On Percolation and $\mathcal{NP}$-Hardness

Daniel Reichman[*]   Igor Shinkar[†]

Abstract

We consider the robustness of computational hardness of problems whose input is obtained by applying independent random deletions to worst-case instances. For some classical $\mathcal{NP}$-hard problems on graphs, such as Coloring, Vertex-Cover, and Hamiltonicity, we examine the complexity of these problems when edges (or vertices) of an arbitrary graph are deleted independently with probability $1 - p > 0$. We prove that for n-vertex graphs, these problems remain as hard as in the worst-case, as long as $p > \frac{1}{n^{1-\epsilon}}$ for arbitrary $\epsilon \in (0, 1)$, unless $\mathcal{NP} \subseteq \mathcal{BPP}$.

We also prove hardness results for Constraint Satisfaction Problems, where random deletions are applied to clauses or variables, as well as the Subset-Sum problem, where items of a given instance are deleted at random.

1   Introduction

The theory of $\mathcal{NP}$-hardness suggests that we are unlikely to find optimal or near optimal solutions to $\mathcal{NP}$-hard problems in polynomial time. This theory applies to worst-case settings where one considers the worst running-time over the worst possible input. It is less clear whether these hardness results apply to "real-life" instances. One way to address this question is to examine to what extent known $\mathcal{NP}$-hardness results are stable under random perturbations, as it seems reasonable to assume that a given instance of a problem may be subjected to noise originating from multiple sources.

In this work we study *worst-case* instances that are subjected to *random perturbations* of a specific type, namely, random deletions. We focus on the following deletion process, known as *edge percolation*: given a graph G consider a random subgraph of G obtained by deleting each edge of G *independently* with probability $1-p$ (where $p \in (0,1)$ may depend on the size of the instance). This model generalizes familiar random graph models such as the Erdös-Rényi random graph $G(n,p)$. Instead of focusing on deleting edges at random from the complete graph, our starting graph G may be chosen arbitrarily out of all n-vertex graphs. Then the edges of G are deleted independently with probability $1-p$. The case of *vertex* percolation, where the vertices of a given graph are deleted independently at random is examined as well. We also study random deletions in other $\mathcal{NP}$-complete problems, such as 3-SAT and Subset-Sum.

Throughout we refer to instances that are subjected to random deletions as *percolated instances*. Our main question is whether such percolated instances remain hard to solve by polynomial-time algorithms, under reasonable assumptions from complexity theory.

A first example.   Consider the 3-Coloring Problem, where given a graph G we need to decide whether G is 3-colorable. Suppose we sample a random subgraph G' of G, by deleting each edge of G independently with probability $1/2$, and ask whether the resulting graph is 3-colorable (one can prove similar results when edges are deleted with probability smaller than $1/2$, but we focus on the case where $p = 1/2$ for concreteness). Is there a polynomial time algorithm can decide with high probability whether G' is 3-colorable? Or does the problem remain hard in the sense that an efficient algorithm that determines whether G' is 3-colorable would imply that every problem in $\mathcal{NP}$ admits an efficient algorithm?

We demonstrate that a polynomial-time algorithm that decides whether G' is 3-colorable is unlikely. We show it by considering the following polynomial time reduction from the 3-Coloring Problem to itself. Given an n-vertex graph H the reduction outputs a graph G that is an R-blow-up of H for $R = C\sqrt{\log(n)}$ where $C > 0$ is large enough. That is, each vertex of H is replaced by a cloud of R vertices that form an independent set in G, and each edge of H is replaces with a complete $R \times R$ bipartite graph in G between the corresponding clouds in G. It is clear that H is 3-colorable if and only if G is 3-colorable.

In fact, the foregoing reduction satisfies a stronger robustness property for the random subgraph G' of G. Namely, if H is 3-colorable, then G is 3-colorable, and hence G' is also 3-colorable with probability 1. On the other hand, if H is not 3-colorable, then G is not 3-colorable, and with high probability G' is not 3-colorable either. Indeed, for any edge (v_1, v_2) in H let U_1, U_2 be two clouds in G corresponding to v_1 and v_2. Fixing two arbitrary sets $U_1' \subseteq U_1$ and $U_2' \subseteq U_2$ each of size at least $R/3$, the probability there is no edge connecting a vertex from U_1 to a vertex in U_2 is at most $2^{-R^2/9} = 2^{-C' \log n}$. By union bound we get that with high probability (over the sampling of G')

for any two clouds U_1, U_2 corresponding to an edge in H and any $U_1' \subseteq U_1$ and $U_2' \subseteq U_2$ each of size at least $R/3$ there is at least one edge between U_1' and U_2'. Therefore, with high probability any 3-coloring of G' can be decoded to a 3-coloring of H by coloring each vertex v of H with the color that appears the largest number of times in the coloring of the corresponding cloud in G' (breaking ties arbitrarily). This suggests that unless $\mathcal{NP} \subseteq \text{co}\mathcal{RP}$ there is no polynomial time algorithm that given a 3-colorable graph G finds a legal 3-coloring of a random subgraph of G obtained by subsampling every edge with probability $1/2$.

1.1 Motivation

There is a large body of research dealing with computational problems on random graphs and formulas [19]. This study has resulted with several algorithms which have proven effective on random instances. A more recent line of research suggests that efficient algorithms for finding exact or approximate solutions to computational problems on random objects may not exist [1, 11, 31]. Other works have demonstrated that assuming problems on randomly generated instances to be hard, implies hardness of approximation results for certain optimization problems that are not known to follow from worst-case assumptions [15]. These results raise the question of what kind of hardness results for solving optimization problems exactly or approximately for percolated instances can be derived when the original instance is selected in a worst-case fashion. We note that proving hardness results for our model should be an easier task than proving hardness results for random instances such as those arising, for example, from the Erdös-Rényi random graph $G(n, p)$, as we have more freedom in choosing the instance that is subjected to random deletions.

The study of random discrete structures has resulted with a wide range of mathematical tools which have proven instrumental in proving rigorous results regarding such structures [9, 19, 21, 29]. Our hybrid model may offer the opportunity to apply these methods to a broader range of distributions of instances of $\mathcal{NP}$-hard problems.

1.2 Our results

We consider several classical $\mathcal{NP}$-hard problems, for which we prove that they remain hard also on percolated instance. Unless stated otherwise, n stands for the number of vertices in the graph.

