$D = \{i \in [n] : \text{The } i\text{-th iteration is a free step}\}.$$

Note that $\sigma_0 = \sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{U^0}} = \sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{V}}$ and $\sigma_n = \sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{U^n}} = \sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{W}}$ and thus

$$d(\sigma_0, \sigma_n) = d(\sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{U^0}}, \sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{U^n}}) = d(\sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{V}}, \sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{W}})$$

Since we can write $d(\sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{V}}, \sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{W}}) = \sum_{i=1}^n \mathbb{1}\left(\sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{V}}(x_{s(i)}) \neq \sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{W}}(x_{s(i)})\right)$, we have

$$\begin{aligned} d(\sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{V}}, \sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{W}}) &= \sum_{i=1}^n \mathbb{1}\left(\sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{V}}\left(x_{s(i)}\right) \neq \sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{W}}\left(x_{s(i)}\right)\right) \\ &\geq \sum_{i \in D} \mathbb{1}\left(\sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{V}}\left(x_{s(i)}\right) \neq \sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{W}}\left(x_{s(i)}\right)\right). \end{aligned}$$

In free steps, the local rule τ gives the value $1/2$ to the decimation algorithm. Therefore, for any $i \in D$, $\sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{V}}\left(x_{s(i)}\right) \neq \sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{W}}\left(x_{s(i)}\right)$ if and only if either $\mathbf{V}_i < 1/2 < \mathbf{W}_i$ or $\mathbf{W}_i < 1/2 < \mathbf{V}_i$. Therefore, we have

$$\sum_{i \in D} \mathbb{1}\left(\sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{V}}\left(x_{s(i)}\right) \neq \sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{V}}\left(x_{s(i)}\right)\right) = \sum_{i \in D} \mathbb{1}\left(\mathbf{V}_i < 1/2 < \mathbf{W}_i \text{ or } \mathbf{W}_i < 1/2 < \mathbf{V}_i\right)$$

Note that the random variables $\mathbf{V}_1, \mathbf{V}_2, \ldots, \mathbf{V}_n, \mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_n$ are i.i.d. over uniform distributions on $[0, 1]$. Thus, $\sum_{i \in D} \mathbb{1}\left(\mathbf{V}_i < 1/2 < \mathbf{W}_i \text{ or } \mathbf{W}_i < 1/2 < \mathbf{V}_i\right)$ is distributed over the binomial distribution $B(|D|, 1/2)$ with parameters $|D|$ and $1/2$.

Assume that the algorithm $\mathtt{DEC}_\tau$ is δ-free on the random k-XORSAT instance $\mathbf{\Phi} \sim \mathbf{\Phi}_k(n, rn)$ for some $\delta > 0$, which implies that w.h.p. $|D| \geq \delta n$. Note that D only depends on $\mathbf{\Phi}$ and $\mathbf{Z}$. Hence, we have

$$\begin{aligned} \mathbb{E}\left[d(\sigma_0, \sigma_n)\right] &= \mathbb{E}_{\mathbf{\Phi},\mathbf{Z},\mathbf{V},\mathbf{W}}\left[\sum_{i \in D} \mathbb{1}\left(\mathbf{V}_i < 1/2 < \mathbf{W}_i \text{ or } \mathbf{W}_i < 1/2 < \mathbf{V}_i\right)\right] \\ &= \mathbb{E}_{\mathbf{V},\mathbf{W}}\mathbb{E}_{\mathbf{\Phi},\mathbf{Z}}\left[\sum_{i \in D} \mathbb{1}\left(\mathbf{V}_i < 1/2 < \mathbf{W}_i \text{ or } \mathbf{W}_i < 1/2 < \mathbf{V}_i\right)\right] \\ &= (1/2) \cdot \mathbb{E}\left[|D|\right] \\ &\geq (\delta/2)n + o(n) \end{aligned}$$

□

Next, we will show that, if the τ-decimation algorithm is "free enough" (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 9. *For any $k \geq 3$ and $r \in (r_{\text{clt}}(k), r_{\text{sat}}(k))$, 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*

$$\left| d(\pi(\sigma_0), \pi(\sigma_{i_0})) - \frac{1}{2}\epsilon' n \right| < \frac{1}{4}\epsilon' n, \tag{8}$$

where $\epsilon(k,r)$ is given in Lemma 3.

Proof. Assume the τ-decimation algorithm $\texttt{DEC}_\tau$ is strictly $2\mu(k,r)$-free on the random instance $\mathbf{\Phi} \sim \mathbf{\Phi}_k(n, rn)$. Then, there exists $\delta > 2\mu(k,r)$ such that $\texttt{DEC}_\tau$ is δ-free on $\mathbf{\Phi}$.

We first show that there exists $0 \leq i_0 \leq n$ such that the expected value of $d(\pi(\sigma_0), \pi(\sigma_{i_0}))$ is close to $\frac{1}{2}\epsilon' n$ for some $\epsilon' < \epsilon(k,r)$. Consider the sequence $\{\mathbb{E}\left[d(\pi(\sigma_0), \pi(\sigma_i))\right]\}_{i=0}^n$ in which the first item is

$$\mathbb{E}\left[d(\pi(\sigma_0), \pi(\sigma_0))\right] = 0. \tag{9}$$

By Lemma 8, we know that $\mathbb{E}\left[d(\sigma_0, \sigma_n)\right] \geq (\delta/2)n + o(n)$. Moreover, by Theorem 4, we know that w.h.p. there are $\mu(k,r)n + o(n)$ variables not in the core instance $\mathbf{\Phi}_c$. Note that the projection function π only remove variables not in the core instance $\mathbf{\Phi}_c$. Therefore, we have $d(\pi(\sigma_0), \pi(\sigma_n)) \geq d(\sigma_0, \sigma_n) - n^*$ where n^* is the number of variables not in the core instance $\mathbf{\Phi}_c$. Therefore, we have

$$\mathbb{E}\left[d(\sigma_0, \sigma_n)\right] \leq \mathbb{E}\left[d(\pi(\sigma_0), \pi(\sigma_n))\right] + \mathbb{E}\left[n^*\right] \leq \mathbb{E}\left[d(\pi(\sigma_0), \pi(\sigma_n))\right] + \mu(k,r)n + o(n)$$

By re-arranging the terms, we have

$$\mathbb{E}\left[d(\pi(\sigma_0), \pi(\sigma_n))\right] \geq (\delta/2 - \mu(k,r))n + o(n) \tag{10}$$

with $\delta/2 - \mu(k,r) > 0$. From Lemma 7, we know that with probability $1 - \exp(-\ln n (\ln \ln n)^{\xi/4})$ we have

$$d(\sigma_{i-1}, \sigma_i) < n^{1/6} \quad \text{for all } i \in [n]. \tag{11}$$

By the triangle inequality of the metric d and the linearity of the expectation, we know that for

$1 \leq i \leq n$ the difference of two consecutive expected values in the sequence is

$$\begin{aligned}
&\mathbb{E}\left[d(\pi(\sigma_0), \pi(\sigma_i))\right] - \mathbb{E}\left[d(\pi(\sigma_0), \pi(\sigma_{i-1}))\right] \\
&\leq \mathbb{E}\left[d(\pi(\sigma_0), \pi(\sigma_{i-1}))\right] + \mathbb{E}\left[d(\pi(\sigma_{i-1}), \pi(\sigma_i))\right] - \mathbb{E}\left[d(\pi(\sigma_0), \pi(\sigma_{i-1}))\right] \\
&= \mathbb{E}\left[d(\pi(\sigma_{i-1}), \pi(\sigma_i))\right] \\
&\leq \mathbb{E}\left[d(\sigma_{i-1}, \sigma_i)\right] \\
&\leq n^{1/6} + o(1). \qquad (12)
\end{aligned}$$

Combining (9), (10) and (12), we know that there exists $0 \leq i_0 \leq n$ and $0 < \epsilon' < \min\{\delta/2 - \mu(k,r), \epsilon(k,r)\}$ such that

$$\mathbb{E}\left[d(\pi(\sigma_0), \pi(\sigma_{i_0}))\right] \in \left[\frac{1}{2}\epsilon' n,\ \frac{1}{2}\epsilon' n + n^{1/6}\right]. \qquad (13)$$

Next, we prove that $d(\pi(\sigma_0), \pi(\sigma_{i_0}))$ concentrates around its mean. Given two vectors $A \in \{0,1\}^n$ and $B \in \{0,1\}^n$, we write $A \cdot B = (A_1, \ldots, A_n, B_1, \ldots, B_n)$ and $A \oplus_i B = (A_1, \ldots, A_i, B_{i+1}, \ldots, B_n)$ for $i \in [n]$. A function $f : D_1 \times D_2 \times \ldots D_n \to \mathbb{R}$ satisfies the **bounded differences** property if there exist $c_1, c_2, \ldots, c_n \in \mathbb{R}$ such that for any $x_1 \in D_1$, $x_2 \in D_2$, $\ldots$, $x_n \in D_n$, $y_i \in D_i$ and $i \leq [n]$,

$$|f(x_1, ..., x_{i-1}, x_i, x_{i+1}, ..., x_n) - f(x_1, ..., x_{i-1}, y_i, x_{i+1}, ..., x_n)| \leq c_i.$$

Note that $\mathbf{U^{i_0}} = \mathbf{W} \oplus_{i_0} \mathbf{V}$. For arbitrary instance Φ and ordering vector Z, we have

$$d(\pi(\sigma_{\Phi,Z,\mathbf{U^0}}), \pi(\sigma_{\Phi,Z,\mathbf{U^{i_0}}})) = d(\pi(\sigma_{\Phi,Z,\mathbf{V}}), \pi(\sigma_{\Phi,Z,\mathbf{W}\oplus_{i_0}\mathbf{V}})).$$

So, given a random instance $\mathbf{\Phi}$ and a random ordering vector $\mathbf{Z}$, we can write $d(\pi(\sigma_0), \pi(\sigma_{i_0}))$ as a function $f : \{0,1\}^{2n} \to \mathbb{R}$ on variables $\mathbf{V} \cdot \mathbf{W}$ given by $f(\mathbf{V} \cdot \mathbf{W}) = d(\pi(\sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{V}}), \pi(\sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{W}\oplus_{i_0}\mathbf{V}}))$. Conditioning on (11), we can verify that f satisfies bounded differences property with $c_i = 2n^{1/6}$ for $i \in [n]$, and thus, by McDiarmid's inequality, we have

$$\begin{aligned}
&\Pr\left[\left|d(\pi(\sigma_0), \pi(\sigma_{i_0})) - \frac{1}{2}\epsilon' n\right| \geq \frac{1}{4}\epsilon' n\right] \\
&\leq \Pr\left[\left|d(\pi(\sigma_0), \pi(\sigma_{i_0})) - \mathbb{E}\left[d(\pi(\sigma_0), \pi(\sigma_{i_0}))\right]\right| \geq \frac{1}{4}\epsilon' n - n^{1/6}\right] \\
&= \Pr\left[\left|f(V \cdot W) - \mathbb{E}\left[f(V \cdot W)\right]\right| \geq \frac{1}{4}\epsilon' n - n^{1/6}\right] \\
&\leq 2\exp\left(-\frac{2\left(\frac{1}{4}\epsilon' n - n^{1/6}\right)^2}{(n+i_0)n^{1/6}}\right) \\
&\leq 2\exp\left(-\frac{1}{8}n^{5/6} + o(n^{5/6})\right).
\end{aligned}$$

Since the condition (11) holds with probability $1 - \exp(-\ln n (\ln\ln n)^{\xi/4}) \to 1$ as $n \to \infty$, the inequality (8) holds with high probability. $\square$

Now we show that the probability of both the output assignments $\sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{U^0}}$ and $\sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{U^{i_0}}}$ being solutions for the instance $\mathbf{\Phi}$ is lower bounded by α_n^2.

Lemma 10. *For any $i \in [n]$, we have*

$$\Pr\left[\, \sigma_0 \in \mathcal{S}(\Phi) \text{ and } \sigma_i \in \mathcal{S}(\Phi) \,\right] \geq \alpha_n^2. \tag{14}$$

Proof. Fix an arbitrary $i \in [n]$. Note that we have $\mathbf{U^0} = (\mathbf{V}_1, \mathbf{V}_2, \ldots, \mathbf{V}_n)$ and $\mathbf{U^i} = (\mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_i, \mathbf{V}_{i+1}, \ldots, \mathbf{V}_n)$, where $\mathbf{V}_1, \mathbf{V}_2, \ldots, \mathbf{V}_n, \mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_n$ are uniformly distributed over $[0,1]$, independently. Conditioning on $\mathbf{\Phi}, \mathbf{Z}, \mathbf{V}_{i+1}, \ldots, \mathbf{V}_n$, the assignment $\sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{U^0}}$ only depends on $\mathbf{V}_1, \ldots, \mathbf{V}_i$, and the assignment $\sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{U^i}}$ only depends on $\mathbf{W}_1, \ldots, \mathbf{W}_i$, and we have

$$\begin{aligned}
&\mathbb{E}_{\mathbf{V}_1,\ldots,\mathbf{V}_i,\mathbf{W}_1,\ldots,\mathbf{W}_i}\left[\, \mathbb{1}(\sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{U^0}} \in \mathcal{S}(\Phi)) \cdot \mathbb{1}(\sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{U^i}} \in \mathcal{S}(\Phi)) \,\right] \\
&= \left(\mathbb{E}_{\mathbf{V}_1,\ldots,\mathbf{V}_i} \mathbb{1}(\sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{U^0}} \in \mathcal{S}(\Phi))\right) \cdot \left(\mathbb{E}_{\mathbf{W}_1,\ldots,\mathbf{W}_i} \mathbb{1}(\sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{U^i}} \in \mathcal{S}(\Phi))\right) \\
&= \left(\mathbb{E}_{\mathbf{V}_1,\ldots,\mathbf{V}_i} \mathbb{1}(\sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{V}} \in \mathcal{S}(\Phi))\right)^2
\end{aligned}$$

Therefore, by Jensen's inequality, we have

$$\begin{aligned}
&\Pr_{\mathbf{\Phi},\mathbf{Z},\mathbf{V},\mathbf{W}}\left[\, \sigma_0 \in \mathcal{S}(\Phi) \text{ and } \sigma_i \in \mathcal{S}(\Phi) \,\right] \\
&= \Pr_{\mathbf{\Phi},\mathbf{Z},\mathbf{V},\mathbf{W}}\left[\, \sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{U^0}} \in \mathcal{S}(\Phi) \text{ and } \sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{U^i}} \in \mathcal{S}(\Phi) \,\right] \\
&= \mathbb{E}_{\mathbf{\Phi},\mathbf{Z},\mathbf{V}_{i+1},\ldots,\mathbf{V}_n} \mathbb{E}_{\mathbf{V}_1,\ldots,\mathbf{V}_i,\mathbf{W}_1,\ldots,\mathbf{W}_i}\left[\, \mathbb{1}(\sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{U^0}} \in \mathcal{S}(\Phi)) \cdot \mathbb{1}(\sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{U^i}} \in \mathcal{S}(\Phi)) \,\right] \\
&= \mathbb{E}_{\mathbf{\Phi},\mathbf{Z},\mathbf{V}_{i+1},\ldots,\mathbf{V}_n} \left(\mathbb{E}_{\mathbf{V}_1,\ldots,\mathbf{V}_i} \mathbb{1}(\sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{V}} \in \mathcal{S}(\Phi))\right)^2 \\
&\geq \left(\mathbb{E}_{\mathbf{\Phi},\mathbf{Z},\mathbf{V}_{i+1},\ldots,\mathbf{V}_n} \mathbb{E}_{\mathbf{V}_1,\ldots,\mathbf{V}_i} \mathbb{1}(\sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{V}} \in \mathcal{S}(\Phi))\right)^2 \\
&= \left(\Pr_{\mathbf{\Phi},\mathbf{Z},\mathbf{V}}\left[\, \sigma_{\mathbf{\Phi},\mathbf{Z},\mathbf{V}} \in \mathcal{S}(\Phi) \,\right]\right)^2 \\
&= \alpha_n^2.
\end{aligned}$$

$\square$

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

Proof of Theorem 1. From Lemma 9, there exists $0 \leq i_0 \leq n$ and $0 < \epsilon' < \epsilon(k,r)$ such that (8) holds with high probability. So, 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 $\Pr[\,\mathcal{A}\,] = 1 - o(1)$. On the other hand, from Lemma 10, (14) holds for $i = i_0$ with probability at least α_n^2. So, we denote by $\mathcal{B}$ the event of

$$\sigma_0 \in \mathcal{S}(\Phi) \text{ and } \sigma_{i_0} \in \mathcal{S}(\Phi),$$

and $\Pr[\mathcal{B}] \geq \alpha_n^2$. Note that we have

$$\begin{aligned}
\Pr[\mathcal{A} \cap \mathcal{B}] &= 1 - \Pr[(\text{Not } \mathcal{A}) \cup (\text{Not } \mathcal{B})] \\
&\geq 1 - \Pr[\text{Not } \mathcal{A}] - \Pr[\text{Not } \mathcal{B}] \\
&= \Pr[\mathcal{A}] - (1 - \Pr[\mathcal{B}]) \\
&\geq 1 - o(1) - (1 - \alpha_n^2) \\
&= \alpha_n^2 - o(1).
\end{aligned}$$

Thus, we have $\alpha_n \leq \Pr[\mathcal{A} \cap \mathcal{B}]^{1/2} + o(1)$.

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 $((1/4)\epsilon' n, (3/4)\epsilon' n) \subsetneq (o(n), \epsilon(k,r)n)$, which takes place with probability at most $o(1)$ by Lemma 3. So, we have $\Pr[\mathcal{A} \cap \mathcal{B}] \leq o(1)$, and thus $\alpha_n \leq o(1)$. $\square$

4 Two sequential local algorithms

In this section, we will discuss the freeness of two sequential local algorithms, which leads us to the proofs for Theorem 2 and Theorem 3.

4.1 Preliminaries

Degree profile The distribution of the degrees of nodes in a factor graph can be described by *degree profile*, which plays an important role in our analysis. Given a factor graph G, let n_i be the number of variable nodes of degree i for all i. Similarly, let m_i be the number of equations nodes of degree i for all i. Furthermore, let $\widehat{n}$ be the total number of variable nodes and $\widehat{m}$ be the total number of equation nodes. It is obvious that $\widehat{n} = n$ and $\widehat{m} = m$ for the factor graph of a k-XORSAT instance, but keep in mind that the total number $\widehat{n}$ of variable nodes and the total number $\widehat{m}$ of equation nodes decrease during the process of the algorithm. The **degree distribution of variable nodes** is given by the sequence $\Lambda = \{\Lambda_i\}_{i \geq 0}$, where $\Lambda_i = n_i/\widehat{n}$, and the **degree distribution of equation nodes** is given by the sequence $P = \{P_i\}_{i \geq 0}$, where $P_i = m_i/\widehat{m}$. Then, the **degree profile** of the factor graph G is given by (Λ, P). Sometimes, the degree distributions Λ and P can also be represented by the polynomials $\Lambda(x) = \sum_{i \geq 0} \Lambda_i x^i$ and $P(x) = \sum_{i \geq 0} P_i x^i$, respectively. With this representation, we can write $\sum_{i \geq 1} i\Lambda_i = \Lambda'(1)$ and $\sum_{i \geq 1} iP_i = P'(1)$.

Given the factor graph G of a random k-XORSAT instance $\mathbf{\Phi} \sim \mathbf{\Phi}_k(n, rn)$, G is uniformly distributed over the ensemble of k-uniform factor graph $\mathbb{G}_n(k, rn)$. It is clear that the degree distribution of equation nodes is given by $P(x) = x^k$. The degree distribution of variable nodes is also known to converge in distribution to independent Poisson random variables with mean kr. Precisely speaking, w.h.p. for any $0 \leq i \leq rn$, we have

$$\Lambda_i = e^{-kr}\frac{(kr)^i}{i!} + o(1).$$

We can also describe the degree profile from *edge perspective*. The **edge perspective degree distribution of variable nodes** is given by $\lambda = \{\lambda_i\}_{i\geq 1}$, where $\lambda_i = i\Lambda_i / \sum_{j\geq 1} j\Lambda_j$, and the **edge perspective degree distribution of equation nodes** is given by $\rho = \{\rho_i\}_{i\geq 1}$, where $\rho_i = iP_i / \sum_{j\geq 1} jP_j$. Then, the **edge perspective degree profile** of the factor graph G is given by (λ, ρ). Similar to Λ and P, the edge perspective degree distribution λ and ρ can be written as the polynomials $\lambda(x) = \sum_{i\geq 1} \lambda_i x^{i-1}$ and $\rho(x) = \sum_{i\geq 1} \rho_i x^{i-1}$, respectively.

The ensemble of factor graphs with prescribed degree profile is called the **ensemble of degree constrained factor graphs** $\mathbb{D}_n(\Lambda, P)$, which is the set of all factor graphs of n variable nodes with degree profile (Λ, P) with the uniform distribution. Note that the number m of the function nodes is restricted to satisfy the equation $\Lambda'(1)n = P'(1)m$.

Local tree-like structure In a factor graph G, we can define the **length** of a path $(v_1, v_2, ..., v_l)$ to be the number of edges in the path. Then, the **distance** between two nodes is defined to be the length of the shortest path between them. By convention, we set the length to be $+\infty$ between two nodes if there is no path connecting them. With this notion of *distance*, we can define the *local neighborhood* of a node. Given a factor graph G, the **local neighborhood** (or simply **neighborhood**) $B_G(x, R)$ of a node x of radius $R \geq 0$ is defined to be the subgraph of G induced by all nodes of distances at most R from x and all edges between those nodes. The local neighborhood $B_G(x, R)$ also represents an XORSAT instance with the variables and clauses inside the neighborhood.

The local neighborhood of a variable in a random k-uniform factor graph looks like a tree. In particular, it looks similar to a *random R-generation tree.* For any non-negative even number $R \geq 0$, the **R-generation tree ensemble** $\mathbb{T}_R(\Lambda, P)$ of a given degree profile (Λ, P) is defined as follows. When $R = 0$, the ensemble contains only one element, a single isolated node, and call it the **variable node of the generation** 0. Assume $R > 0$. We first generate a tree T from the $(R-2)$-generation tree ensemble $\mathbb{T}_{R-2}(\Lambda, P)$. For each variable node x of generation $R-2$, we draw an independent integer $i \geq 1$ distributed according to λ_i (or Λ_i if $R = 2$), and add $i-1$ function nodes, which are connected to x as its children. Then, for each of these function nodes a, we draw an independent integer $j \geq 1$ distributed according to ρ_j, and add $j-1$ variable nodes, which are connected to a as its children and called the **variable nodes of the generation** R.

In particular, Mézard and Montanari [MM09] shows that the local structure of a variable node of a random factor graph from $\mathbb{D}_n(\Lambda, P)$ and $\mathbb{G}_k(n, m)$ converges to this tree ensemble.

The more details of the following theorem can be found in [MM09].

Theorem 5. *Let (Λ, P) be a fixed degree profile, G be a random factor graph in the $\mathbb{D}_n(\Lambda, P)$ ensemble (and $\mathbb{G}_k(n,m)$ respectively), x be a variable node chosen uniformly at random from G, and R be a non-negative even number. Then, the local neighborhood $B_G(x,R)$ of the factor graph G converges in distribution to $\mathbb{T}_R(\Lambda, P)$ (and $\mathbb{T}_R(e^{kr(x-1)}, x^k)$ respectively) as $n \to \infty$.*

4.2 Proof of Theorem 2 and Theorem 3

We first give the proofs of Theorem 2 and Theorem 3. By using the Wormald's method of differential equations, we can calculate the degree profile of the factor graph $G_{\mathbf{\Phi}_t}$ of the residue instance $\mathbf{\Phi}_t$ 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 11 shows the freeness of the `UC`-decimation algorithm, which use Unit Clause Propagation `UC` as its local rule.

Lemma 11. *For $k \geq 3$ and $r > 0$, the `UC`-decimation algorithm $\texttt{DEC}_{\texttt{UC}}$ 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. Surprisingly, 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 12. *Assume the local rule τ outputs the exact marginal probability of a selected variable for any instance whose factor graph is a tree. For $k \geq 3$ and $r > 0$, the τ-decimation algorithm $\texttt{DEC}_\tau$ 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 $x \in \mathbb{R}$.

The proof for Lemma 11 and Lemma 12 can be found in Section 4.3 and 4.5, respectively. With these two lemmas, we move on to prove Theorem 2 and Theorem 3.

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 assumption described in Theorem 3, are strictly $2\mu(k,r)$-free. The results immediately follow by applying Theorem 1. From Lemma 11 and 12, 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 13, a lower bound of $w_1(k,r)$ in Lemma 14, and a lower bound of and $w_e(k,r)$ in Lemma 15. The proofs of these three lemmas are given in Section 4.6, 4.7 and 4.8, respectively.

Lemma 13. *For any $k \geq 4$ and $r \in (r_{\text{clt}}(k), r_{\text{sat}}(k))$, 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 14. *For any $k \geq k_0 \geq 3$ and $r \in (r_{\text{clt}}(k), r_{\text{sat}}(k))$, $w_1(k,r) \geq w_1^*(k_0)$, 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 15. *For any $k \geq k_0 \geq 3$ and $r \in (r_{\text{clt}}(k), r_{\text{sat}}(k))$, we have $w_e(k,r) \geq w_e^*(k_0, r_{\text{sat}}(k_0))$, where $w_e^*(k,r) = x^-(k,r) - kr^2(x^-(k,r))^k$ and*

$$x^{\pm}(k,r) = \left(\frac{1 \pm \sqrt{1 - 4(kr)^{-2}[(kr)^{\frac{1}{k-1}} - 1]}}{2}\right)^{\frac{1}{k-2}}. \tag{15}$$

Proof of Theorem 2. Let $k \geq 9$ and $r \in (r_{\text{clt}}(k), r_{\text{sat}}(k))$. By Lemma 13 and 14, 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 11, $\texttt{DEC}_{\texttt{UC}}$ is strictly $2\mu(k,r)$-free. The result follows. □

Proof of Theorem 3. Let $k \geq 13$ and $r \in (r_{\text{clt}}(k), r_{\text{sat}}(k))$. By Lemma 13 and 15, 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 12, $\texttt{DEC}_{\tau}$ is strictly $2\mu(k,r)$-free. The result follows. □

4.3 Proof of Lemma 11: Freeness of $\texttt{DEC}_{\texttt{UC}}$

In this section, we will approximate the number of free steps run by the $\texttt{UC}$-decimation $\texttt{DEC}_{\texttt{UC}}$ on the random k-XORSAT instance $\mathbf{\Phi} \sim \mathbf{\Phi}_k(n, rn)$, which implies that $\texttt{DEC}_{\texttt{UC}}$ is $w_1(k,r)$-free. To be specific, we will show the proof of Lemma 11, by using the Wormald's method of differential equations [Wor95].

We start from approximating the degree profile of the factor graph of the residue instance after t iterations. The degree profiles can help us to calculate the probability of the next iteration being a free step. Note that the notations for describing the degree profile are defined in Section 4.1. For each of those notations, we append "(t)" to them to specify the values right after t iterations.

Lemma 16. *For any local rule τ, after t iterations of DEC$_\tau$, w.h.p.the number of variables of degree i is given by*

$$n_i(t) = \binom{rn}{i}\left(\frac{k}{n}\right)^i\left(1-\frac{k}{n}\right)^{rn-i}(n-t)+o(n), \quad \textit{for } i=0,1,...,m, \tag{16}$$

and the number of equations of degree i is given by

$$m_i(t) = \binom{k}{i}\left(\frac{t}{n}\right)^{k-i}\left(1-\frac{t}{n}\right)^{i} rn+o(n), \quad \textit{for } i=1,2,...,k. \tag{17}$$

Proof. We are going to track the changes in the numbers of variable nodes and equation nodes of different degrees throughout the process of the algorithm, by using the method of differential equations from [Wor95].

To apply the method of differential equations, we need to compute the expected changes in the numbers of variable nodes and equation nodes of different degrees. Note that exactly one variable node is removed in each iteration, so the total number $\widehat{n}(t)$ of variable nodes is $n-t$ after t iterations.

Suppose we are at time $t \geq 0$ (i.e. after t iterations). The selected variable node $x_{s(t)}$ is going to be removed from the factor graph $G_{\mathbf{\Phi_t}}$. The selected variable node $x_{s(t)}$ is of degree i with probability $n_i(t)/\widehat{n}(t)$. So, for $i=0,1,2,...,m$, the expected change in the number n_i of variable nodes of degree i is $-1 \times n_i(t)/\widehat{n}(t)$. This yields

$$\mathbb{E}\left[n_i(t+1)-n_i(t)\right] = -\frac{n_i(t)}{\widehat{n}(t)} = -\frac{n_i(t)}{n-t} \tag{18}$$

for $i=0,1,...,m$. Note that the residue factor graph $\mathbf{\Phi_t}$ at time t is distributed over the ensemble $\mathbb{D}_n(\Lambda',P')$, where $\Lambda'_i = n_i(t)/\widehat{n}(t)$ for all i and $P'_j = m_j(t)/\widehat{m}(t)$ for all j. By Theorem 5, the local neighborhood $B_{\mathbf{\Phi_t}}(x_{s(t)},R)$ of the selected variable $x_{s(t)}$ of radius R in $G_{\mathbf{\Phi_t}}$ converges in distribution to $\mathbb{T}_R(\Lambda',P')$. Therefore, with probability $n_j/\widehat{n}$ for $j=0,1,2,...,m$, there are j equation nodes directly adjacent to the selected variable node $x_{s(t)}$. For those equation nodes, the degree of each equation node is l with probability $\rho_l(t)$, and decreases by 1 due to the removal of the selected variable node $x_{s(t)}$. In the other words, the expected changes in the numbers of equation nodes of different degree are given by

$$\begin{aligned}\mathbb{E}\left[m_k(t+1)-m_k(t)\right] &= \sum_{j=1}^{m} j\frac{n_j(t)}{\widehat{n}(t)}\cdot(\rho_k(t)\cdot(-1)) \\ &= \left(\sum_{j=1}^{m} j\frac{n_j(t)}{n-t}\right)\frac{-km_k(t)}{\sum_{j=1}^{k} jm_j(t)} \\ &= \left(\sum_{j=1}^{m} j\frac{n_j(t)}{n-t}\right)\frac{-km_k(t)}{\sum_{j=1}^{m} jn_j(t)} \\ &= \frac{-km_k(t)}{n-t}\end{aligned} \tag{19}$$

and

$$\begin{aligned}
\mathbb{E}\left[m_i(t+1)-m_i(t)\right] &= \sum_{j=1}^{m} j\frac{n_j}{\widehat{n}}\cdot(\rho_{i+1}(t)\cdot(+1)+\rho_i(t)\cdot(-1)) \\
&= \left(\sum_{j=1}^{m} j\frac{n_j(t)}{n-t}\right)\frac{(i+1)m_{i+1}(t)-im_i(t)}{\sum_{j=1}^{k} jm_j(t)} \\
&= \left(\sum_{j=1}^{m} j\frac{n_j(t)}{n-t}\right)\frac{(i+1)m_{i+1}(t)-im_i(t)}{\sum_{j=1}^{m} jn_j(t)} \\
&= \frac{(i+1)m_{i+1}(t)-im_i(t)}{n-t} \qquad \text{for } i=1,2,...,k-1. \qquad (20)
\end{aligned}$$

In the above calculation, we use the fact that $\sum_{j=1}^{k} jm_j(t)=\sum_{j=1}^{m} jn_j(t)$, which holds because both $\sum_{j=1}^{k} jm_j(t)$ and $\sum_{j=1}^{m} jn_j(t)$ are equal to the number of edges at time t.