- For the Maximum Independent Set problem we use the hardness of approximation result of [16] to show that for edge percolation, where we keep each edge of a given graph with probability $p > \frac{1}{n^{1-\varepsilon}}$ for some $\varepsilon \in (0, 1)$ it is hard to approximate the maximal independent set on percolated instances within any factor better than $\Omega(\frac{1}{pn^{1-\epsilon}})$ We also show that the chromatic number of a percolated instance in hard to approximate within $O(pn^{1-\varepsilon})$. Note that for $p > \frac{1}{n^{1-\varepsilon}}$ (in fact, for $p > \frac{C \log(n)}{n}$) such random percolated graphs have maximal degree at most $O(pn)$ with high probability, and hence can be colored efficiently using $O(pn)$ colors.

  We also prove that for vertex deletion these problems remain as hard to approximate as in the worst-case, as long as the vertices remain in the graph independently with probability $p > \frac{1}{n^{1-\varepsilon}}$ for some $\varepsilon \in (0, 1)$. More specifically, denoting by m the number of remaining vertices in the vertex percolated subgraph, it is hard to approximate its chromatic number or independence number within a factor of $m^{1-\delta}$ (resp. $\frac{1}{m^{1-\delta}}$) for arbitrary constant $\delta \in (0, 1)$.

3

- For the Vertex-Cover problem, we prove that for any constant $\delta > 0$ an algorithm that gives $2 - \delta$ approximation for percolated instances implies also a $2 - 2\delta$ approximation algorithm for worst-case instances. Our results hold for both edge and vertex percolation, where the edges or the vertices of a given graph remain with probability $p > \frac{1}{n^{1-\varepsilon}}$ for some $\varepsilon \in (0,1)$. In particular, assuming the Unique Games Conjecture, the results of [25] imply there is no randomized polynomial time algorithm that with high probability gives $2 - \delta$ approximation for the Vertex Cover problem on percolated instances.

- For the Hamiltonicity problem, we prove hardness results for percolated instances with respect to edge percolation on directed graphs. We show that the problem where one needs to determine whether a graph contains a Hamiltonian cycle is also hard for percolated graphs, where each edge of a given graph is kept in the graph with probability $p > \frac{1}{n^{1-\varepsilon}}$) for any $\varepsilon \in (0,1)$.

- We also consider percolation of 3-SAT instances where clauses are deleted at random with probability p. We prove that, unless $\mathcal{NP} \subseteq \text{co}\mathcal{RP}$ for every $\varepsilon, \delta \in (0,1)$ if clauses of a given 3-SAT survive with probability $p > \frac{1}{n^{2-\delta}}$, then $(7/8 + \varepsilon)$-approximation on percolated instances is hard, as it is the case for worst-case instances. This result is nearly tight, as known algorithms for random 3-SAT formulas imply that for sufficiently small $c > 0$ if clauses survive deletions with probability $p > \frac{c}{n^2}$, the resulting formula admits a satisfying assignment which can be found efficiently with high probability (see the related works section for more details).

  More generally, we prove that unless $\mathcal{NP} \subseteq \mathcal{BPP}$ arbitrary k-ary Boolean CSP problems are as hard to approximate on percolated instances as in the worst-case, as long as each clause is percolated with probability $p > \frac{1}{n^{k-1-\varepsilon}}$ for any $\varepsilon \in (0,1)$.

  The key step in the proof is establishing that any hardness of approximation of a k-ary CSP problems can be translated to the same hardness approximation on instances whose number of constraints is $n^{k-\eta}$ for arbitrary small $\eta > 0$. For example, relying on the result of Håstad [22] we show that 3-SAT is $\mathcal{NP}$-hard to approximate with a ratio better than $7/8 + \epsilon$ even on instances that contain at least $n^{3-\eta}$ clauses.

  We also consider variable percolation, where each variable is deleted with probability $p > \frac{1}{n^{1-\varepsilon}}$ for any $\varepsilon \in (0,1)$ (when a variable is removed all clauses containing it are removed as well). Similar ideas as those applied for the clause percolation case imply that such percolated instance are essentially as hard as in the worst case.

- We study percolation on instances of the Subset-Sum problem, where each item of the set is deleted with probability $1 - p$. We show that the problem remains hard as long as $p = \Omega(\frac{1}{n^{1/2-\varepsilon}})$ for some $\varepsilon \in (0, 1/2)$, where n is the number of items in the given instance.

1.3 Our techniques

In proving hardness results for percolated instances we use the concept of *robust reductions* which we explain next. It will be convenient to consider promise problems. Recall, that a promise problem is a generalization of a decision problem, where for the problem L there are two disjoint subsets L_{YES} and L_{NO}, such that an algorithm that solves L must accept all the inputs in L_{YES} and reject all inputs in L_{NO}. If the input does not belong to $L_{YES} \cup L_{NO}$, there is no requirement on the output of the algorithm.

Definition 1.1. *For each $y \in \{0,1\}^*$ let $Perc(y)$ be a distribution on $\{0,1\}^*$, that is samplable in time that is polynomial in $|y|$.*

For two promise problems $A = (A_{YES}, A_{NO})$ and $B = (B_{YES}, B_{NO})$ a polynomial time reduction r from A to B is said to be Perc-robust if

1. *For all $x \in A_{YES}$ it holds that $r(x) \in B_{YES}$, and $\Pr[Perc(r(x)) \in B_{YES}] > 1 - o(1)$.*

2. *For all $x \in A_{NO}$ it holds that $r(x) \in B_{NO}$, and $\Pr[Perc(r(x)) \in B_{NO}] > 1 - o(1)$.*

If in the first item we have $\Pr[Perc(r(x)) \in B_{YES}] = 1$, then we say that r is a Perc-robust co$\mathcal{RP}$-reduction. Similarly, if in the second item we have $\Pr[Perc(r(x)) \in B_{NO}] = 1$, then we say that r is a Perc-robust $\mathcal{RP}$-reduction.

Let us elaborate on how robust reductions apply when graph percolation is concerned. Let A, B be $\mathcal{NP}$ languages over graphs, and given a graph y, let $Perc(y)$ be vertex or edge percolation of y. In such setting a reduction r is said to be $Perc$-robust if it satisfies the standard definition of a reduction, i.e., $x \in A$ if and only if $r(x) \in B$, and *in addition* the containment of $r(x)$ in B_{YES} (or in the B_{NO}) is robust to random deletions that are captured by the distribution $Perc(r(x))$. As a concrete example, consider the language 3-SAT consisting of all satisfiable 3-CNF formulas, and the language HamCycle consisting of graphs containing a Hamiltonian cycle. Consider a reduction from 3-SAT to HamCycle that given a 3-CNF formula ϕ produces a graph G. Let G_p be a random subgraph of G obtained from G by including each edge of G independently with probability p. The reduction is said to be robust with respect to edge percolation if the following two assertions hold: (1) if ϕ is satisfiable, then G contains a Hamiltonian cycle and G_p contains a Hamiltonian cycle with high probability, and (2) if ϕ is not satisfiable, then G is not Hamiltonian, and with high probability G_p is not Hamiltonian either.