Let $x=t/n$ be the normalized time. Furthermore, let

$$\begin{aligned}
y_i &= y_i(x) = m_i(xn)/n \quad \text{for } i=1,2,3,...,k, \\
z_i &= z_i(x) = n_i(xn)/n \quad \text{for } i=0,1,2,...,m, \quad \text{and} \\
z &= z(x) = \widehat{n}(xn)/n.
\end{aligned}$$

Then, the equations (18), (19) and (20) suggest the following different equations:

$$\begin{aligned}
\frac{dz_i}{dx} &= -\frac{z_i}{1-x} && \text{for } i=0,1,2,...,m && (21)\\
\frac{dy_k}{dx} &= \frac{-ky_k}{1-x} && && (22)\\
\frac{dy_i}{dx} &= \frac{(i+1)y_{i+1}-iy_i}{1-x} && \text{for } i=1,2,3,...,k-1 && (23)
\end{aligned}$$

At time $t=0$ (i.e. before running the algorithm), the number $n_i(0)$ of variable nodes of degree i is $\binom{rn}{i}\left(\frac{k}{n}\right)^i\left(1-\frac{k}{n}\right)^{rn-i}n+o(n)$ for all i. Since all rn equation nodes are of degree k at time $t=0$, $m_k(0)=rn$ and $m_i(0)=0$ for all i. These suggest the following initial conditions for those different equations above:

$$\begin{aligned}
z_i(0) &= \binom{rn}{i}\left(\frac{k}{n}\right)^i\left(1-\frac{k}{n}\right)^{rn-i} && \text{for } i=0,1,2,...,m && (24)\\
y_k(0) &= r && && (25)\\
y_i(0) &= 0 && \text{for } i=1,2,3,...,k-1 && (26)
\end{aligned}$$

The solution of the different equations with the initial conditions above is given by

$$\begin{aligned}
z_i(x) &= \binom{rn}{i}\left(\frac{k}{n}\right)^i\left(1-\frac{k}{n}\right)^{rn-i}(1-x) && \text{for } i=0,1,2,...,m\\
y_i(x) &= r\binom{k}{i}x^{k-1}(1-x)^i && \text{for } i=1,2,3,...,k
\end{aligned}$$

for any $0 \leq x \leq 1$. (The steps of solving the differential equations are shown at Section 4.4.) By the Wormald's method of differential equations, at time $t \geq 0$, w.h.p. the number $n_i(t)$ of variable nodes of degree i is equal to $z_i(t/n)n + o(n)$ for all i, and the number $m_i(t)$ of equation nodes of degree i is equal to $y_i(t/n)n + o(n)$. □

In an iteration of the UC-decimation algorithm $\texttt{DEC}_{\texttt{UC}}$, the local rule UC gives $1/2$ when there is no unit clause containing the selected variable $x_{s(t)}$. Therefore, an iteration is a free step if and only if there is no equation node of degree 1 adjacent to the selected variable node $x_{s(t)}$. With the degree profiles from Lemma 16, we can calculate the probability of an iteration being a free step, and thus approximate the total number of free steps.

Proof of Lemma 11. We track the number of free steps after t iterations, by using the Wormald's method of differential equations. Let $q(t)$ be the number of free steps after t iterations. Note that $q(0) = 0$. At time $t \geq 0$, if there exists at least one equation node of degree 1 adjacent to the selected variable node $x_{s(t)}$, then the $(t+1)$-st iteration is not a free step and $q(t+1) = q(t)$. In the other words, if all equation nodes adjacent to the selected variable node $x_{s(t)}$ are of degree not equal to 1, then the $(t+1)$-st iteration is a free step and $q(t+1) = q(t) + 1$. By Theorem 5, the probability that all equation nodes adjacent to the selected variable node are of degree not equal to 1 is given by $\sum_{j=0}^{m} (n_j(t)/\widehat{n}(t))(1 - \rho_1(t))^i$. Therefore, we have

$$\begin{aligned}
\mathbb{E}\left[q(t+1) - q(t)\right] &= \sum_{i=0}^{m} \frac{n_i(t)}{\widehat{n}(t)} (1 - \rho_1(t))^i \cdot (+1) \\
&= \sum_{i=0}^{rn} \frac{n_i(t)}{n-t} \left(1 - \frac{m_1(t)}{\sum_{j=1}^{k} j m_j(t)}\right)^i . \qquad (27)
\end{aligned}$$

From (17) and the polynomial identity $\sum_{i=1}^{n} i\binom{n}{i}(1-x)^i x^{n-i} = n(1-x)$, we can simplify $\sum_{j=1}^{k} j m_j(t)$ by

$$\begin{aligned}
\sum_{j=1}^{k} j m_j(t) &= \sum_{j=1}^{k} j \binom{k}{i} \left(\frac{t}{n}\right)^{k-i} \left(1 - \frac{t}{n}\right)^i rn + o(n) \\
&= \left(k \cdot \left(1 - \frac{t}{n}\right)\right) rn + o(n) \\
&= kr(n-t) + o(n).
\end{aligned}$$

Then, by (16), (17) and the fact that $\lim_{n\to\infty}(1-\frac{kx^{k-1}}{n})^{rn} = e^{-krx^{k-1}}$, we have

$$\begin{aligned}
\mathbb{E}\left[q(t+1)-q(t)\right] &= \sum_{i=0}^{m} \frac{n_i(t)}{n-t}\left(1-\frac{\binom{k}{1}\left(\frac{t}{n}\right)^{k-1}\left(1-\frac{t}{n}\right)rn}{kr(n-t)}\right)^{i} + o(1) \\
&= \sum_{i=0}^{m} \frac{n_i(t)}{n-t}\left(1-\left(\frac{t}{n}\right)^{k-1}\right)^{i} + o(1) \\
&= \sum_{i=0}^{m} n_i(0)\left(1-\left(\frac{t}{n}\right)^{k-1}\right)^{i} + o(1) \\
&= \sum_{i=0}^{m} \binom{rn}{i}\left(\frac{k}{n}\right)^{i}\left(1-\frac{k}{n}\right)^{rn-i}\left(1-\left(\frac{t}{n}\right)^{k-1}\right)^{i} + o(1) \\
&= \sum_{i=0}^{m} \binom{rn}{i}\left[\frac{k}{n}-\frac{k}{n}\left(\frac{t}{n}\right)^{k-1}\right]^{i}\left(1-\frac{k}{n}\right)^{rn-i} + o(1) \\
&= \left[\frac{k}{n}-\frac{k}{n}\left(\frac{t}{n}\right)^{k-1}+1-\frac{k}{n}\right]^{rn} + o(1) \\
&= \left[1-\frac{k}{n}\left(\frac{t}{n}\right)^{k-1}\right]^{rn} + o(1) \\
&= e^{-kr(t/n)^{k-1}} + o(1) \qquad (28)
\end{aligned}$$

By letting $x = t/n$ and $w = w(x) = q(xn)/n$, the equation (28) and the fact that $q(0) = 0$ suggest the differential equation

$$\frac{dw}{dx} = e^{-krx^{k-1}} \qquad (29)$$

and the initial condition

$$w(0) = 0. \qquad (30)$$

By the Wormald's method of differential equations, w.h.p. the number $q(t)$ of free steps after t iterations is equal to $w(t/n)n + o(n)$.

By the second fundamental theorem of calculus, we can write them as a definite integral

$$w(x) - w(0) = \int_0^x e^{-krt^{k-1}}\,dt$$

Note that $\frac{d}{dt}(krt^{k-1}) = kr(k-1)t^{k-2}$. Using integration by substitution, we have

$$\begin{aligned}
w(x) &= 0 + \int_0^x e^{-krt^{k-1}} dt \\
&= \int_0^x e^{-krx^{k-1}} \frac{1}{kr(k-1)} t^{2-k} \cdot kr(k-1)t^{k-2} dt \\
&= \int_0^{krx^{k-1}} e^{-krt^{k-1}} \frac{1}{kr(k-1)} t^{2-k} d(krt^{k-1}) \\
&= \frac{(kr)^{\frac{1}{1-k}}}{k-1} \int_0^{krx^{k-1}} e^{-krt^{k-1}} (kr)^{\frac{2-k}{k-1}} t^{2-k} d(krt^{k-1}) \\
&= \frac{(kr)^{\frac{1}{1-k}}}{k-1} \int_0^{krx^{k-1}} (krt^{k-1})^{\frac{1}{k-1}-1} e^{-krt^{k-1}} d(krt^{k-1}) \\
&= \frac{(kr)^{\frac{1}{1-k}}}{k-1} \gamma\left(\frac{1}{k-1}, krx^{k-1}\right)
\end{aligned}$$

where γ is the lower incomplete gamma function defined by

$$\gamma(a, x) \equiv \int_0^x t^{a-1} e^{-t} dt.$$

Hence, w.h.p. the total number of free steps run by the algorithm is $w_1(k, r)n + o(n)$, where

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

Hence, $\texttt{DEC}_{\texttt{UC}}$ is $w_1(k, r)$-free. $\square$

4.4 Solving differential equations (21), (22) and (23)

In this section, we show how to solve the differential equations (21), (22) and (23) with the initial conditions (24), (25) and (26). For $i = 0, 1, ..., m$, the differential equations in (21) can be written as

$$\frac{1}{z_i} dz_i = \frac{-1}{1-x} dx.$$

By separation of variable in integration and the initial condition (24), we have

$$\begin{aligned}
z_i(x) &= (1-x)z_i(0) \\
&= \binom{rn}{i} \left(\frac{k}{n}\right)^i \left(1 - \frac{k}{n}\right)^{rn-i} (1-x) \quad \text{for all } i = 0, 1, ..., m. \qquad (31)
\end{aligned}$$

Next, we are going to use induction from k to 1 to prove that

$$y_i(x) = r\binom{k}{i} x^{k-i} (1-x)^i$$

First, the differential equation (22) can be written as

$$\frac{1}{y_k}dy_k = k \cdot \frac{-1}{1-x}dx$$

By separation of variable in integration and the initial condition (25), we have

$$y_k(x) = (1-x)^k y_k(0) = r(1-x)^k$$

Now we assume $y_{i+1}(x) = r\binom{k}{i+1}x^{k-i-1}(1-x)^{i+1}$ for some $1 \le i < k$. From the differential equations in (23), we have

$$\frac{dy_i}{dx} + \frac{i}{1-x}y_i = \frac{i+1}{1-x}y_{i+1}$$

By applying the standard method for the first order linear differential equations and the induction assumption, we have

$$\begin{aligned}
y_i(x) &= \frac{\int \mu(x)\frac{i+1}{1-x}y_{i+1}dx + c}{\mu(x)} \\
&= \frac{\int \frac{1}{(1-x)^i}\frac{i+1}{1-x}r\binom{k}{i+1}x^{k-i-1}(1-x)^{i+1}dx + c}{\frac{1}{(1-x)^i}} \\
&= \frac{i+1}{k-i}r\binom{k}{i+1}x^{k-1}(1-x)^i + c(1-x)^i \\
&= r\binom{k}{i}x^{k-1}(1-x)^i + c(1-x)^i
\end{aligned}$$

where c is a constant and $\mu(x) = \exp\left(\int \frac{i}{1-x}dx\right) = \frac{1}{(1-x)^i}$. Since $1 \le i < k$, $y_i(0) = 0$ and thus $c = 0$. Hence, we have

$$y_i(x) = r\binom{k}{i}x^{k-i}(1-x)^i, \tag{32}$$

which completes the induction.

4.5 Proof of Lemma 12: Freeness of $\texttt{DEC}_\tau$

Now we assume the local rule τ output the exact marginal probability of a selected variable for any XORSAT instance which factor graph is a tree. We then show approximate the number of free steps run by the τ-decimation algorithm $\texttt{DEC}_\tau$ on the random k-XORSAT instance $\mathbf{\Phi} \sim \mathbf{\Phi}_k(n, rn)$. To be specific, we show the proof of Lemma 12, by using the Wormald's method of differential equations [Wor95].

In this section, the approach is slightly different from the proof in Section 4.3 since we do not have the details on how the local rule τ calculates its output value. However, since we assume that the local rule τ can give the exact marginal probability, we can determine the value output by the local rule τ, by studying the structure of the local neighborhood $B_{\mathbf{\Phi_t}}(x_{s(t)}, R)$ at time

$t \geq 0$.

Assume we are at time $t \geq 0$. We consider the local neighborhood $B_{\mathbf{\Phi_t}}(x_{s(t)}, R)$ of the selected variable $x_{s(t)}$ of radius $R \geq 0$. Note that the residue factor graph $\mathbf{\Phi_t}$ at time t is distributed over the ensemble $\mathbb{D}_n(\Lambda', P')$, where $\Lambda'_i = n_i(t)/\widehat{n}(t)$ for all i and $P'_j = m_j(t)/\widehat{m}(t)$ for all j. By Theorem 5, the local neighborhood $B_{\mathbf{\Phi_t}}(x_{s(t)}, R)$ converges in distribution to the tree ensemble $\mathbb{T}_R(\Lambda', P')$ at time t. The values of Λ'_i and P'_i are given by (16) and (17) since Lemma 16 can be applied to τ-decimation with any local rule τ. By our assumption, the value output by the local rule τ is equal to the marginal probability of the selected variable $x_{s(t)}$ on a random solution for the XORSAT instance induced by the local neighborhood $B_{\mathbf{\Phi_t}}(x_{s(t)}, R)$. So, we can obtain the value output by the local rule by calculating the marginal probability of $x_{s(t)}$, using the tree ensemble $\mathbb{T}_R(\Lambda', P')$.

For any $0 \leq l \leq R$, we denote by $T_l(x')$ a subtree of $B_{\mathbf{\Phi_t}}(x_{s(t)}, R)$ rooted at some node x' whose distance from $x_{s(t)}$ is $R - l$. According to the tree ensemble $\mathbb{T}_R(\Lambda', P')$, the tree $T_l(x)$ is i.i.d. for all variable nodes x' of same distance $R - l$ from $x_{s(t)}$. By abusing the notation, we can simply omit "(x)" and write $T_l = T_l(x)$. For any factor graph T which is a tree rooted at a variable node x', we say the tree T **has a free root** if in the XORSAT instance induced by T the marginal distribution of the root variable x' is an even distribution over $\{0, 1\}$. Now, we are going to prove that for any $0 \leq l \leq R-1$ the tree T_l has a free root with probability at least $S_l(t/n) + o(1)$, where the sequence $\{S_l(x)\}_{l \geq 0}$ is given by

$$S_0(x) = 1, \quad \text{and} \quad S_l(x) = \exp\left(-kr\left((1-x)\left(1 - S_{l-1}(x)\right) + x\right)^{k-1}\right) \quad \text{for any } l \geq 1 \tag{33}$$

for any $x \in \mathbb{R}$.

Lemma 17. *For $0 \leq l \leq R-1$, the tree T_l has a free root with probability at least $S_l(t/n) + o(1)$.*

Proof. First, we consider the tree T_0. The root variable node x' of T_0 has distance R from the selected variable $x_{s(t)}$. Since the radius of $B_{\mathbf{\Phi_t}}(x_{s(t)}, R)$ is R, the variable x' does not have any child node. Thus, T_0 consists of only one variable node, namely x', and no equation node. We can assign either 0 or 1 to x without violating any equation. Hence, the tree T_0 has a free root with probability $S_0(t/n) = 1$.

For $1 \leq l \leq R-1$, consider the subtree T_l of local neighborhood $B_{\mathbf{\Phi_t}}(x_{s(t)}, R)$, rooted at the variable node x_a of distance $R - l$ from $x_{s(t)}$. The variable node x_a has $i - 1$ child equation nodes with probability $\lambda_i(t)$ for $1 \leq i \leq m$. Each of those equation nodes e_a has $j - 1$ child variable nodes x_b with probability $\rho_j(t)$ for $1 \leq j \leq k$. We also know that each of these child variable nodes x_b is the root of an i.i.d. subtree T_{l-1}. In other words, each of those equation nodes e_a is connected to the roots of $j - 1$ i.i.d. subtrees T_{l-1} as its children.

For an equation node e_a mentioned above, if at least one of its child subtree T_{l-1} has a free root, then there are at least two solutions for the subtree T_{l-1}, one assigns 0 to the root x_b of subtree T_{l-1}, and another one assigns 1 to the root x_b of subtree T_{l-1}. Therefore, no matter what value we assign to x_a, we are able to choose a suitable assignment for the variables in that subtree T_{l-1} so that the equation of e_a and all equations in T_{l-1} are satisfied.

Note that, from (16) and (17), we know that at time t

$$\lambda_i(t) = i\binom{rn}{i}\left(\frac{k}{n}\right)^i\left(1-\frac{k}{n}\right)^{rn-i}\frac{1}{kr} + o(1) \quad \text{for } 1 \le i \le m, \text{ and}$$
$$\rho_j(t) = j\binom{k}{j}\left(\frac{t}{n}\right)^{k-j}\left(1-\frac{t}{n}\right)^j\frac{1}{k\left(1-\frac{t}{n}\right)} + o(1) \quad \text{for } 1 \le j \le k.$$

So, the probability of the equation node e_a having at least one of its child subtree T_{l-1} having a free root is given by

$$\begin{aligned}
&1-\sum_{j=1}^{k}\rho_j(t)(1-S_{l-1})^{j-1}\\
&= 1-\sum_{j=1}^{k} j\binom{k}{j}\left(\frac{k}{n}\right)^{k-j}\left(1-\frac{t}{n}\right)^j(1-S_{l-1})^j\cdot\frac{1}{k\left(1-\frac{t}{n}\right)(1-S_{l-1})} + o(1)\\
&= 1-\sum_{j=1}^{k} j\binom{k}{j}\left(\frac{k}{n}\right)^{k-j}\left[\left(1-\frac{t}{n}\right)(1-S_{l-1})\right]^j\cdot\frac{1}{k\left(1-\frac{t}{n}\right)(1-S_{l-1})} + o(1)\\
&= 1-k\left[\left(1-\frac{t}{n}\right)(1-S_{l-1})+\frac{t}{n}\right]^{k-1}\left(1-\frac{t}{n}\right)(1-S_{l-1})\\
&\qquad\qquad\qquad\qquad\cdot\frac{1}{k\left(1-\frac{t}{n}\right)(1-S_{l-1})} + o(1)\\
&= 1-\left[\left(1-\frac{t}{n}\right)(1-S_{l-1})+\frac{t}{n}\right]^{k-1} + o(1)\\
&\equiv S_l^*,
\end{aligned}$$

where $S_{l-1} = S_{l-1}(t/n)$. Note that $\lim_{n\to\infty}(1+\alpha/n)^n = e^\alpha$ for all $\alpha \in \mathbb{R}$. Then, the subtree T_l has a free root with probability

$$\begin{aligned}
\sum_{i=1}^{rn}\lambda_i(t)(S_l^*)^{i-1} &= \sum_{i=1}^{rn} i\binom{rn}{i}\left(\frac{k}{n}\right)^i\left(1-\frac{k}{n}\right)^{rn-i}(S_l^*)^i\cdot\frac{1}{krS_l^*} + o(1)\\
&= \sum_{i=1}^{rn} i\binom{rn}{i}\left(\frac{k}{n}S_l^*\right)^i\left(1-\frac{k}{n}\right)^{rn-i}\cdot\frac{1}{krS_l^*} + o(1)\\
&= rn\left[\frac{k}{n}S_l^*+\left(1-\frac{k}{n}\right)\right]^{rn-1}\left(\frac{k}{n}S_l^*\right)\cdot\frac{1}{krS_l^*} + o(1)\\
&= \left[\frac{k}{n}S_l^*+\left(1-\frac{k}{n}\right)\right]^{rn-1} + o(1)\\
&= \left(1+\frac{k(S_l^*-1)}{n}\right)^{rn-1} + o(1)\\
&= \exp(kr(S_l^*-1)) + o(1)\\
&= \exp\left(-kr\left[\left(1-\frac{t}{n}\right)(1-S_{l-1})+\frac{t}{n}\right]^{k-1}\right) + o(1)\\
&= S_l(t/n) + o(1).
\end{aligned}$$

□

Now, we can calculate the probability that the local neighborhood $B_{\mathbf{\Phi_t}}(x_{s(t)}, R)$ of the selected variable $x_{s(t)}$ of radius $R > 0$ has a free root with probability at least $S_R(t/n)$. The proof is similar to the proof of Lemma 17, except replacing $\lambda_i(t)$ with $\Lambda_i(t)$.

Lemma 18. *At time $t \geq 0$, the local neighborhood $B_{\mathbf{\Phi_t}}(x_{s(t)}, R)$ of the selected variable $x_{s(t)}$ of radius $R \geq 0$ has a free root with probability at least $S_R(t/n) + o(1)$.*

Proof. The root variable node $x_{s(t)}$ has i child equation node with probability $\Lambda_i(t)$ for $0 \leq i \leq m$. Each of those equation nodes e_a has $j-1$ child variable nodes x_b with probability $\rho_j(t)$, and each of these child variable nodes x_b is the root of an i.i.d. subtree T_{R-1}. In other words, each of those equation nodes e_a is connected to the roots of $j-1$ i.i.d. subtrees T_{R-1} as its children.

For an equation node e_a mentioned above, if at least one of its child subtree T_{R-1} has a free root, then we are able to obtain a satisfying assignment for all variable nodes in the child subtree T_{R-1} which assigns either 1 or 0 to the root x_b of subtree T_{R-1}. Therefore, no matter what value we assign to x_a, we are able to choose a suitable assignment for the variables in that subtree T_{R-1} so that the equation of e_a and all equations in T_{R-1} are satisfied.

Similar to the proof of Lemma 17, the probability that the equation node e_a has at least one of its child subtree T_{R-1} having a free root is given by

$$1 - \sum_{j=1}^{k} \rho_j(t)(1 - S_{R-1})^{j-1} = \exp\left(-kr\left[\left(1 - \frac{t}{n}\right)(1 - S_{R-1}) + \frac{t}{n}\right]^{k-1}\right) + o(1) \equiv S_R^*,$$

where $S_{R-1} = S_{R-1}(t/n)$. From (16), at time t we have $\Lambda_i(t) = \binom{rn}{i}\left(\frac{k}{n}\right)^i\left(1 - \frac{k}{n}\right)^{rn-i} + o(1)$ for $0 \leq i \leq m$. Note that $\lim_{n\to\infty}(1 + \alpha/n)^n = e^\alpha$ for all $\alpha \in \mathbb{R}$. Hence, the probability that the local neighborhood $B_{\mathbf{\Phi_t}}(x_{s(t)}, R)$ has a free root is given by

$$\begin{aligned}
\sum_{i=0}^{rn} \Lambda_i(t)(S_R^*)^{i-1} &= \sum_{i=0}^{rn} \binom{rn}{i}\left(\frac{k}{n}\right)^i\left(1 - \frac{k}{n}\right)^{rn-i}(S_R^*)^i + o(1) \\
&= \sum_{i=0}^{rn} \binom{rn}{i}\left(\frac{k}{n}S_R^*\right)^i\left(1 - \frac{k}{n}\right)^{rn-i} + o(1) \\
&= \left[\frac{k}{n}S_R^* + \left(1 - \frac{k}{n}\right)\right]^{rn} + o(1) \\
&= \left[1 + \frac{k(S_R^* - 1)}{n}\right]^{rn} + o(1) \\
&= \exp\left(kr(S_R^* - 1)\right) + o(1) \\
&= \exp\left(-kr\left[\left(1 - \frac{t}{n}\right)(1 - S_{R-1}) + \frac{t}{n}\right]^{k-1}\right) + o(1) \\
&= S_R(t/n) + o(1),
\end{aligned}$$

□

Proof of Lemma 12. Lemma 18 implies that at time $t \geq 0$ the local rule τ gives the value $1/2$ to the decimation algorithm with probability at least $S_R(t/n)$. Now we can calculate the number of free steps run by the τ-decimation algorithm by tracking the number throughout the process of the algorithm. We know that the $(t+1)$-st iteration is a free step with probability at least $S_R(t/n) + o(1)$. Let $q(t)$ be a lower bound of the number of free steps after t iterations, with $q(0) = 0$ and

$$\mathbb{E}\left[q_e(t+1) - q_e(t)\right] = S_R(t/n) + o(1). \tag{34}$$

By letting $x = t/n$ and $w = w(x) = q(xn)/n$, the equation (34) suggests the differential equation

$$\frac{dw}{dx} = S_R(x) \tag{35}$$

with the initial condition

$$w(x) = 0 \tag{36}$$

since $q(0) = 0$. By the Wormald's method of differential equations, w.h.p. the lower bound $q(t)$ of the number of free steps after t iterations is given by

$$\left(\int_0^{t/n} S_R(x)dx\right) n + o(n).$$

Hence, w.h.p. the total number of free steps run by the τ-decimation algorithm is lower bounded by $w_e(k,r)n + o(n)$, where $w_e(k,r)$ is given by

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

Hence, $\texttt{DEC}_\tau$ is $w_e(k,r)$-free. □

4.6 Proof of Lemma 13: Upper bound of cluster diameter

In this section, we give the proof of Lemma 13, which gives an upper bound for the diameter of a cluster that can be written as a closed form expression. From Lemma 4, we know that for any $k \geq 3$ and $r_{\text{core}}(k) < r < r_{\text{sat}}(k)$ w.h.p. the diameter of a cluster is upper bounded by $\mu(k,r)n + o(n)$, where

$$\mu(k,r) = \exp(-krQ_{k,r}^{k-1}) + krQ_{k,r}^{k-1}\exp(-krQ_{k,r}^{k-1})$$

and $Q_{k,r}$ is the largest solution of the fixed point equation $Q = 1 - \exp(-krQ^{k-1})$ with the given values of k and r. This implicit expression would make calculations complicated. So, we slightly relax the upper bound to obtain a simpler expression $\mu_u(k)$ in Lemma 13. Note that this lemma can only be applied when $k \geq 4$ due to some calculation restriction, which will be mentioned later in this section.

To prove Lemma 13, we first re-write the fixed point equation $Q = 1 - \exp(-krQ^{k-1})$ as

$$krQ^{k-1} = -\ln(1-Q).$$

Since $Q_{k,r}$ satisfies this equation, we can write $\mu(k,r)$ as

$$\begin{aligned}\mu(k,r) &= \exp(-krQ_{k,r}^{k-1}) + krQ_{k,r}^{k-1}\exp(-krQ_{k,r}^{k-1}) \\ &= (1-Q_{k,r}) - (1-Q_{k,r})\ln(1-Q_{k,r}) \end{aligned} \tag{37}$$

Note that $Q_{k,r}$ must lie in the interval $[0,1)$ as

$$\begin{aligned} &Q \leq 0 < 1 - \exp(-krQ^{k-1}) \quad \text{for any } Q \leq 0 \quad \textit{and} \\ &Q > 1 \geq 1 - \exp(-krQ^{k-1}) \quad \text{for any } Q > 1. \end{aligned}$$

Further note that the real-valued function $f(x) = (1-x) - (1-x)\ln(1-x)$ is strictly decreasing on $[0,1)$. So, it suffices to find a lower bound of $Q_{k,r}$, in order to find an upper bound of $\mu(k,r)$.

Next, we try to show that $e^{-1/k}$ is a lower bound of $Q_{k,r}$. To facilitate the calculation, we define a real-valued analytic function $G : [3,+\infty) \times [0,1] \times [0,1] \to \mathbb{R}$ by

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

which have the following derivatives:

$$\frac{\partial G}{\partial r} = \exp(-krQ^{k-1}) \cdot kQ^{k-1} \tag{38}$$

$$\frac{\partial G}{\partial Q} = \exp(-krQ^{k-1}) \cdot kr(k-1)Q^{k-2} - 1 \tag{39}$$

$$\frac{\partial^2 G}{\partial Q^2} = \exp(-krQ^{k-1}) \cdot kr(k-1)Q^{k-3}[(k-2) - kr(k-1)Q^{k-1}] \tag{40}$$

Before proving that $e^{-1/k}$ is a lower bound of $Q_{k,r}$, we give the following two lemmas, Lemma 19 and Lemma 20, which will be used later.

Lemma 19. *For any $k \geq 3$, $G(k, r_{\text{core}}(k), e^{-1/k}) \leq 0$.*

Proof. We prove it by contradiction. Assume $G(k, r_{\text{core}}(k), e^{-1/k}) > 0$ for some $k \geq 3$. By the continuity of G, there exists $r' < r_{\text{core}}(k)$ such that $G(k, r', e^{-1/k}) > 0$. Note that $G(k, r', 1) = 1 - \exp(-kr') - 1 < 0$. Again, by the continuity of G, there exists $Q' \in (e^{-1/k}, 1)$ such that $G(k, r', Q') = 0$. Since $0 < r' < r_{\text{core}}(k)$, this contradicts the definition of $r_{\text{core}}(k)$. $\square$

Lemma 20. *For any $k \geq 3$, there exists $r_1(k) \in (r_{\text{core}}(k), 1)$ such that*

$$G(k, r, e^{-1/k}) \begin{cases} < 0 & \text{for } r_{\text{core}}(k) \leq r < r_1(k) \\ = 0 & \text{for } r = r_1(k) \\ > 0 & \text{for } r_1(k) < r \leq 1 \end{cases}$$

Proof. Note that $\lim_{r\to\infty} G(k, r, e^{-1/k}) = \lim_{r\to\infty} 1-\exp(-kre^{-(k-1)/k})-e^{-1/k} = 1-e^{-1/k} > 0$. That means for sufficiently large r, $G(k, r, e^{-1/k}) > 0$. On the other hands, from Lemma 19, we know that $G(k, r_{\text{core}}(k), e^{-1/k}) \leq 0$ for any $k \geq 3$. Moreover, from (38), we have $\frac{\partial}{\partial r} G(k, r, e^{-k}) > 0$, which implies $G(k, r, e^{-1/k})$ is strictly increasing with r for $r > 0$. Therefore, there exists $r_1 = r_1(k) \in (r_{\text{core}}(k), 1)$ such that $G(k, r, e^{-1/k}) < 0$ for $r_{\text{core}}(k) \leq r < r_1(k)$, $G(k, r, e^{-1/k}) > 0$ for $r_1(k) < r \leq 1$ and $G(k, r_1(k), e^{-1/k}) = 0$. $\square$

With Lemma 19 and Lemma 20, we can prove the $e^{-1/k}$ is a lower bound for $Q_{k,r}$. We split the proof into two cases: the case of $r \in [r_{\text{core}}(k), r_1(k)]$, and the case of $r \in (r_1(k), 1]$. We first study the latter case, which is easier to be proved.

Lemma 21. *For any $k \geq 3$ and $r \in (r_1(k), 1]$, we have $Q_{k,r} > e^{-1/k}$.*

Proof. From Lemma 20, we have $G(k, r, e^{-1/k}) > 0$ since $r > r_1(k)$. Note that $G(k, r, 1) = 1 - \exp(-kr) - 1 < 0$. By the continuity of G, there exists at least one $Q' \in (e^{-1/k}, 1)$ such that $G(k, r, Q') = 0$, which can be written as $Q' = 1 - \exp(-kr(Q')^{k-1})$. By the definition of $Q_{k,r}$, we then have $Q_{k,r} \geq Q' > e^{-1/k}$. $\square$

Next, we study the case of $r \in [r_{\text{core}}(k), r_1(k)]$. To prove that $Q_{k,r} > e^{-1/k}$, we need the two following facts from the basic calculus. With these two facts, we can prove that $Q_{k,r} > e^{-1/k}$ for $r \in [r_{\text{core}}(k), r_1(k)]$ in Lemma 22. Note that the condition $k \geq 4$ is required in the calculation in the proof of Lemma 22. This is the reason why Lemma 13 requires $k \geq 4$.