We make two remarks regarding the example above. First note that if $G = (V, E)$ does not contain a Hamiltonian cycle, then neither does any graph $G' = (V, E')$ where $E' \subseteq E$. Therefore, if such robust reduction exists, then it is necessarily a robust $\mathcal{RP}$-reduction.

Note also that such reduction must be such that if ϕ is satisfiable, then G contains a Hamiltonian cycle, and furthermore G must contain many Hamiltonian cycles, even if ϕ has only a small (e.g., constant number of satisfying assignments. Indeed, if G contained only K Hamiltonian cycles for some constant K, then, G_p is unlikely to be Hamiltonian, as typically only pn edges (n is the number of vertices of G) of each cycle will remain after percolating the edges. That is, such a reduction *cannot* be a parsimonious reduction in the sense that the reduction preserves the number of $\mathcal{NP}$-witnesses.

The existence of such a reduction implies that the Hamiltonicity problem is in some sense $\mathcal{NP}$-hard on percolated instances. Below we explain this hardness more precisely. We start with the following definition.

Definition 1.2. *Let $L = (L_{YES}, L_{NO})$ be a promise problem, and for each y instance of L, let $Perc(y)$ be a distribution on instances of L that is samplable in time that is polynomial in $|y|$.*

The problem $L = (L_{YES}, L_{NO})$ is said to be $\mathcal{NP}$-hard under a Perc-robust reduction if there exists a Perc-robust reduction from an $\mathcal{NP}$-hard problem to L.

We use the term $Perc$-robust to avoid confusion with other notions of robust reductions that have appeared in the literature. In order to ease readability, we will often write robust reductions instead, always refereing to $perc$-robust reductions as defined above.

Proposition 1.3. *Let $L = (L_{YES}, L_{NO})$ be a promise problem, and for each y instance of L, let $Perc(y)$ be a distribution on instances of L that is samplable in time that is polynomial in $|y|$.*

If L is $\mathcal{NP}$-hard under a Perc-robust reduction, then there is no polynomial time algorithm that when given an input y decides with high probability whether $Perc(y) \in B$, unless $\mathcal{NP} \subseteq \mathcal{BPP}$.

If the foregoing hardness holds under a Perc-robust $\mathcal{RP}$-reduction (co$\mathcal{RP}$-reduction), then the same conclusion holds, unless $\mathcal{NP} \subseteq \mathcal{RP}$ (resp. $\mathcal{NP} \subseteq$ co$\mathcal{RP}$).

For co$\mathcal{RP}$-reduction we have the following search decision version of the foregoing proposition.

Proposition 1.4. *Let $L = (L_{YES}, L_{NO})$ be a promise problem, and for each y instance of L let $Perc(y)$ be a distribution on instances of L that is samplable in time that is polynomial in $|y|$.*

If L is $\mathcal{NP}$-hard under a Perc-robust reduction, then, there is no polynomial time algorithm that when given an input y with high probability finds a witness for the assertion $Perc(y) \in B$, unless $\mathcal{NP} \subseteq$ co$\mathcal{RP}$.

An example of an application of Proposition 1.4, consider the 3-SAT problem. Recall that by a result of Håstad [22] given a satisfiable 3-SAT instance Φ it is $\mathcal{NP}$-hard to find an assignment that satisfies significantly more than 7/8 fraction of the constraints of Φ. A stronger conclusion follows from Theorem 4.3. Namely, given a satisfiable 3-SAT instance Φ it is hard to find an assignment that satisfies significantly more than 7/8 fraction of the constraints in a random subformula of Φ, obtained from Φ be deleting each clause with probability, say, $p = 1/2$ (while, any assignment that satisfies Φ, also satisfies every subformula of Φ).

To construct robust reductions we use two methods. One is to apply hardness of approximation results implied by the PCP Theorem [7, 6]. Intuitively, the gap between YES-case and NO-case in such hardness results, makes it possible to prove that percolated instances remain hard as random deletions will not affect the optimum by much, keeping (with high probability) the distinction between the YES-case and the NO-case.

When known hardness results do not suffice (as it is the case for the Vertex Cover problem), or when hardness of approximation results are unlikely, (as is the case for the Subset-Sum problem which admits a PTAS) we "blowup" the instance in a certain way and prove that this blowup preserves certain combinatorial properties even when edges (or vertices) are deleted with high probability. The most standard blowup technique is to replace, given a graph G, every vertex of G with a large independent set and connect two independent sets that correspond to adjacent vertices of G by a complete bipartite graph. This method has been previously used to prove the $\mathcal{NP}$-hardness of Feedback Arc Set on tournaments [2]. Other variants of blowup are used for the Hamiltonian cycle problem and Subset-Sum.

1.4 Related Works

Randomly subsampling subgraphs by including each edge independently in the sample with probability p has been studied extensively in works concerned with cuts and flows (e.g., [23]). More recently, sampling subgraphs has been used to find independent sets [17] (the main sampling technique used, e.g., the layers model is not independent-it introduces dependencies between sampled vertices). The effect of subsampling variables in mathematical relaxations of constraint satisfaction problems on the value of these relaxations was studied in [8]. Edge-percolated graphs have been also used to construct hard-instance for specific algorithms. For example, [27] proved that the well known greedy coloring algorithm performs poorly on the complete r-partite graph in which every

edge is removed independently with probability $1/2$ and $r = n^\epsilon$ for $\epsilon > 0$. Namely, for this graph G, even if vertices are considered in a random order by the greedy algorithm, with high probability $\Omega(\frac{n}{\log n})$ colors are used to color the percolated graph whereas $\chi(G) \leq n^\epsilon$.

The work of [20] examined the problem of finding a maximum independent set in regular graphs where the weights of the vertices are i.i.d. exponential random variables. In this work the authors prove that for 3-regular graphs, it is the case that for every ϵ the problem admits a $(1 - \epsilon)$ approximation in time $n^k g(\epsilon)$ where k is independent of n and $g(\epsilon)$ depends only on ϵ. They also prove that for large enough (constant) degree Δ, the problem of approximating the expected size of a maximum independent set in such randomly weighted graphs is essentially as hard as solving MIS on graphs with maximal degree Δ. Our hardness results are based on different ideas than those of [20].