Fact 1. *Let $f : [a, b] \to \mathbb{R}$ be an analytic function. If $f''(x) > 0$ for any $x \in (a, b)$, then $\max_{x\in[a,b]} f(x) = \max\{f(a), f(b)\}$.*

Fact 2. *Let $f : [a, b] \to \mathbb{R}$ be an analytic function. If there exists $c \in (a, b)$ such that*

$$f''(x) \begin{cases} > 0 & \text{for } x \in (a, c) \\ = 0 & \text{for } x = c \\ < 0 & \text{for } x \in (c, b) \end{cases}$$

and $f'(b) \geq 0$, then $\max_{x\in[a,b]} f(x) = \max\{f(a), f(b)\}$.

Lemma 22. *For any $k \geq 4$ and $r \in [r_c, r_1]$, where $r_c = r_{\text{core}}(k)$ and $r_1 = r_1(k)$, we have $Q_{k,r} > e^{-1/k}$.*

Proof. Given arbitrary $k \geq 3$ and $Q > 0$, from (38), we know that $G(k,r,Q)$ is strictly increasing with $r \in [r_c, r_1]$, and thus we have $Q(k,r,Q) < Q(k,r_1,Q)$ for any $r \in [r_c, r_1]$.

From (40), when we view G as a function of Q with fixed $k \geq 3$ and $r > 0$ there are two points of inflection:

$$Q = 0 \quad \text{and} \quad Q = \left(\frac{k-2}{kr(k-1)}\right)^{1/(k-1)}$$

Now, we consider the following two cases separately:

1. $\left(\frac{k-2}{kr(k-1)}\right)^{1/(k-1)} < e^{-1/k}$
2. $e^{-1/k} \leq \left(\frac{k-2}{kr(k-1)}\right)^{1/(k-1)}$

Case 1. Assume $(\frac{k-2}{kr(k-1)})^{1/(k-1)} < e^{-1/k}$. Then we have

$$\frac{\partial^2}{\partial Q^2} G(k,r_1,Q) \begin{cases} > 0 & \text{for } 0 < Q < \left(\frac{k-2}{kr(k-1)}\right)^{1/(k-1)} \\ = 0 & \text{for } Q = \left(\frac{k-2}{kr(k-1)}\right)^{1/(k-1)} \\ < 0 & \text{for } \left(\frac{k-2}{kr(k-1)}\right)^{1/(k-1)} < Q < e^{-1/k} \end{cases}$$

Moreover, since $G(k,r_1,e^{-1/k}) = 0$, we have $e^{-1/k} = 1 - \exp(-kr_1 e^{-(k-1)/k})$. Therefore, we have

$$\begin{aligned} \frac{\partial}{\partial Q} G(k,r_1,Q)\Big|_{Q=e^{-1/k}} &= \exp(-kr_1 e^{-(k-1)/k}) \cdot kr_1(k-1)e^{-(k-2)/k} - 1 \\ &= \exp(-kr_1 e^{-(k-1)/k}) \cdot kr_1 e^{-(k-1)/k} \cdot (k-1)e^{1/k} - 1 \\ &= (1 - e^{-1/k}) \cdot [-\ln(1 - e^{-1/k})] \cdot (k-1)e^{1/k} - 1 \\ &> 0 \end{aligned}$$

for $k \geq 4$. By Fact 2, we have $G(k,r_1,Q) < \max\{G(k,r_1,0), G(k,r_1,e^{-1/k})\}$ for any $G \in (0, e^{-1/k})$. Note that both $G(k,r_1,0)$ and $G(k,r_1,e^{-1/k})$ are less than 0. So, $G(k,r,Q) \leq G(k,r_1,Q) < 0$ for any $k \geq 4$, $r \in [r_c, r_1]$ and $Q \in [0, e^{-1/k}]$. Therefore, $Q_{k,r}$ cannot be in $[0, e^{-1/k}]$ by its definition, and hence $Q_{k,r} > e^{-1/k}$.

Case 2. Assume $e^{-1/k} \leq (\frac{k-2}{kr(k-1)})^{1/(k-1)}$. From (40), we have

$$\frac{\partial^2}{\partial Q^2} G(k,r_1,Q) > 0$$

for $Q \in [0, e^{-1/k}]$. By Fact 1, we know that for $G \in [0, e^{-1/k}]$

$$G(k,r_1,Q) \leq \max\{G(k,r_1,0), G(k,r_1,e^{-1/k})\}.$$

Since both $G(k,r_1,0)$ and $G(k,r_1,e^{-1/k})$ are less than 0, we have $G(k,r,Q) \leq G(k,r_1,Q) < 0$

for any $k \geq 3$, $r \in [r_c, r_1]$ and $Q \in [0, e^{-1/k}]$. Therefore, $Q_{k,r}$ cannot be in $[0, e^{-1/k}]$ by its definition, and hence $Q_{k,r} > e^{-1/k}$. $\square$

Now, we can complete the proof for Lemma 13.

Proof of Lemma 13. Let $k \geq 4$ and $r \in (r_{\text{clt}}(k), r_{\text{sat}}(k))$. Note the $r_{\text{clt}}(k) = r_{\text{core}}(k)$ for random k-XORSAT. From the fixed point equation $Q = 1 - \exp(-krQ^{k-1})$, we have $krQ_{k,r}^{k-1} = -\ln(1 - Q_{k,r})$. Note that the real-valued function $(1-x) - (1-x)\ln(1-x)$ is strictly decreasing on $[0, 1)$, and $Q_{k,r} \in [0, 1)$. By Lemma 21 and Lemma 22, we have $Q_{k,r} > e^{-1/k}$. Combining all these, the result is followed by

$$\begin{aligned}\mu(k,r) &= \exp(-krQ_{k,r}^{k-1}) + krQ_{k,r}^{k-1}\exp(-krQ_{k,r}^{k-1}) \\ &= (1 - Q_{k,r}) - (1 - Q_{k,r})\ln(1 - Q_{k,r}) \\ &< (1 - e^{-1/k}) - (1 - e^{-1/k})\ln(1 - e^{-1/k}) \\ &= \mu_u(k).\end{aligned}$$

$\square$

4.7 Proof of Lemma 14: Lower bound of $w_1(k, r)$

In this section, we prove Lemma 14, which gives a lower bound for $w_1(k, r)$. To do that, we first prove that $w_1(k, r)$ is lower bounded by

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

for $k \geq 3$ and $r \in [0, 1]$. We then prove that $w_1^*(k)$ is decreasing with integer $k \geq 3$, and thus $w_1(k, r)$ has a lower bound $w_1^*(k_0)$ for any $k \geq k_0 \geq 3$.

We start from proving that $w_1(k, r)$ is decreasing with $r \in [0, 1]$, which implies $w_1(k, r) \geq w_1(k, 1)$ for any $r \in [0, 1]$ and $k \geq 3$.

Lemma 23. *$w_1(k, r)$ is decreasing with $r \in [0, 1]$ for any $k \geq 3$.*

Proof. We obtain the derivative of $w_1(k, r)$ with respect to r by the followings.

$$\begin{aligned}\frac{\partial w_1}{\partial r} &= \frac{\frac{1}{1-k}(kr)^{\frac{1}{1-k}-1}k}{k-1}\gamma\left(\frac{1}{k-1}, kr\right) + \frac{(kr)^{\frac{1}{1-k}}}{k-1}(kr)^{\frac{1}{1-k}-1}e^{-kr}k \\ &= \frac{k(kr)^{\frac{1}{1-k}-1}}{k-1}\left[\frac{1}{1-k}\gamma\left(\frac{1}{k-1}, kr\right) + (kr)^{\frac{1}{k-1}}e^{-kr}\right] \\ &= \frac{k(kr)^{\frac{1}{1-k}-1}}{k-1}h_k(r)\end{aligned}$$

where $h_k(r)$ is given by

$$h_k(r) = \frac{1}{1-k}\gamma\left(\frac{1}{k-1}, kr\right) + (kr)^{\frac{1}{k-1}}e^{-kr}.$$

Note that $h_k(0) = 0$ and its derivative is given by

$$\begin{aligned}\frac{dh_k}{dr} &= \frac{1}{1-k}(kr)^{\frac{1}{k-1}-1}e^{-kr}k + \frac{1}{k-1}(kr)^{\frac{1}{k-1}-1}ke^{-kr} + (kr)^{\frac{1}{k-1}}e^{-kr}(-k) \\ &= -k(kr)^{\frac{1}{k-1}}e^{-kr} \\ &\le 0.\end{aligned}$$

Therefore, $h_k(r)$ is decreasing with r and $h_k(r) \le h_k(0) = 0$ for any $r \in [0,1]$. Hence, $\frac{\partial w_1}{\partial r} \le 0$ and thus w_1 is decreasing with $r \in [0,1]$ for any $k \ge 3$. □

By the above lemma, we know that $w_1(k,r) \ge w_1(k,1)$ for any $r \in [0,1]$. Next we will prove that $w_1(k,1)$ is lower bounded by $w_1^*(k)$.

Lemma 24. *For $k \ge 3$, $w_1(k,1) \ge w_1^*(k)$.*

Proof. For any $k \ge 3$ and any $t \ge 0$ we have

$$k > k\left(\frac{k}{k+1}\right)^{k-1} \quad \text{and} \quad t^{\frac{1}{k-1}-1}e^{-t} > 0,$$

This implies that

$$w_1(k,1) = \frac{k^{\frac{1}{1-k}}}{k-1}\gamma\left(\frac{1}{k-1}, k\right) > \frac{k^{\frac{1}{1-k}}}{k-1}\gamma\left(\frac{1}{k-1}, k\left(\frac{k}{k+1}\right)^{k-1}\right) = w_1^*(k). \tag{41}$$

□

Next, we prove that $\{w_1^*(k)\}_{k\ge3}$ is an increasing sequence, which implies that $w_1^*(k) \ge w_1^*(k_0)$ for any $k \ge k_0$.

Lemma 25. *$\{w_1^*(k)\}_{k\ge3}$ is an increasing sequence.*

Proof. Note that for $k, x > 0$ we have

$$\int e^{-kt^{k-1}}dx = -\frac{k^{\frac{1}{1-k}}}{k-1}\,\Gamma\left(\frac{1}{k-1}, kx^{k-1}\right) + \text{constant}.$$

By replacing k with $k+1$, we also have

$$\int e^{-(k+1)t^k}dx = -\frac{(k+1)^{\frac{1}{-k}}}{k}\,\Gamma\left(\frac{1}{k}, (k+1)x^k\right) + \text{constant}.$$

Therefore, we have

$$\begin{aligned}\int_0^{\frac{k}{k+1}} e^{-kt^{k-1}}dx &= \frac{k^{\frac{1}{1-k}}}{k-1}\left[\Gamma\left(\frac{1}{k-1}\right) - \Gamma\left(\frac{1}{k-1}, k\left(\frac{k}{k+1}\right)^{k-1}\right)\right] \\ &= \frac{k^{\frac{1}{1-k}}}{k-1}\,\gamma\left(\frac{1}{k-1}, k\left(\frac{k}{k+1}\right)^{k-1}\right) \\ &= w_1^*(k)\end{aligned}$$

and

$$\begin{aligned}
\int_0^{\frac{k}{k+1}} e^{-(k+1)t^k} dx &= \frac{(k+1)^{\frac{1}{-k}}}{k} \left[\Gamma\left(\frac{1}{k}\right) - \Gamma\left(\frac{1}{k}, (k+1)\left(\frac{k}{k+1}\right)^k\right)\right] \\
&= \frac{(k+1)^{\frac{1}{-k}}}{k} \gamma\left(\frac{1}{k}, (k+1)\left(\frac{k}{k+1}\right)^k\right) \\
&< \frac{(k+1)^{\frac{1}{-k}}}{k} \gamma\left(\frac{1}{k}, (k+1)\left(\frac{k+1}{k+2}\right)^k\right) \\
&= w_1^*(k+1).
\end{aligned}$$

The above inequality is based on the fact that the lower incomplete gamma function $\gamma(a,x)$ is strictly increasing with $x \geq 0$ and $\frac{k}{k+1} < \frac{k+1}{k+2}$.

For $0 \leq x \leq \frac{k}{k+1}$, we have $e^{-kx^{k-1}} \leq e^{-(k+1)x^k}$. Therefore, we have

$$w_1^*(k) = \int_0^{\frac{k}{k+1}} e^{-kt^{k-1}} dx \leq \int_0^{\frac{k}{k+1}} e^{-(k+1)t^k} dx \leq w_1^*(k+1)$$

and thus $\{w_1^*(k)\}_{k\geq 3}$ is an increasing sequence. □

Combining above lemmas, we can complete the proof of Lemma 14.

Proof of Lemma 14. From Lemma 23, 24 and 25, we have $w_1(k,r) \geq w_1(k,1) \geq w_1^*(k) \geq w_1^*(k_0)$ for any $k \geq k_0 \geq 3$ and $r \in [0,1]$. □

4.8 Proof of Lemma 15: Lower bound of $w_e(k,r)$

In this section, we prove Lemma 15, which gives a lower bound for $w_e(k,r)$. Recall that

$$S_0(x) = 1 \text{ and } S_l(x) = \exp\left(-kr\left((1-x)(1-S_{l-1}(x)) + x\right)^{k-1}\right) \quad \text{for any } l \geq 1 \text{ and } x \in \mathbb{R}. \tag{42}$$

We first show that $\{S_l(x)\}_{l\geq 0}$ is decreasing.

Lemma 26. *For any $k \geq 3$, $r \in [0,1]$ and $x \in [0,1]$, we have $0 < S_l(x) \leq 1$ for any $l \geq 0$, and the sequence $\{S_l(x)\}_{l\geq 0}$ is decreasing.*

Proof. Without loss of generality, we write $S_l = S_l(x)$. First, we prove that $0 < S_l \leq 1$ for all $l \geq 0$, by induction. Note that $S_0 = 1$. If $0 < S_l \leq 1$ for some $l \geq 0$, then it is easy to see from (42) that we also have $0 < S_{l+1} \leq 1$. Therefore, we know that $0 < S_l \leq 1$ for all $l \geq 0$.

Then, we prove $S_{l+1} \leq S_l$ for any $l \geq 0$ by induction again. For $l = 0$, we have $S_1 = \exp(-kr((1-x)(1-S_0)+x)^{k-1}) = \exp(-krx^{k-1}) \leq 1 = S_0$. Assume that $S_{l+1} \leq S_l$ for some

$l \geq 0$. Then, we have

$$\begin{aligned} S_{l+2} &= \exp(-kr((1-x)(1-S_{l+1})+x)^{k-1}) \\ &\leq \exp(-kr((1-x)(1-S_l)+x)^{k-1}) \\ &= S_{l+1}. \end{aligned}$$

Therefore, $S_{l+1} \leq S_l$ is true for all $l \geq 0$. The result follows. $\square$

Since the sequence $\{S_l(x)\}_{l\geq 0}$ is decreasing and bounded from below by 0, by monotone convergence theorem, it converges as $l \to \infty$. In particular, $\{S_l(x)\}_{l\geq 0}$ converges to $\hat{S}(x)$ as $l \to \infty$, where $\hat{S}(x)$ is the largest solution of the fixed point equation

$$S = \exp(-kr((1-x)(1-S)+x)^{k-1}), \tag{43}$$

and $S_l(x) \geq \hat{S}(x)$ for any $l \geq 0$.

Lemma 27. *Given $k \geq 3$ and $r \in [0,1]$, we have $\hat{S}(x) \geq 1-(kr)^2x^{k-1}$ for any $0 \leq x < x^-(k,r)$, where*

$$x^{\pm}(k,r) = \left(\frac{1 \pm \sqrt{1-4(kr)^{-2}[(kr)^{\frac{1}{k-1}}-1]}}{2} \right)^{\frac{1}{k-2}}. \tag{44}$$

Proof. Define the analytic real-valued function $F(k,r,x,s)$ by

$$F(k,r,x,s) = \exp(-kr[(1-x)(1-s)+x]^{k-1}) - s$$

for any $k \geq 3$, $r \in [0,1]$, $x \in [0,1]$ and $s \in \mathbb{R}$. Note that we have

$$\begin{aligned} F(k,r,x,s) &= \exp(-kr[(1-x)(1-s)+x]^{k-1}) - s \\ &\geq 1 - kr[(1-x)(1-s)+x]^{k-1} - s \\ &\geq 1 - kr[(1-x^{k-2})(1-s)+x]^{k-1} - s \end{aligned}$$

since $\exp(-y) \geq 1-y$ for any $y \in \mathbb{R}$, and $x \in [0,1]$. Now we set $s = 1-(kr)^2x^{k-1}$. With the polynomial identity $X^{k-1} - Y^{k-1} = (X - Y) \sum_{i=1}^{k-1} X^{k-1-i} Y^{i-1}$, we then have

$$\begin{aligned}
&F(k, r, x, 1 - (kr)^2 x^{k-1}) \\
&\geq (kr)^2 x^{k-1} - kr[(1 - x^{k-2})(kr)^2 x^{k-1} + x]^{k-1} \\
&= kr \left([(kr)^{\frac{1}{k-1}} x]^{k-1} - [(1 - x^{k-2})(kr)^2 x^{k-1} + x]^{k-1} \right) \\
&= kr \left([(kr)^{\frac{1}{k-1}} x] - [(1 - x^{k-2})(kr)^2 x^{k-1} + x] \right) \cdot F_2(k, r, x) \\
&= kr \left((kr)^{\frac{1}{k-1}} x - (kr)^2 x^{k-1} + (kr)^2 x^{2k-3} - x \right) \cdot F_2(k, r, x) \\
&= krx \left((kr)^2 (x^{k-2})^2 - (kr)^2 (x^{k-2}) + ((kr)^{\frac{1}{k-1}} - 1) \right) \cdot F_2(k, r, x) \\
&= krx \cdot F_1(k, r, x) \cdot F_2(k, r, x)
\end{aligned}$$

where

$$\begin{aligned}
F_1(k, r, x) &= (kr)^2 (x^{k-2})^2 - (kr)^2 (x^{k-2}) + ((kr)^{\frac{1}{k-1}} - 1) \quad \text{and} \\
F_2(k, r, x) &= \sum_{i=1}^{k-1} [(kr)^{\frac{1}{k-1}} x]^{k-1-i} [(1 - x^{k-2})(kr)^2 x^{k-1} + x]^{i-1}.
\end{aligned}$$

By the quadratic formula and (44), we know that $F_1(k, r, x) \geq 0$ if $x \leq x^-(k, r)$ or $x \geq x^+(k, r)$. In particular, $F_1(k, r, x) = 0$ if $x = x^-(k, r)$ or $x = x^+(k, r)$. For $x \geq 0$, we have $(kr)^{\frac{1}{k-1}} x \geq 0$ and $(1 - x)^{k-2} (kr)^2 x^{k-1} + x \geq 0$, and thus $F_2(k, r, x) \geq 0$. Hence, we have $F(k, r, x, 1 - (kr)^2 x^{k-1}) \geq 0$ for any $0 \leq x \leq x^-(k, r)$.

Now we view F as a function of s. For $0 \leq x \leq x^-(k, r)$, we know that $F(k, r, x, 1 - (kr)^2 x^{k-1}) \geq 0$ and $F(k, r, x, 1) = \exp(-krx^{k-1}) - 1 \leq 0$. By the continuity of F as a function of s, there exists at least one $s_0 \in [1 - krx^{k-1}, 1]$ such that $F(k, r, x, s_0) = 0$, which implies

$$s_0 = \exp(-kr((1 - x)(1 - s_0) + x)^{k-1}).$$

By definition, $\hat{S}(k, r, x)$ is the largest solution of the fixed point equation (43), so we have $\hat{S}(x) \geq s_0 \geq 1 - krx^{k-1}$. □

Proof of Lemma 15. From Lemma 26, for $x \in (r_{\text{clt}}(k), r_{\text{sat}}(k))$, the sequence $\{S_l(x)\}_{l \geq 0}$ is decreasing and lower bounded by 0. By the monotone convergence theorem, the sequence $\{S_l(x)\}_{l \geq 0}$ converges as $l \to \infty$. In particular, it converges to $\hat{S}(x)$ which is the largest solution of the fixed point equation in (43), and $S_l(x) \geq \hat{S}(x)$ for all $l \geq 0$. Therefore, we have

$$w_e(k, r) = \int_0^1 S_R(x) dx \geq \int_0^1 \hat{S}(x) dx.$$

Furthermore, since $\{S_l(x)\}_{l \geq 0}$ is lower bounded by 0, $\hat{S}(x)$ is non-negative for any $x \in [0, 1]$.

Thus, we have

$$\int_0^1 \hat{S}(x)dx \geq \int_0^{x^-(k,r)} \hat{S}(x)dx.$$

Then, by Lemma 27, we have $\hat{S}(x) \geq 1 - (kr)^2 x^{k-1}$ for any $0 \leq x \leq x^-(k,r)$, which implies

$$\begin{aligned}\int_0^1 \hat{S}(x)dx &\geq \int_0^{x^-(k,r)} (1 - (kr)^2 x^{k-1})dx \\ &\geq x^-(k,r) - kr^2(x^-(k,r))^k \\ &= w_e^*(k,r).\end{aligned}$$

By directly differentiating $w_e^*(k,r)$ with respect to k and r, we can check that $w_e^*(k,r)$ is increasing with k for $k \geq 3$ and decreasing with r for $r \in (r_{\text{clt}}(k), r_{\text{sat}}(k))$. Therefore, we have $w_e(k,r) \geq w_e^*(k_0, r_{\text{sat}}(k_0))$. □

5 Future direction

We sketch out some ideas on the future development based on this thesis. One is to extend our result to the Unit Clause Algorithm `UCA`. The other one is to extend the result to random NAE-k-SAT.

5.1 Unit clause algorithm

In this section, we outline the idea of extending our results to the Unit Clause algorithm `UCA`. Recall that `UCA` runs in the following way: Recursively, assign a suitable value to the variable in a unit clause to satisfy the unit clause if exists, or assign a randomly chosen value to a randomly chosen unassigned variable otherwise. In Section 1.3, we discussed the difference between the `UC`-decimation $\texttt{DEC}_{\texttt{UC}}$ and the Unit Clause algorithm `UCA`, which is their strategies on choosing variables. In each iteration, the `UC`-decimation $\texttt{DEC}_{\texttt{UC}}$ chooses a variable uniformly at random from all unassigned variables, while the Unit Clause algorithm `UCA` chooses a variable involving in a unit clause if such a variable exists, or uniformly at random from all unassigned variable otherwise. Informally speaking, `UC`-decimation $\texttt{DEC}_{\texttt{UC}}$ chooses variable in a completely random manner, while the Unit Clause algorithm `UCA` prefers variables in unit clauses first if they exist.

As we mentioned in Section 2.1, we need the algorithm to have two properties, in order to apply the OGP-based approach based on the basic form of OGP. We have to verify that (i) the algorithm is insensitive to its internal randomness, and (ii) dependent to its internal randomness. Moreover, in our argument for the sequential local algorithms, (ii) can be verified by approximating the number of free steps.

Let's consider the requirement (ii) first, namely, find the number of free steps run by `UCA`. In the t-th step, the action taken by the `UCA` depends on the existence of unit clauses in the residue instance Φ_t. The existence of unit clauses could depend on the random choice of unassigned variable in the last executed free step. Therefore, it is not easy to predict whether a particular step is a free step or a forced step. To overcome this difficulty, we first make the following three observations.

1. The first iteration must be a free step, since a k-XORSAT instance does not contain any unit clause in the beginning of the process.
2. After a free step, if there is one or more unit clauses, then the algorithm runs (possibly multiple) forced steps until no unit clause is left.
3. After (2), since no unit clause is left, the next iteration must be a free step.

Now we can see that the algorithm basically repeats the process described in (2). Then, we can bundle a free step and the forced steps following it together as a single step such that each bundled step performs the same action. Therefore, we re-write the Unit Clause algorithm `UCA` as follows.

Algorithm 4 Unit Clause algorithm `UCA`

```
1: Input: $k$-XORSAT instance $\Phi$
2: Set $\Phi_0 = \Phi$ and $t = 0$.
3: while $\Phi_t$ is not an empty instance do
4:     Select an unassigned variable $x^*$ from $\Phi_t$, uniformly at random.
5:     Set $\sigma(x^*) = \begin{cases} 1 & \text{with probability } 1/2 \\ 0 & \text{with probability } 1/2 \end{cases}$.
6:     Obtain $\Phi'_t$ from $\Phi_t$ by:
           (i) Remove $x^*$.
           (ii) For clauses having $x^*$ before (i), add $\sigma(x^*)$ to its right-hand-side value.
           (iii) Remove all clauses that no longer contain any variable.
7:     while there exists at least one unit clause do
8:         for each of such unit clause $x_i = b_j$ do
9:             Set $\sigma(x_i) = b_j$.
10:            Update $\Phi'_t$ by:
                   (i) Remove $x_i$.
                   (ii) For clauses having $x_i$ before (i), add $\sigma(x_i)$ to its right-hand-side value.
                   (iii) Remove all clauses that no longer contain any variable.
11:        end for
12:    end while
13:    Set $\Phi_{t+1} = \Phi'_t$.
14:    $t \leftarrow t + 1$.
15: end while
16: Output: assignment $\sigma$
```

Now, each iteration (Line 4-14) consists one free step (Line 4-6), and zero or more forced steps (Line 7-14). To avoid ambiguity of the word "step", we call such an iteration **a bundle step**, which reflects the fact that it may consist of multiple (free and forced) steps. Since each bundled step contains exactly one free step, the number of bundled steps is equal to the number of free steps. The number of bundled steps could be approximated by Wormald's method of differential equations, similar to the proofs in Section 4.3 and Section 4.5.

Now, we consider the requirement (i). After applying the Wormald's method of differential equations, we should have the degree profile of the factor graph of Φ_t at each bundled step. Then, we can approximate the average number of variables involving in a unit clause, newly created by a free step or a forced step. This average number should be smaller than one for all bundled steps except the last one. It is because having this average number greater than one for a bundled step implies the number of free steps in that bundled step grows to the infinite when the forced steps "propagate" away from the variable x^* selected by the free step, and those forced steps assign values to all remaining variables to end the algorithm. Furthermore, having this average number smaller than one for a bundled step implies the free steps in that bundled step cannot "propagate" far away from the selected variable x^*. Therefore, all variables assigned values by free steps must have distance at most R from x^*, for some large constant R, independent of n. Then, we can treat a bundled step as a local rule with radius R, and apply Lemma 6 to achieve (i). With (i) and (ii), we can apply our new OGP-based approach to show that w.h.p. `UCA` cannot find solutions for a random k-XORSAT instance.

5.2 Random NAE-k-SAT

We are interested in applying our new OGP-based approach with OGP of sub-instances, to the random NAE-k-SAT problem. Here are some developing ideas and some heuristic calculation to test the ideas.

The **not-all-equal k-satisfaction problem** (NAE-k-SAT) is a Boolean constraint satisfaction problem closely analogous to the k-SAT problem. In a NAE-k-SAT instance, each clause contains literals of k variables, where each literal is either one of the n variables or a negation of one of the n variables. A clause is satisfied by an assignment $\sigma : \{x_i : 1 \le i \le n\} \to \{0, 1\}$ if at least one of its literals is assigned 1 and at least one of its literals is assigned 0. In the other words, a clause is satisfied by σ if all of its literals are not assigned the same value. For a random NAE-k-SAT instance, each clause is drawn over the $2^k \binom{n}{k}$ possibilities uniformly at random, independently.

The satisfiability threshold $r_{\text{sat}}(k)$ of random NAE-k-SAT was established by Coja-Oghlan and Panagiotou [CP12], and they determined $r_{\text{sat}}(k) = 2^{k-1} \ln 2 - \ln 2/2 - 1/4 - o_k(1)$ asymptotically. Moreover, Ding, Sly and Sun [DSS14] precisely determined the satisfiability threshold $r_{\text{sat}}(k, d)$ of random d-regular NAE-k-SAT problem in which each variable involves in exactly d clauses, by using the notion of *frozen configuration* to represent clusters. Similar to other random CSPs, all known polynomial-time algorithms for random NAE-k-SAT works only when the clause density is much below the satisfiability threshold. For random NAE-k-SAT, the best known polynomial-time algorithm is a very simple algorithm, Unit Clause Algorithm, which can find solutions w.h.p. when the clause density r is below $\rho 2^{k-1}/k$ for some universal constant $\rho > 0$ for sufficiently large k [AKKT02]. There is a statistical-to-computation gap of a multiplicative factor k. Gamarnik and Sudan used the OGP-based approach with m-OGP to show that a natural class of algorithms, the balanced sequential local algorithms, fails to find solutions w.h.p. for density $r > (1 + o_k(1))2^{k-1}\frac{\ln^2 k}{k}$ for sufficiently large k. There is still a short of a multiplicative factor $\ln^2 k$ with the best known algorithms.

We are interested in closing this $\ln^2 k$ gap by applying the new OGP-based approach with OGP of sub-instances. To do that, we have to pick a suitable sub-instance so that after removal of the variables not in the sub-instance, two solutions of the NAE-k-SAT instance become two solutions of the sub-instance, and the two solutions of the sub-instance are either close to each other or far from each other. The obvious candidate is the core instance obtained by applying the peeling algorithm, which removes all variables of degree at most 1. Unfortunately, it is not a suitable one for random NAE-k-SAT, with the following counterexample. Assume we have an instance, which has a clause c containing $x_1, x_2, \overline{x_3}$. We further assume x_3 is of degree 1, and σ is an assignment with $\sigma(x_1) = \sigma(x_2) = \sigma(x_3) = 0$. Obviously, σ satisfies the clause c. However, it is easy to see that the peeling algorithm removes the variable x_3, so in the core instance the assignment σ with $\sigma(x_1) = \sigma(x_2) = 0$ does not satisfy the resultant clause c' only containing x_1 and x_2.

To design a new variable removal strategy to obtain a suitable sub-instance for random NAE-k-SAT, apart from what the peeling algorithm does, we make two following observation. The first observation is that the variable removal strategy of the peeling algorithm only considers the factor graph structure. To avoid the situation we have in the above example, the new variable removal strategy should also consider the solutions we concern. The second observation is that the peeling algorithm on k-XORSAT instances precisely removes variables so that a cluster in the solution space "collapses" into a single core solution or a core cluster with $o(n)$ diameter. This provides the sparse structure in the solution space of the sub-instance so that the sub-instance exhibits OGP.

From these observations, the next candidate of variable removal strategy is to remove variables that all clauses are still satisfied by the two solutions after variable removal, where the two solutions are provided beforehand. This strategy guarantees that after variable removal the two solutions of the whole instance becomes two solutions for the sub-instance, to avoid the situation we have in the counterexample for peeling algorithm on random NAE-k-SAT. Moreover, this strategy can combine every pair of solutions of distance 1 between them, into a single solution of the sub-instance, which matches the second observation above.