The field of stochastic optimization is concerned with solving computational problems where elements of the instance (e.g., weights, the existence of edges) are random variables. Typically, the main focus in this line of works is to design algorithms that make decisions (at least for part of the input) before the random variables have been instantiated, with good *expected* guarantees (e.g., [13, 26]). Our work is exclusively concerned with fully instantiated problems. In addition, we focus on a very specific type of uncertainty, where every random variable is either zero or one. As a result, the sampled objects admit a straightforward combinatorial interpretation (e.g., randomly sampled subgraphs or formulas) that is lacking when considering random variables such as the exponential distribution. In addition, dealing with a restricted family of random variables makes it more challenging to prove hardness results regarding instances with edges or vertices whose weights are distributed as these random variables.

When p is sufficiently small, algorithms for random graphs and random formulas can be proven to find the optimal solution (with high probability) for percolated instance. For example, for graph coloring, it is known that for $p = \frac{1+\varepsilon}{n}$ with some positive constant $\varepsilon > 0$, with high probability $G(n, p)$ is 2-degenerate, and hence can be 3-colored in polynomial time [28]. Since the property of being 2-degenerate is monotone, and as 2-colorability can be decided in polynomial time, it follows that for every n-vertex graph and $p \leq \frac{1+\epsilon}{n}$, one can find in polynomial time with high probability a coloring of the edge percolated graph with the minimum number of colors.

Similar reasoning applies to 3-SAT formulas. It is well known that there exists $c > 0$ such that a random 3-SAT formula in which each possible clause is added independently with probability $p = \frac{c}{n^2}$ can be solved with high probability using the pure literal heuristic [10]. As observed in [10], if this heuristic fails in finding a satisfying assignment for a formula ϕ it will still fail to find a satisfying assignment if clauses are added to ϕ. This implies that for any n-variable 3-SAT formula Φ, if $p \leq \frac{c}{n^2}$, then the clause-percolated formula is satisfiable and furthermore a satisfying assignment can be found in polynomial time using the pure literal heuristic.

1.5 Preliminaries

In this work, we will only consider simple graphs without multiple edges and self loops. When directed graphs are concerned we allow the two directed edges (u, v) and (v, u) to coexist-such a situation is not considered as having multiple edges.

Given a graph $G = (V, E)$ (that may be directed or undirected) and $p \in (0, 1)$, we denote by $G_{p,e} = (V, E')$ the probability space of graphs on the same set of vertices, where each edge $e \in E$ is contained in E' independently with probability p. We will say that $G_{p,e}$ is obtained by edge

percolation. We define $G_{p,v} = (V', E')$ as the probability space of graphs, in which every vertex $v \in V$ is contained in V' independently with probability p, and E' is the subgraph of G induced by the vertices V'. We will sometime say that $G_{p,v}$ is obtained from G by vertex percolation. When dealing the running time on percolated instances we will always measure running time in terms of the size of the percolated instance. For edge percolation, it makes little difference as far as polynomial-time algorithms are concerned, as the percolated and original graphs have the same number of vertices. For vertex percolation, this is not the case, since for tiny values of p the size of the percolated graph will be typically much smaller than the size of original graph. In this work we will be only dealing with the case where $p = \frac{1}{n^{1-\Omega(1)}}$, hence with high probability the size of the percolated and the original graphs are polynomially related as well.

Given a graph property $\mathcal{P}$ and a sequence of probability distributions $(\mu_n)_n$ over n-vertex graphs, we will say that $\mathcal{P}$ holds with high probability if $\lim_{n \to \infty} \Pr_{G \sim \mu_n}[G \in \mathcal{P}] = 1$.

We say that an algorithm approximates a maximization problem within a ratio of $0 < a \leq 1$ (where a to depend on the size of the instance) if it returns a feasible solution that is at least $a \cdot OPT$, where OPT is the value of the optimal solution. Similarly, we say that an algorithm approximates a minimization problem within a ratio of $b \geq 1$, if it returns a feasible solution that is at most $b \cdot OPT$ where OPT is the value of the optimal solution.

We shall rely on the following version of the Chernoff bound (see, e.g., [32]).

Theorem 1.5 (Multiplicative Chernoff bound). *Let $X_1, \ldots, X_n$ be independent 0-1 random variables with $\Pr[X_i = 1] = p$. Then,*

$$\Pr[|\sum_{i=1}^{n} X_i - pn| \geq \varepsilon pn] \leq e^{-C\varepsilon^2 pn},$$

for some absolute constant $C > 0$.

Corollary 1.6. *Let $X_1^{(1)}, \ldots, X_n^{(1)}, \ldots, X_1^{(m)}, \ldots, X_n^{(m)}$ be independent 0-1 random variables with $\Pr[X_i^{(j)} = 1] = p$. Then, for some absolute constant $C > 0$ it holds that*

$$\Pr[\exists j \in [m] : |\sum_{i=1}^{n} X_i^{(j)} - pn| \geq \sqrt{Cpn \log(m)}] \leq m^{-3}.$$

Proof. By the multiplicative Chernoff bound above for each $j \in [m]$ it holds that $\Pr[|\sum_{i=1}^{n} X_i^{(j)} - pn| \geq \sqrt{Cpn \log(m)}] \leq e^{-C \log(m)} < m^{-4}$, where $C > 0$ is some absolute constant. Therefore,

$$\Pr[\exists j \in [m] : |\sum_{i=1}^{n} X_i^{(j)} - pn| \geq \sqrt{Cpn \log(m)}] = 1 - \Pr[\forall j \in [m] : |\sum_{i=1}^{n} X_i^{(j)} - pn| \leq \sqrt{Cpn \log(m)}]$$
$$\leq 1 - (1 - m^{-4})^m$$
$$\leq m^{-3},$$

as required. $\square$

2 Graph Coloring, Independent Set and Percolation

An independent set in a graph $G = (V, E)$ is a set of vertices that spans no edge. The independence number of a graph is the size of an independent set of maximum size. Given a graph G, we denote

the independence number of G by $\alpha(G)$. A legal coloring of a graph G is an assignment of colors to vertices that no two adjacent vertices have the same color. The chromatic number of G, denoted by $\chi(G)$ is the minimum number of colors that allows a legal coloring of G. Clearly, $\chi(G) \cdot \alpha(G) \geq n$.

In this section we demonstrate the hardness of approximating $\alpha(G)$ and $\chi(G)$ in percolated graphs for both edge and vertex percolation. We start with edge percolation.

Edge percolation Here a crucial observation is that a graph without large independent sets cannot contain large sets that span a too small number of edges. We need first the following lemma, due to Turan (see, e.g. [3]).