Suppose we have a random NAE-k-SAT instance $\mathbf{\Phi}$ of n variables and rn clauses, and two assignments σ_1, σ_2 of distance αn between them. Let x_0 be an arbitrary variable in $\mathbf{\Phi}$. We know that x_0 is involved in i clauses with probability $e^{-kr}\frac{(kr)^i}{i!} + o(1)$. For each clause c involving

x_0, each of the other $k-1$ variables is assigned the same value by σ_1 and σ_2 with probability $(1-\alpha)^{k-1}+o(1)$. Therefore, for each clause c involved by x_0, the probability that c is satisfied by both σ_1 and σ_2 without the variable x_0 is given by

$$\begin{aligned}&(1-\alpha)^{k-1}\cdot\frac{2^{k-1}-2}{2^{k-1}}+(1-(1-\alpha)^{k-1})\cdot\frac{2^{k-1}-4}{2^{k-1}}\\&=1-2^{2-k}(2-(1-\alpha)^{k-1}).\end{aligned}$$

Therefore, x_0 is immediately removed by this new variable removal strategy with probability

$$\begin{aligned}&\sum_{i\geq 0}e^{-kr}\frac{(kr)^i}{i!}\cdot\left[1-2^{2-k}(2-(1-\alpha)^{k-1})\right]^i\\&=\sum_{i\geq 0}e^{-kr}\frac{\left[kr\left(1-2^{2-k}(2-(1-\alpha)^{k-1})\right)\right]^i}{i!}\\&\approx e^{-kr\left(1-2^{2-k}(2-(1-\alpha)^{k-1})\right)-kr}\\&=e^{-rk2^{2-k}(2-(1-\alpha)^{k-1})}\\&\to\begin{cases}1 & \text{if } r=o_k\left(\frac{2^{k-2}}{k}\frac{1}{2-(1-\alpha)^{k-1}}\right)\\ 0 & \text{if } r=\omega_k\left(\frac{2^{k-2}}{k}\frac{1}{2-(1-\alpha)^{k-1}}\right)\end{cases}\qquad\text{as } k\to\infty\end{aligned}$$

From this calculation, it is likely that this sub-instance emerges at density $r=\frac{2^{k-2}}{k}\frac{1}{2-(1-\alpha)^{k-1}}$, which is close to the believed clustering threshold. It is analogous to the behaviour of the core instance for random k-XORSAT. If we can further show that this sub-instance exhibits OGP, it might be possible to apply our new OGP-based approach to determine the clustering threshold and algorithmic threshold of random NAE-k-SAT.

A Phase transition of random k-SAT

Satisfiability threshold Scientists spent decades to search the satisfiability threshold of the random k-SAT, which became a long-standing open problem in probabilistic combinatorics, theoretical computer science and statistical physics. The *Satisfiability Threshold Conjecture* claims that there exists $r_{\text{sat}}(k)>0$ such that

- for $r<r_{\text{sat}}(k)$, the random k-SAT instance $\mathbf{\Phi}$ is satisfiable with high probability;
- for $r>r_{\text{sat}}(k)$, the random k-SAT instance $\mathbf{\Phi}$ is unsatisfiable with high probability.

The satisfiability threshold $r_{\text{sat}}(k)$ of the random k-SAT for $k=2$ was found decades ago, with $r_{\text{sat}}(2)=1$, independently by Chvátal and Reed [CR92], Goerdt [Goe96], and Fernandez de la Vega [FDLV01]. However, finding the satisfiability threshold for general $k\geq 3$ was more complicated. The first moment method was used long time ago by Franco and Paull [FP83] to yield an upper bound on the conjectured threshold $r_{\text{sat}}(k)$. Kirousis, Kranakis, Krizanc and Stamatiou [KKKS98] improved the argument to show a better upper bound $r_{\text{sat}}(k)\leq 2^k\ln 2-\frac{1+\ln 2}{2}+o_k(1)$.

Friedgut [Fri99] made a breakthrough by proving the existence of an n-dependent satisfiability threshold $r_{\text{sat}}(k,n)$ for the random k-SAT (as well as other random constraint satisfiability problems), which means that there exists a sequence of number of $r_{\text{sat}}(k,n)$ depending on n such that

- for $r < (1-\epsilon)r_{\text{sat}}(k,n)$, a random k-SAT instance is satisfiable with high probability;
- for $r > (1+\epsilon)r_{\text{sat}}(k,n)$, a random k-SAT instance is unsatisfiable with high probability;

for some small $\epsilon > 0$. Although this result does not give an n-independent threshold, we can define the bounds on the conjectured threshold as follows:

$$\begin{aligned} r^*(k) &= \liminf_{n\to\infty}\{r : \text{A random } k\text{-SAT instance } \mathbf{\Phi}(n,rn) \text{ is unsatisfiable w.h.p.}\} \quad \text{and} \\ r_*(k) &= \limsup_{n\to\infty}\{r : \text{A random } k\text{-SAT instance } \mathbf{\Phi}(n,rn) \text{ is satisfiable w.h.p.}\}. \end{aligned}$$

It is clear that $r_*(k) \le r_{\text{sat}}(k,n) \le r^*(k)$ for sufficiently large n. If the sharp n-independen satisfiability threshold $r_{\text{sat}}(k)$ exists, then we have $r_*(k) = r_{\text{sat}}(k) = r^*(k)$ for sufficiently large n. From [KKKS98] mentioned above, we know $r^*(k) \le 2^k \ln 2 - \frac{1+\ln 2}{2} + o_k(1)$.

On the other hand, Frieze and Wormald [FW05] started using the second moment method to yield the lower bound on $r_*(k)$. Achlioptas and Moore [AM02] showed that $r_*(k) \ge 2^{k-1}\ln 2 - d_k$, where $d_k \to (1+\ln 2)/2$, which is below the upper bound by a factor of 2. After that, Achlioptas and Peres [AP04] gave a better lower bound $r_*(k) \ge 2^k \ln 2 - (k+1)\frac{\ln 2}{2} + o_k(1)$, which narrows down the additive gap between the upper bound and the lower bound to $\frac{k\ln 2}{2} + \frac{1}{2} + o_k(1)$. By the non-rigorous argument, the *1-step replica symmetry breaking (1RSB)* of the *cavity method* from physicists, the satisfiability threshold $r_{\text{sat}}(k)$ was predicted [MMZ06] to be

$$r_{\text{sat}}(k) = 2^k \ln 2 - \frac{1+\ln 2}{2} + o_k(1).$$

Eventually, Coja-Oghlan and Panagiotou [CP16] closed the asymptotic gap by showing that

$$2^k \ln 2 - \frac{1+\ln 2}{2} - \epsilon_k \le r_*(k) \quad \text{and} \quad r^*(k) \le 2^k \ln 2 - \frac{1+\ln 2}{2} + \epsilon_k,$$

where these inequalities hold with $\epsilon_k = 2^{-k/2+o(k)}$. Ding, Sly and Sun [DSS15] further showed that the sharp satisfiability threshold $r_{\text{sat}}(k)$ does exist, and its value matches the 1RSB prediction, for any $k \ge k_0$, where k_0 is some constant greater than 3.

Algorithmic threshold The satisfiability threshold of random k-SAT is located by *non-constructive* proofs, which means they only guarantee the existence of solutions when the clause density is below the threshold, without providing an efficient way to construct solutions. Over the decades, numerous algorithms were introduced to find solutions for satisfiable instances efficiently, which works pretty well when the clause density is relatively low. Preciously speaking, many algorithms can solve random k-SAT w.h.p. for $r < \rho\frac{2^k}{k}$ with some universal constant

$\rho > 0$. For example, based on the unit-clause heuristic and its generalization, Chao and Franco [CF86, CF90] introduced the algorithm UC that can solve random 3-SAT with uniformly positive probability for

$$r < \frac{1}{2}\left(\frac{k-1}{k-2}\right)^{k-2}\frac{2^k}{k},$$

and the algorithm GUC that can solve random k-SAT with high probability for

$$r < 0.46125\left(\frac{k-1}{k-2}\right)^{k-2}\frac{2^k}{k+1} - 1$$

and $4 \leq k \leq 40$. Based on their works, Chvátal and Reed [CR92] introduced the algorithm SC that can solve random k-SAT with high probability for

$$r < \frac{1}{8}\left(\frac{k-1}{k-3}\right)^{k-3}\frac{k-1}{k-2}\cdot\frac{2^k}{k}$$

for general $k \geq 3$. However, none of them can find a solution in polynomial time when the density is close to the satisfiability threshold. To make matters worse, some algorithms are known to fails in solving random k-SAT. For example, a family of DPLL-type algorithms cannot solve random k-SAT in linear-time with uniformly positive probability for $k \geq \left(\frac{11}{4}\right)\frac{2^k}{k}$ and $k \geq 4$ [ABM04]. Up to date, the best known algorithm from Coja-Oghlan [Coj10] succeeds w.h.p. for $r \leq (1-\epsilon_k)\frac{2^k}{k}\ln k$ where $\epsilon_k \to 0$, and seems to fail beyond. There is a multiplicative gap $\frac{k}{\ln k}$ between the satisfiability threshold and the largest density where algorithms are known to solve a random instance with non-vanishing probability. In the other words, in the regime of $\frac{2^k}{k}\ln k < r < 2^k \ln 2 - \frac{1+\ln 2}{2} + o_k(1) = r_{\text{sat}}(k)$, called the *hard-SAT phase*, we know a random instance has solutions w.h.p., but we do not have an efficient way to find a solution. One could conjecture that there is an **algorithmic threshold** $r_{\text{alg}}(k)$ such that for $r_{\text{alg}}(k) < r < r_{\text{sat}}(k)$ no polynomial-time algorithm can find a solution of random k-SAT with non-vanishing probability.

Clustering threshold In order to understand the algorithmic obstacles, there is another line of research that attempts to analyse the geometric structures of the solution spaces and suspect that the geometric structures could be the answer. Mézard, Mora and Zecchina [MMZ05] first observed that there exists a range of clause density for which w.h.p. there is no pair of solutions at distance αn for any $\alpha \in I$ for some $I \subset (0,1)$. This was the first evidence of the disconnectivity of the solution space. Daudé, Mézard, Mora and Zecchina [DMMZ08] further proved that there is an n-dependent threshold separating the regime with this disconnectivity and the regime without this disconnectivity.

To concretely describe the disconnectivity of the geometric structure of a solution space, we can convert the solution space into a graph, by adding an edge between two solutions if the Hamming distance between them is $o(1)$. In this graph, the **clusters** are the connected components, and a **cluster region** is defined to be a non-empty union of clusters.

Under these definitions, Achlioptas, Coja-Oghlan and Ricci-Tersenghi [ACR11] proved that, for large k and clause density sufficiently close to the satisfiability threshold $r_{\text{sat}}(k)$, w.h.p. the solution space of a random k-SAT instance can be partitioned into an exponential number of non-empty cluster regions, and the distance between every pair of cluster regions is at least $\beta_k n$, for some constant $\beta_k < 1/2$. In another paper, Achlioptas and Coja-Oghlan [AC08] showed that the same thing happens for

$$(1+\epsilon_k)\frac{2^k}{k}\ln k \le r \le (1-\epsilon_k)2^k \ln 2$$

for some sequence $\epsilon_k \to 0$, and conjectured that for density below $(1-\epsilon_k)\frac{2^k}{k}\ln k$ w.h.p the solution space is a giant component. This leads us to the **clustering threshold** $r_{\text{clt}}(k)$ which states that

- for $r < r_{\text{clt}}(k)$, the solution space is a giant connected component;
- for $r_{\text{clt}}(k) < r < r_{\text{sat}}(k)$, the solution space can be partitioned into an exponential number of well-separated cluster regions;

[AC08] suggested that the clustering threshold $r_{\text{clt}}(k)$ is located at $\frac{2^k}{k}\ln k$, which is asymptotically equal to the algorithmic threshold of random k-SAT, coincidently. It is natural to suspect that this sharp change of geometric structure of the solution space is the related to the algorithmic hardness of random k-SAT.

B Proof of Lemma 1 and 2

Lemma 1. Let $Z = Z(\alpha n)$ be the number of pairs of solutions, with distance αn between the two solutions, for a random k-XORSAT instance. In the other words, given a random instance with linear system representation $Ax = b$,

$$Z = \sum_{\substack{\sigma,\sigma' \in \{0,1\}^n \\ d(\sigma,\sigma')=\alpha n}} \mathbb{1}(A\sigma = b \text{ and } A\sigma' = b).$$

By linearity of expectation, the expected value of Z is given by

$$\mathbb{E}[Z] = \sum_{\substack{\sigma,\sigma' \in \{0,1\}^n \\ d(\sigma,\sigma')=\alpha n}} \Pr[\, A\sigma = b \text{ and } A\sigma' = b \,].$$

Now we consider the calculation of $\Pr[\, A\sigma = b \text{ and } A\sigma' = b \,]$. Since each equation in the linear system are chosen identically and independently, the summand can be written as

$$(\Pr[\, A_1\sigma = b_1 \text{ and } A_1\sigma' = b_1 \,])^{rn},$$

where $A_1x = b_1$ is the first equation in the linear system. In addition, by condition probability formula we have

$$\begin{aligned}\Pr[\,A\sigma = b \text{ and } A\sigma' = b\,] &= (\Pr[\,A_1\sigma = b_1 \;\text{ and }\; A_1\sigma' = b_1\,])^{rn} \\ &= (\Pr[\,A_1\sigma = b_1 \;\text{ and }\; A_1\sigma = A_1\sigma'\,])^{rn} \\ &= (\Pr[\,A_1\sigma = b_1 \mid A_1\sigma = A_1\sigma'\,]\Pr[\,A_1\sigma = A_1\sigma'\,])^{rn} \\ &= \left(\frac{1}{2}\Pr[\,A_1\sigma = A_1\sigma'\,]\right)^{rn}\end{aligned}$$

Since $d(\sigma, \sigma') = \alpha n$, there are αn variables having different values and $(1-\alpha)n$ variables having same values when we compare the assignments σ and σ'. The random equation $A_1\sigma = A_1\sigma'$ holds if and only if the equation chooses even number of variables from those αn variables having different values in σ and σ'. So, we have

$$\begin{aligned}&\Pr[\,A_1\sigma = A_1\sigma'\,] \\ &= \sum_{i=0}^{\lfloor k/2 \rfloor} \Pr[\,2i \text{ variables with different values in } \sigma \text{ and } \sigma' \text{ are chosen by } A_1\,] \\ &= \sum_{i=0}^{\lfloor k/2 \rfloor} \frac{\binom{\alpha n}{2i}\cdot\binom{(1-\alpha)n}{k-2i}}{\binom{n}{k}} \\ &= \sum_{i=0}^{\lfloor k/2 \rfloor} \binom{k}{2i}\alpha^{2i}(1-\alpha)^{k-2i} \\ &= \frac{1}{2}(1 + (1-2\alpha)^k)\end{aligned}$$

From above formula, we can see that the value of $\Pr[\,A_1\sigma = A_1\sigma'\,]$ is independent of the choices of σ and σ', and only depends on the distance between σ and σ'. So we have

$$\begin{aligned}\mathbb{E}[Z] &= \sum_{\substack{\sigma,\sigma' \in \{0,1\}^n \\ d(\sigma,\sigma') = \alpha n}} \Pr[\,A_1\sigma = b_1 \;\text{ and }\; A_1\sigma' = b_1\,]^{rn} \\ &= \sum_{\substack{\sigma,\sigma' \in \{0,1\}^n \\ d(\sigma,\sigma') = \alpha n}} \left(\frac{1}{2}\Pr[\,A_1\sigma = A_1\sigma'\,]\right)^{rn} \\ &= 2^n \binom{n}{\alpha n}\left(\frac{1 + (1-2\alpha)^k}{4}\right)^{rn}\end{aligned}$$

By applying the Stirling's approximation for $\binom{n}{\alpha n}$ we have

$$\begin{aligned}\mathbb{E}[Z] &= 2^n \frac{1}{\sqrt{2\pi n}}\frac{1}{\sqrt{\alpha(1-\alpha)}}\left(\frac{1}{\alpha^\alpha(1-\alpha)^{1-\alpha}}\right)^n \left(\frac{1 + (1-2\alpha)^k}{4}\right)^{rn} + o(n) \\ &= \frac{1}{\sqrt{2\pi n}}\frac{1}{\sqrt{\alpha(1-\alpha)}}\left(\frac{2}{\alpha^\alpha(1-\alpha)^{1-\alpha}}\left(\frac{1 + (1-2\alpha)^k}{4}\right)^r\right)^n + o(n) \\ &= \frac{1}{\sqrt{2\pi n}}\frac{1}{\sqrt{\alpha(1-\alpha)}} f(k, r, \alpha)^n + o(n)\end{aligned}$$

where $f(k, r, \alpha)$ is defined by

$$f(k, r, \alpha) \equiv \frac{2}{\alpha^\alpha(1-\alpha)^{1-\alpha}} \left(\frac{1 + (1-2\alpha)^k}{4} \right)^r .$$

For convenience, we simply assume $\alpha^\alpha(1-\alpha)^{1-\alpha} = 1$ when $\alpha = 0$ or $\alpha = 1$. □

Fix $k \geq 3$ and $r > 0$. If $f(k, r, \alpha') < 1$ for some $\alpha' \in [0, 1]$, then the expectation $\mathbb{E}[Z(\alpha' n)]$ converges to 0 as $n \to \infty$. By Markov's inequality, we have $\Pr[Z(\alpha' n) > 0] \leq \mathbb{E}[\alpha' n]$, and thus $\Pr[Z(\alpha' n) > 0]$ also converges to 0 as $n \to \infty$. That means the probability of having at least one pair of solutions with distance $\alpha' n$ between them converges to 0. In the other words, w.h.p. there is no such pair of solutions. So, if we can find an interval $(u_1, u_2) \subset [0, 1]$ such that $f(k, r, \alpha) < 1$ for any $\alpha \in (u_1, u_2)$, we can say that w.h.p. there is no pair of solutions with distance αn between them, for any $\alpha \in (u_1, u_2)$. In such case, w.h.p. the distance between every pair of solutions is either $\leq u_1 n$, or $\geq u_2 n$, that is, w.h.p. the random instance $\mathbf{\Phi}$ exhibits the overlap gap property with $v_1 = u_1 n$ and $v_2 = u_2 n$.