Lemma 2.1. *Every graph H with l vertices and e edges contains an independent set of size at least $\frac{l^2}{2e+l}$.*

Corollary 2.2. *Let $G = (V, E)$ be an n-vertex graph satisfying $\alpha(G) < k$. Then every set of vertices of size $l \geq k$ spans at least $l(l-k)/2k$ edges.*

Proof. Let H be a subgraph of G induced by l vertices, and suppose that H spans e edges. Then, by Lemma 2.1 we have $\alpha(H) \geq \frac{l^2}{2e+l}$. On the other hand, $\alpha(H) \leq \alpha(G) \leq k$, and hence $\frac{l^2}{2e+l} \leq k$, as required. $\square$

Lemma 2.3. *Let $G = (V, E)$ be an n-vertex graph. Then, with high probability $\alpha(G_{p,e}) \leq O\left(\frac{\alpha(G)}{p} \log(np)\right)$.*

Proof. Let $k = \alpha(G) + 1$. Let $C > 0$ be a large enough constant. By the corollary above, every set of size $l = C \frac{\alpha(G)}{p} \log(np)$, spans at least $\frac{l(l-k)}{2k}$ edges. Hence, by taking union bound over all subsets of size l, the probability there exists a set of size l in G_p that spans no edge is at most

$$\binom{n}{l} \cdot (1-p)^{\frac{l(l-k)}{2k}} < \left(\frac{en}{l}\right)^l \cdot \exp\left(-p \cdot \frac{l(l-k)}{2k}\right) < (np)^{-\Omega(l)},$$

where the last inequality uses the choices of l and k, implying that $\left(\frac{en}{l}\right)^l < (np)^l$ and $\exp(-p\frac{l(l-k)}{2k}) < \exp(-\Omega(l \cdot \log(np))) = (np)^{-\Omega(l)}$. $\square$

We observe that in general, the upper bound above cannot be improved, as it is well known that the independence number of $G(n, p)$ is $O\left(\frac{\log(np)}{p}\right)$ with high probability (see, e.g., [9]).

We are now ready to prove that it is hard to approximate the independence number and the chromatic number on edge percolated graphs. For this we consider the following gap problem which we call Gap-Coloring(χ, α), where the YES-instances are all graphs G with $\chi(G) \leq \chi$ and the NO-instances are all graphs G with $\alpha(G) \leq \alpha$. (We assume that for n-vertex graphs G the parameters of the problem are such that $\alpha(G) \cdot \chi(G) < n$, so that the YES-instances and the NO-instances are disjoint sets.)

Theorem 2.4. *Let $\varepsilon \in (0, 1)$ be a fixed constant. Let $p > \frac{1}{n^{1-2\varepsilon}}$, and let $\chi = n^\varepsilon$ and $\alpha = \frac{n^\varepsilon}{p}$, where n denotes the number of vertices in a graph. Then, the Gap-Coloring(χ, α) problem is $\mathcal{NP}$-hard under a robust reduction with respect to edge percolation with parameter p.*

In particular, unless $\mathcal{NP} \subseteq \mathcal{BPP}$ there is no polynomial time algorithm that approximates either $\alpha(G_{p,e})$ or $\chi(G_{p,e})$ within a factor $\frac{1}{pn^{1-2\epsilon}}$ (resp. $pn^{1-2\epsilon}$).

Proof. By a result of Feige and Kilian [16], for any $\varepsilon > 0$ it is $\mathcal{NP}$-hard to decide whether a given n-vertex graph $G = (V, E)$ satisfies $\chi(G) \leq n^{\varepsilon}$ or $\alpha(G) \leq n^{\varepsilon/2}$. Let $\tilde{G} = G_{p,e}$ be the p-edge percolated subgraph of G. Next we claim the following.

YES-case: If $\chi(G) \leq n^{\varepsilon}$, then $\chi(\tilde{G}) < n^{\varepsilon}$.

NO-case: If $\alpha(G) \leq n^{\varepsilon/2}$, then $\alpha(\tilde{G}) < \frac{n^{\varepsilon}}{p}$.

The YES-case is clear, since $\tilde{G}$ is obtained from G by removing edges which can only decrease the chromatic number.

For the NO-case suppose that $\alpha(G) \leq n^{\varepsilon/2}$. Then, by Corollary 2.3 it follows that with high probability $\alpha(G_{p,e}) \leq O\left(\frac{\alpha(G)}{p} \log(np)\right) < \frac{n^{\varepsilon}}{p}$, as required.

The "in particular" part of the theorem follows from the fact that an n-vertex graph G it holds that $\chi(G) \cdot \alpha(G) \geq n$. $\square$

Remark. *Note that for constant $p > 0$ (e.g., $p = 1/2$) this theorem establishes an inapproximability for the independence number of $G_{p,e}$, that matches the inapproximability for the worst case.*

Remark. *Note also that for $p > \frac{1}{n^{1-\varepsilon}}$ (in fact, for $p > \frac{C \log(n)}{n}$) such random percolated graphs have maximal degree at most $O(pn)$ with high probability. Therefore, such graphs $\tilde{G}$ can be colored efficiently using $O(pn)$ colors. In particular, with high probability $\tilde{G}$ contains an independent set of size $\Omega(1/p)$ and hence, $\alpha(\tilde{G})$ can be approximated within a factor of $1/pn$ on p-percolated instances.*

Vertex percolation We now move on to deal with vertex percolation. We show that approximating the $\alpha(G)$ and $\chi(G)$ on percolated instances is essentially as hard as worst-case instances, even if vertices remain with probability $\frac{1}{n^{1-\varepsilon}}$, where n is the number of vertices in the graph for any $\varepsilon \in (0, 1)$. We do it again by proving hardness of the gap problem Gap-Coloring for percolated instances.

Note that in the case of vertex percolation, we (in)approximablity guarantee should depend on the number of vertices in the percolated graph $G_{p,v}$, and not in the original graph.

Theorem 2.5. *Let $\varepsilon, \delta \in (0, 1)$ be fixed constants. Then, for any $p > \frac{1}{n^{1-\delta}}$ the Gap-Coloring(χ, α) problem is $\mathcal{NP}$-hard under a robust reduction with respect to vertex percolation with parameter p, where $\chi = m^{\varepsilon/2}$ and $\alpha = m^{\varepsilon/2}$, with m denoting the number of vertices in the vertex percolated graph.*

In particular, unless $\mathcal{NP} \subseteq \mathcal{BPP}$ there is no polynomial time algorithm that approximates either $\alpha(G_{p,v})$ or $\chi(G_{p,v})$ within a factor $m^{1-\varepsilon}$ for constant any $\varepsilon > 0$.

Proof. For a given $p > \frac{1}{n^{1-\delta}}$ let $c = \frac{\log(pn)}{\log(n)} \in (\delta, 1)$ be such that $p = \frac{1}{n^{1-c}}$, and let $\eta = \varepsilon \cdot c/3$. By a result of Feige and Kilian [16], it is $\mathcal{NP}$-hard to decide whether a given n-vertex graph $G = (V, E)$ satisfies $\chi(G) \leq n^{\eta}$ or $\alpha(G) \leq n^{\eta}$.

Let $\tilde{G} = G_{p,v}$ be the p-vertex percolated subgraph of G, and let m be the number of vertices in $\tilde{G}$. By concentration bounds, we have $|m - pn| < 0.1pn$ with high probability, and we shall assume from now on that this is indeed the case. By the choice of the parameters this implies $n^{\eta} < m^{\varepsilon/2}$ Therefore, if $\chi(G) \leq n^{\eta}$, then $\chi(\tilde{G}) \leq n^{\eta} < m^{\varepsilon}$. On the other hand, if $\alpha(G) \leq n^{\eta}$, then $\alpha(\tilde{G}) < n^{\eta} < m^{\varepsilon}$, and the proof follows. $\square$

2.1 Vertex Cover

An vertex cover in a graph $G = (V, E)$ is a set of vertices $S \subseteq V$ such that every edge $e \in E$ is incident to at least one vertex in S. that spans no edge. In the Minimum Vertex Cover problem we are given a graph G and our goal is to find a vertex cover of G of minimal size.

Note that for an n-vertex graph G it holds that G contains a vertex cover of size k if and only if it contains an independent set of size $n - k$.

There is a simple factor 2 approximation algorithm for the Minimum Vertex Cover problem [32]. On the hardness side, the problem is $\mathcal{NP}$-hard to approximate within a factor of 1.3606 [14], and assuming the Unique Games Conjecture is known to be $\mathcal{NP}$-hard to approximate within a factor of $(2 - \varepsilon)$ for any constant $\varepsilon > 0$ [25]. We prove that the same hardness result hold also when instead of worst-case instances one considers

We will need the following definition.

Definition 2.6. *Given a graph G, the R-blowup of G is a graph $G' = (V', E')$, where every vertex v is replaced by an independent set $\widetilde{v}$ of size R, which we also call the cloud corresponding to v. If $(u, v) \in E$, then $\widetilde{u}$ and $\widetilde{v}$ are connected by a complete $R \times R$ bipartite graph.*

Edge percolation We have the following simple lemma regarding independent sets in edge percolated subgraph of $K_{R,R}$.

Lemma 2.7. *Consider the complete bipartite graph $K_{R,R}$ with bipartition A, B, and let $G_{p,e}$ be the edge percolation of $K_{R,R}$ with probability p. Then, the probability that there is an independent set I in $G_{p,e}$ such that $|I \cap A| = |I \cap B| = C \log(R)/p$ is at most R^{-3}, where C is a large enough constant independent of n or p.*

Proof. For fixed sets $S_A \subseteq A$ and $S_B \subseteq B$ each of size $C \log(R)/p$ the probability that S_A and S_B span no edge is $(1 - p)^{(C \log(R)/p)^2}$. Therefore, by union bound over all S_A and S_B the probability that there is is an independent set I in $G_{p,e}$ with $|I \cap A| = |I \cap B| = C \log(R)/p$ is at most

$$\binom{R}{C \log(R)/p}^2 (1-p)^{(C \log(R)/p)^2} \le m^{2C \log(R)/p} e^{-p(C \log(R)/p)^2}$$

which is at most R^{-3} for large enough C. $\qquad\square$

Consider the following Gap-Vertex-Cover(c, s) problem where the YES-instances are graphs that have a vertex cover of size cn, and NO-instances are all graphs whose minimal vertex cover is larger than sn, where n is the number of vertices in G. Note that, equivalently, the YES-instances are graphs that contain an independent set of size $\alpha(G) \ge (1 - c)n$, the NO-instances are graphs whose maximal independent set is of size $\alpha(G) \le (1 - s)n$.

We remark that the result of Khot and Regev [25] proves that assuming the Unique Games Conjecture the problem Gap-Vertex-Cover$(1 - \varepsilon, 1/2 + \varepsilon)$ if $\mathcal{NP}$-hard for all constant $\varepsilon > 0$. We use this fact in order to hardness of approximation for this problem on percolated instances.

Theorem 2.8. *Let $\varepsilon, \delta \in (0, 1)$ be fixed constants. Assuming the Unique Games Conjecture, Gap-Vertex-Cover$(1 - \varepsilon, 1/2 + \varepsilon)$ is $\mathcal{NP}$-hard under a robust reduction with respect to edge percolation with parameter p for any $p > \frac{1}{n^{1-\delta}}$, where n denotes the number of vertices in the given graph.*

In particular, assuming the Unique Games Conjecture $(2 - \varepsilon)$-approximation of the Vertex Cover problem is hard on edge percolated instances.

Proof. By [25] assuming the Unique Games Conjecture, for any $\varepsilon > 0$ the problem Gap-Vertex-Cover$(1-\varepsilon, 1/2+\varepsilon)$ is $\mathcal{NP}$-hard. Equivalently, given an N-vertex graph G is is $\mathcal{NP}$-hard to distinguish between the case that $\alpha(G) > (1/2-\varepsilon)N$ and the case that $\alpha(G) < \varepsilon N$. We show a reduction from this problem to itself (with slightly larger parameter ε) that is robust for edge percolation.

Consider the reduction that given a graph G outputs the R-blowup of G, which we denote by H, with $R > n$ to be chosen later. That is the graph H is a graph on $n = NR$ vertices, and it is clear that $\alpha(H) = \alpha(G) \cdot R$. Therefore, this is indeed a reduction from the Gap-Vertex-Cover$(1-\varepsilon, 1/2+\varepsilon)$ to itself. We show below that in fact the reduction is robust for edge percolation. In order to do it we prove that with high probability

$$\alpha(G) \cdot R \leq \alpha(\widetilde{H}) \leq \alpha(G) \cdot R + (C \log(R)/p) \cdot N, \tag{1}$$

where $\widetilde{H} = H_{p,e}$ denotes the edge percolation of H with parameter p. Indeed, the left inequality is clear because $\alpha(\widetilde{H}) \geq \alpha(H) = \alpha(G) \cdot R$, since $\widetilde{H}$ is a subgraph of H.