From the formula of the function f, we can see that it decreases with r, which is also visualized in Figure 3. It is more likely to have the OGP if the clause density r is large. We can further determine the minimal clause density $r_1(k)$ for having the OGP. In particular, we have the following lemma.

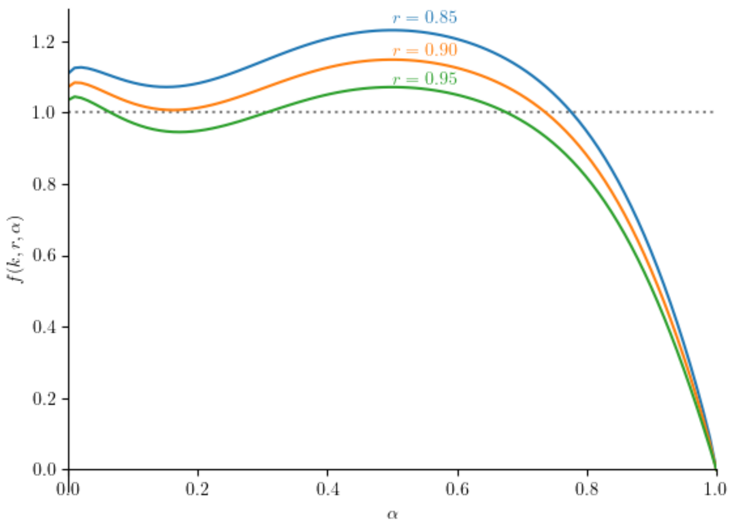


Figure 3: The graph of $f(k, r, \alpha)$ against α, with $k = 3$ and different values of r: (1) $r = 0.85$ in blue at the top, (2) $r = 0.90$ in brown in the middle, and (3) $r = 0.95$ in green at the bottom.

Proof of Lemma 2. If k is odd, then we have $f(k, r, 1) = 0$. In this case, by the continuity of f, there exist $u_1 < \frac{1}{2} \leq u_2 = 1$ such that $f(k, r, \alpha) < 1$ for any $\alpha \in (u_1, u_2)$.

Now we assume k is even. When $r = 0$, it is easy to see that $f(k, r, \alpha) = \frac{2}{\alpha^\alpha(1-\alpha)^{1-\alpha}} > 1$ for any $\alpha \in [0, 1]$. If we fix k and α, then the function f is strictly decreasing with r since

$(1+(1-2\alpha)^k)/4 < 1$. We aim at finding the minimal positive value of r that guarantees there exists an interval $(u_1, u_2) \subset [0,1]$ such that $f(k,r,\alpha) < 1$ for any $\alpha \in (u_1, u_2)$. Furthermore, we know that $f(k,r,\alpha) = f(k,r,1-\alpha)$ for even $k \geq 3$, so it suffices to find the minimal positive value of r that guarantees the existence of such interval (u_1, u_2) within $[0, \frac{1}{2}]$.

By fixing even $k \geq 3$ and $\alpha \in [0, \frac{1}{2}]$, we can treat f as a strictly decreasing continuous function $f_{k,\alpha}(r)$ of r, with $f_{k,\alpha}(0) = 2 > 1$ and $\lim_{r\to\infty} f_{k,\alpha}(r) = 0 < 1$. Therefore, for any even $k \geq 3$ and $\alpha \in [0, \frac{1}{2}]$, there exists a unique $r^*(k,\alpha) > 0$ given by

$$r^*(k,\alpha) = \frac{1 + H(\alpha)}{2 - \log_2(1 + (1-2\alpha)^k)},$$

such that

$$\begin{aligned}
f_{k,\alpha}(r) &> 1 \quad \text{for } r < r^*(k,\alpha),\\
f_{k,\alpha}(r) &= 1 \quad \text{for } r = r^*(k,\alpha), \text{ and}\\
f_{k,\alpha}(r) &< 1 \quad \text{for } r > r^*(k,\alpha)
\end{aligned}$$

where H is the binary entropy function $H(x) = -x\log_2(x) - (1-x)\log_2(1-x)$. Then, we can define $r_1(k)$ and $\alpha_1(k)$ by

$$r_1(k) = \min_{0\leq\alpha\leq\frac{1}{2}} r^*(k,\alpha) \quad \text{and} \quad \alpha_1(k) = \arg\min_{0\leq\alpha\leq\frac{1}{2}} r^*(k,\alpha)$$

with $r^*(k,\alpha_1(k)) = r_1(k)$. Suppose $r > r_1(k)$. Since f is strictly decreasing with r, we have $f(k,r,\alpha_1(k)) < f(k,r_1(k),\alpha_1(k)) = f_{k,\alpha_1(k)}(r_1(k)) = f_{k,\alpha_1(k)}(r^*(k,\alpha_1(k))) = 1$. By the continuity of f, there exist $0 \leq u_1 < \alpha_1(k)$ and $\alpha_1(k) < u_2 \leq \frac{1}{2}$ such that we have $f(k,r,\alpha) < 1$ for any $\alpha \in (u_1, u_2)$.

Therefore, with Lemma 1, we know that the expected number $\mathbb{E}[Z(\alpha n)]$ of pairs of solutions with distance αn between them converges to zero for any $\alpha \in (u_1, u_2)$. By the first moment method, w.h.p. there is no pair of solutions with distance αn between them, for any $\alpha \in (u_1, u_2)$. In the other words, w.h.p. for any pair of solutions σ and σ' of $\mathbf{\Phi}$ the distance $d(\sigma, \sigma')$ between them is either smaller than $u_1 n$ or larger than $u_2 n$ for some u_1 and u_2 with $0 \leq u_1 < u_2$. $\square$

References

[ABM04] Dimitris Achlioptas, Paul Beame, and Michael Molloy. Exponential Bounds for DPLL Below the Satisfiability Threshold. *Proceedings of the fifteenth annual ACM-SIAM symposium on discrete algorithms*, pages 139–140, 2004.

[AC08] 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.

[ACR11] 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.

[AKKT02] Dimitris Achlioptas, Jeong Han Kim, Michael Krivelevich, and Prasad Tetali. Two-coloring random hypergraphs. *Random Structures and Algorithms*, 20(2):249–259, March 2002.

[ALS22a] 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.

[ALS22b] 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.

[AM02] D. Achlioptas and C. Moore. The asymptotic order of the random k-SAT threshold. In *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, pages 779–788, Vancouver, BC, Canada, 2002. IEEE Comput. Soc.

[AM15] Dimitris Achlioptas and Michael Molloy. The solution space geometry of random linear equations. *Random Structures & Algorithms*, 46(2):197–231, March 2015.

[AP04] Dimitris Achlioptas and Yuval Peres. The threshold for random k-SAT is 2ˆk log 2 - O(k). *Journal of the American Mathematical Society*, 17(4):947–973, August 2004.

[AR06] Dimitris Achlioptas and Federico Ricci-Tersenghi. On the solution-space geometry of random constraint satisfaction problems. In *Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing - STOC '06*, page 130, Seattle, WA, USA, 2006. ACM Press.

[BH22] 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.

[BMZ05] A. Braunstein, M. Mézard, and R. Zecchina. Survey propagation: An algorithm for satisfiability. *Random Structures and Algorithms*, 27(2):201–226, September 2005.

[BZ04] Alfredo Braunstein and Riccardo Zecchina. Survey propagation as local equilibrium equations. *Journal of Statistical Mechanics: Theory and Experiment*, 2004(06):P06007, June 2004.

[CF86] Ming-Te Chao and John Franco. Probabilistic Analysis of Two Heuristics for the 3-Satisfiability Problem. *SIAM Journal on Computing*, 15(4):1106–1118, November 1986.

[CF90] Ming-Te Chao and John Franco. Probabilistic analysis of a generalization of the unit-clause literal selection heuristics for the k satisfiability problem. *Information Sciences*, 51(3):289–314, August 1990.

[Coj10] Amin Coja-Oghlan. A Better Algorithm for Random k -SAT. *SIAM Journal on Computing*, 39(7):2823–2864, January 2010.

[Coj11] Amin Coja-Oghlan. On belief propagation guided decimation for random k-SAT. *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 957–966, January 2011.

[Coj17] Amin Coja-Oghlan. Belief Propagation Guided Decimation Fails on Random Formulas. *Journal of the ACM*, 63(6):1–55, February 2017.

[Coo04] Colin Cooper. The cores of random hypergraphs with a given degree sequence. *Random Structures & Algorithms*, 25(4):353–375, December 2004.

[CP12] Amin Coja-Oghlan and Konstantinos Panagiotou. Catching the k-NAESAT threshold. In *Proceedings of the 44th Symposium on Theory of Computing - STOC '12*, page 899, New York, New York, USA, 2012. ACM Press.

[CP16] Amin Coja-Oghlan and Konstantinos Panagiotou. The asymptotic k-SAT threshold. *Advances in Mathematics*, 288:985–1068, January 2016.

[CR92] V. Chvátal and B. Reed. Mick gets some (the odds are on his side) (satisfiability). In *Proceedings., 33rd Annual Symposium on Foundations of Computer Science*, pages 620–627, Pittsburgh, PA, USA, 1992. IEEE.

[CW90] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280, March 1990.

[DGM$^+$10] Martin Dietzfelbinger, Andreas Goerdt, Michael Mitzenmacher, Andrea Montanari, Rasmus Pagh, and Michael Rink. Tight Thresholds for Cuckoo Hashing via XOR-SAT. In *Automata, Languages and Programming*, volume 6198, pages 213–225. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[DM02] Olivier Dubois and Jacques Mandler. The 3-XORSAT threshold. *Comptes Rendus Mathematique*, 335(11):963–966, December 2002.

[DMMZ08] Hervé Daudé, Marc Mézard, Thierry Mora, and Riccardo Zecchina. Pairs of SAT-assignments in random Boolean formulæ. *Theoretical Computer Science*, 393(1-3):260–279, March 2008.

[DSS14] Jian Ding, Allan Sly, and Nike Sun. Satisfiability threshold for random regular NAE-SAT. In *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*, pages 814–822, New York New York, May 2014. ACM.

[DSS15] Jian Ding, Allan Sly, and Nike Sun. Proof of the Satisfiability Conjecture for Large k. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, pages 59–68, Portland Oregon USA, June 2015. ACM.

[FDLV01] W. Fernandez De La Vega. Random 2-SAT: Results and problems. *Theoretical Computer Science*, 265(1-2):131–146, August 2001.

[FP83] John Franco and Marvin Paull. Probabilistic analysis of the Davis Putnam procedure for solving the satisfiability problem. *Discrete Applied Mathematics*, 5(1):77–87, January 1983.

[Fri99] Ehud Friedgut. Sharp thresholds of graph properties, and the \$k\$-sat problem. *Journal of the American Mathematical Society*, 12(4):1017–1054, May 1999.

[FW05] Alan Frieze and Nicholas C. Wormald. Random k-Sat: A Tight Threshold For Moderately Growing k. *Combinatorica*, 25(3):297–305, May 2005.

[Gam21] 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.

[GK23] David Gamarnik and Eren C. Kızıldağ. Algorithmic obstructions in the random number partitioning problem. *The Annals of Applied Probability*, 33(6B), December 2023.

[GKPX23] 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.

[GL18] David Gamarnik and Quan Li. Finding a large submatrix of a Gaussian random matrix. *The Annals of Statistics*, 46(6A), December 2018.

[Goe96] Andreas Goerdt. A Threshold for Unsatisfiability. *Journal of Computer and System Sciences*, 53(3):469–486, December 1996.

[GS02] Carla P. Gomes and Bart Selman. Satisfied with Physics. *Science*, 297(5582):784–785, August 2002.

[GS17a] David Gamarnik and Madhu Sudan. Limits of local algorithms over sparse random graphs. *The Annals of Probability*, 45(4), July 2017.

[GS17b] 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.

[GY11] 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.

[Het16] Samuel Hetterich. Analysing Survey Propagation Guided Decimation on Random Formulas, February 2016.

[IKKM12] 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.

[Kar76] Richard M. Karp. The Probabilistic Analysis of some Combinational Search Algorithms. Technical Report UCB/ERL M581, EECS Department, University of California, Berkeley, 1976.

[Kim04] 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.

[KKKS98] Lefteris M. Kirousis, Evangelos Kranakis, Danny Krizanc, and Yannis C. Stamatiou. Approximating the unsatisfiability threshold of random formulas. *Random Structures and Algorithms*, 12(3):253–269, May 1998.

[KMR$^+$07] Florent Krząkała, Andrea Montanari, Federico Ricci-Tersenghi, Guilhem Semerjian, and Lenka Zdeborová. Gibbs states and the set of solutions of random constraint satisfaction problems. *Proceedings of the National Academy of Sciences*, 104(25):10318–10323, June 2007.

[Kol99] Valentin F. Kolčin. *Random Graphs.* Number 53 in Encyclopedia of Mathematics and Its Applications. Cambridge Univ. Press, Cambridge, 1. publ edition, 1999.

[KSS09] 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.

[MM09] Marc Mézard and Andrea Montanari. *Information, Physics, and Computation.* Oxford Graduate Texts. Oxford University Press, Oxford ; New York, 2009.

[MMW07] 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.

[MMZ05] M. Mézard, T. Mora, and R. Zecchina. Clustering of Solutions in the Random Satisfiability Problem. *Physical Review Letters*, 94(19):197205, May 2005.

[MMZ06] Stephan Mertens, Marc Mézard, and Riccardo Zecchina. Threshold values of random K -SAT from the cavity method. *Random Structures & Algorithms*, 28(3):340–373, May 2006.

[Mol05] Michael Molloy. Cores in random hypergraphs and Boolean formulas. *Random Structures and Algorithms*, 27(1):124–135, August 2005.

[MPZ02] M. Mézard, G. Parisi, and R. Zecchina. Analytic and Algorithmic Solution of Random Satisfiability Problems. *Science*, 297(5582):812–815, August 2002.

[MRZ03] 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.

[PS16] Boris Pittel and Gregory B. Sorkin. The Satisfiability Threshold for k -XORSAT. *Combinatorics, Probability and Computing*, 25(2):236–268, March 2016.

[PX21] 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.

[RS09] 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.

[Wei22] Alexander S. Wein. Optimal low-degree hardness of maximum independent set. *Mathematical Statistics and Learning*, 4(3):221–251, January 2022.

[Wor95] Nicholas C. Wormald. Differential Equations for Random Processes and Random Graphs. *The Annals of Applied Probability*, 5(4), November 1995.