For the right inequality, by Lemma 2.7 with probability at least $(1 - N^2/R^3)$ the following holds: for every edge (u, v) of G the corresponding clouds $\widetilde{u}$ and $\widetilde{v}$ in H are such that there is no independent set I in $\widetilde{H}$, such that $|I \cap \widetilde{u}| \geq C \log(R)/p$ and $|I \cap \widetilde{v}| \geq C \log(R)/p$. Therefore, if I is an independent set that intersects some clouds on more than $C \log(R)/p$, then the vertices corresponding the these clouds must form an independent set in G. Thus, with probability at least $(1 - N^2/R^3)$ we have $\alpha(\widetilde{H}) \leq \alpha(G) \cdot R + (C \log(R)/p) \cdot N$.

Next we choose the parameter R such that the reduction above is indeed a robust reduction for edge percolation with parameter p. For the parameter p let $c = \frac{\log(pn)}{\log(n)}$, and let $R = N^{2/c}$ (where N is the number of vertices in the original graph).

Now, if $\alpha(G) > (1/2-\varepsilon)N$, then by (1) we have $\alpha(\widetilde{H}) \geq \alpha(G) \cdot R > (1/2-\varepsilon)NR = (1/2-\varepsilon)n$, and hence $\widetilde{H}$ contains a vertex cover of size $(1/2+\varepsilon)n$ On the other hand, we claim that if $\alpha(G) < \varepsilon N$, then with high probability $\alpha(\widetilde{H}) < 2\varepsilon n$. Indeed, by the choice of R we have $p = \frac{1}{n^{1-c}} = \frac{1}{(NR)^{1-c}} > \frac{N}{\alpha(G)} \cdot \frac{C \log(R)}{R}$. Therefore, by the right inequality of (1) we have $\alpha(\widetilde{H}) \leq \alpha(G) \cdot R + (C \log(R)/p) \cdot N \leq 2\alpha(G) \cdot R < 2\varepsilon n$, and hence $\widetilde{H}$ does not have a vertex cover of size $(1-\varepsilon)n$. This completes the proof of Theorem 2.8. $\square$

Vertex percolation We proceed with vertex percolation. Note that when considering vertex percolation, the percolation parameter p depends on the number of vertices in the given (worst-case instance) graph, while the performance of the algorithm is measured with respect to the number of vertices in the percolated graph, which is close to pn with high probability

Theorem 2.9. *Let $\varepsilon, \delta \in (0,1)$ be fixed constants. Assuming the Unique Games Conjecture, Gap-Vertex-Cover$(1-\varepsilon, 1/2+\varepsilon)$ is $\mathcal{NP}$-hard under a robust reduction with respect to vertex percolation with parameter p, for any $p > \frac{1}{n^{1-\delta}}$, where n is the number of vertices in the given graph.*

In particular, assuming the Unique Games Conjecture $(2-\varepsilon)$-approximation of the Vertex Cover problem is hard on vertex percolated instances.

Proof. The reduction is the same as in the proof of Theorem 2.8. For the parameters p and ε let $c = \frac{\log(pn)}{\log(n)}$ so that $p = \frac{1}{n^{1-c}}$, and let $R = (\frac{N}{\varepsilon^2})^{1/c}$. Given a graph G the reduction produces the R-blowup of G, which we denote by H. That is the graph H is a graph on $n = NR$ vertices.

Let $\widetilde{H} = H_{p,e}$ denote the vertex percolation of H with parameter p. By Corollary 1.6, with high probability the number of vertices in $\widetilde{H} = H_{p,e}$, which we denote by m is between $pNR -$

$C\sqrt{pNR\log(NR)}$ and $pNR+C\sqrt{pNR\log(NR)}$, and the number of vertices in every cloud of $\widetilde{H}$ is between $pR - C\sqrt{pR\log N}$ and $pR + C\sqrt{pR\log N}$, for some absolute constant $C > 0$ independent of N or p.

Clearly any independent set I in $\widetilde{H}$ gives rise to an independent set in G by taking all vertices v of G such that I intersects the corresponding cloud $\widetilde{v}$. This implies that with high probability it holds (for N large enough) that

$$\alpha(G) \cdot (pR - C\sqrt{pR\log(N)}) \leq \alpha(\widetilde{H}) \leq \alpha(G) \cdot (pR + C\sqrt{pR\log(N)}).$$

By the choice of R we have $R > \frac{C^2 \lg(N)}{\varepsilon^2 p}$, and hence $|\alpha(\widetilde{H}) - \alpha(G)pR| \leq \varepsilon \cdot \alpha(G)pR$. Therefore, denoting by m the number of vertices in $\widetilde{H}$ if $\alpha(G) > (1/2 - \varepsilon)N$, then $\alpha(\widetilde{H}) \geq (1/2 - 3\varepsilon)m$, and hence $\widetilde{H}$ contains a vertex cover of size $(1/2 + 3\varepsilon)m$. On the other hand, if $\alpha(G) < \varepsilon N$, then with high probability $\alpha(\widetilde{H}) < 3\varepsilon m$, and and hence $\widetilde{H}$ does not have a vertex cover of size $(1 - 3\varepsilon)m$. $\square$

3 Hamiltonicity and Percolation

Recall that an Hamiltonian cycle in a graph is a cycle that visits every vertex exactly once. Deciding if a graph (whether directed or undirected) contains a Hamiltonian cycle is a classical $\mathcal{NP}$-hard problem, which we denote by HamCycle. A hamiltonian path, is a simple path that traverses all vertices in the graph.

In this section we prove unless $\mathcal{NP} = \mathrm{co}\mathcal{RP}$, there is no polynomial time algorithm that given a n-vertex directed graph G decides with high probability whether $G_{p,e}$ contains a Hamiltonian cycle for any $p > \frac{1}{n^{1-\epsilon}}$ where $\epsilon \in (0, 1)$.

A natural approach in proving that deciding the Hamiltonicity of percolated instances is hard, is to "blow up" edges. Namely to replace each edge (u, v) by a clique of size k and connect both endpoints of the edges to all vertices of the clique. The idea is that when k is large enough, there is a Hamiltonian path with high probability between all pairs of distinct vertices of the clique. Hence with high probability, we can connect u and v after percolating the edges, by a path that traverses all the vertices of the percolated clique. The problem with this idea, is that the resulting graph after this blowup operation may not be Hamiltonian as there is a new set of vertices for every "edge" in the original graph that needs to be traversed by an Hamiltonian cycle. For *directed* graphs, we overcome this problem by adding to each vertex v a large clique C, adding a directed edge (v, c) for every $c \in C$ and and adding a directed edge (c, u) for every $c \in C$ and $u \in N(v)$ (where $N(v)$ is the set of all vertices having a directed edge from v).

Theorem 3.1. *Let $\varepsilon \in (0, 1)$ be a fixed constant. Then, unless $\mathcal{NP} = \mathcal{RP}$, there is no polynomial time algorithm that when given a directed graph $G = (V, E)$ with n vertices decides with high probability whether $G_{p,e}$ contains a Hamiltonian cycle for any $p > \frac{1}{n^{1-\varepsilon}}$.*

We will need the following claim.

Claim 3.2. *Let $H = (V, E)$ be the directed graph with a source s a sink t, and R vertices $U = \{u_1, \ldots, u_k\}$. The edges of H are*

$$E = \{(s \to u_i) : i \in [R]\} \cup \{(u_i \to t) : i \in [R]\} \cup \{(u_i \to u_j) : i, j \in [R]\}.$$

Let $H' = (V, E')$ be an edge percolation of H, where we keep each directed edge with probability $p = \frac{3\log^5(R)}{R}$. Then, with probability $1 - \frac{1}{R^3}$ there is a Hamiltonian path in H from s to t.

Proof. Let $p_0 \in (0,1)$, and consider the random graph H_{p_0}. Note that with probability at least $1 - 2(1-p_0)^R$ there are two distinct vertices $v_1, v_R \in U$ such that $(s \to v_1), (v_R \to t) \in E'$. Conditioning on these specific $v_1, v_R \in U$, we show that with high probability there is a Hamiltonian path from v_1 to v_R in the subgraph of H_{p_0} induced by U.

By a result of [18, Theorem 1.3] if D is a p_0-edge percolation of the complete directed graph with R vertices with $p_0 = \frac{\log^4(R)}{R}$, then with high probability all edges of D are contained in some Hamiltonian cycle in D. Note that the probability that H_{p_0} contains a Hamiltonian path from v_1 to v_R is equal to the probability that H_{p_0} contains a Hamiltonian cycle that goes through the edge $(v_1 \to v_R)$, conditioned on the event that $(v_1 \to v_R) \in E'$. Therefore, since the distribution of the subgraph of H_{p_0} induced by U is distributed like D, it follows that with high probability the subgraph H_{p_0} induced by U contains a Hamiltonian path from v_1 to v_R, and hence H_{p_0} contains a Hamiltonian path from s to t with probability at least $1/2$.

Next, let $t = 3\log(R)$, and let $p = t \cdot p_0$. We claim that the graph H_p contains an Hamiltonian path from s to t with probability at least $\frac{1}{R^3}$. Observe that if $H'_1, \ldots, H'_t$ are independent copies of H_{p_0}, then the probability that none of the H'_i contains a Hamiltonian path from s to t is at most $(1/2)^t < \frac{1}{R^3}$. Therefore, since each edge of H is contained in $\cup_{i=1}^t H_i$ independently with probability $1 - (1-p_0)^t \leq p$ it follows that H_p contains an Hamiltonian path from s to t with probability at least $\frac{1}{R^3}$, as required. $\qquad\square$

Proof of Theorem 3.1. In order to prove the theorem, we show a reduction that given a directed graph $G = (V, E)$ produces a directed graph $G' = (V', E')$ such that

- If G contains a Hamiltonian cycle, then G' contains a Hamiltonian cycle, and with high probability $G'_{p,e}$ contains a Hamiltonian cycle.

- If G does not contain a Hamiltonian cycle, then neither does G', and hence $G'_{p,e}$ does not contain a Hamiltonian cycle.

The reduction works as follows. Let $V = [N]$ be the vertices of G, and let R be a parameter to be chosen later. The vertices of G' will be $V' = V \bigcup (\cup_{i=1}^N U_i)$, where $U_i = \{u_1^i, \ldots, u_R^i\}$. For each $i \in [N]$ the graph G' contains all edges in both directions inside U_i. For each directed edge $(i \to j) \in E$ we add in G' the directed edges

$$\{(i \to u_\ell^i) : \ell \in [R]\} \cup \{(u_\ell^i \to j) : \ell \in [R]\}.$$

That is, we turn the graph G into G' by adding a clique U_i for each vertex $v_i \in V$, and letting all edges outgoing from v_i go through this clique. This completes the description of the reduction.

Let us first show that that G contains a Hamiltonian cycle if and only if G' contains a Hamiltonian cycle. Indeed, suppose that $C = (\sigma_1, \ldots, \sigma_N)$ is a Hamiltonian cycle in G. Then $C' = (\sigma_1, u_1^{\sigma_1} \ldots, u_R^{\sigma_1}, \ldots, \sigma_N, u_1^{\sigma_N} \ldots, u_R^{\sigma_N})$ is a Hamiltonian cycle in G'. In the other direction, suppose that G' contains a Hamiltonian cycle C'. It is easy to see that any $i \in V$ appearing in C' must be followed immediately by a permutation of all R vertices in U_i. Therefore, by restricting C' to the vertices in V we get a Hamiltonian cycle in G.

Next we show that the reduction above is robust to edge percolation. Let $\tilde{G}' = G'_{p,e}$ be the edge percolation of G'. Clearly if G' does not contain a Hamiltonian cycle, then neither does $\tilde{G}'$. Therefore, it is only left to show that if G' contain a Hamiltonian cycle C, then with high probability $\tilde{G}'$ also contains a Hamiltonian cycle. As explained above a Hamiltonian cycle in G'

is given by a permutation $\sigma = (\sigma_1, \ldots \sigma_N) \in S_N$ and an arbitrary ordering of the vertices in each U_i, i.e., $C' = (\sigma_1, u_1^{\sigma_1} \ldots, u_R^{\sigma_1}, \ldots, \sigma_N, u_1^{\sigma_N} \ldots, u_R^{\sigma_N})$. Note that for each $i \in [N]$ the vertices $\{\sigma_i, u_1^{\sigma_i} \ldots, u_R^{\sigma_i}, \sigma_{i+1}\}$ induce a subgraph isomorphic to the graph H from Claim 3.2. Therefore, by Claim 3.2 if $p > \frac{\log^4(R)}{R}$, then for each $i \in [N]$ with probability $1 - \frac{1}{R^3}$ there is path from σ_i to σ_{i+1} that visits all vertices in U_{σ_i}. By taking union bound over all $i \in [N]$ we get that with probability $1 - \frac{N}{R^3}$ such paths exist for all $i \in [N]$, and by concatenating them we conclude that $\tilde{G}'$ contains a Hamiltonian cycle with high probability.

Finally, we specify the choice of the parameter R. The obtained graph H has $n = NR$ vertices, and the constraints we have are $p > \frac{\log^4 R}{R}$ and $R^3 \gg N$. Therefore, in order to prove the theorem for $p > \frac{1}{n^{1-\varepsilon}}$ with $\varepsilon \in (0,1)$ it is enough to take $R = N^{1/c}$, where $c = \frac{\log(pn)}{\log(n)} > \varepsilon$ such that $p = \frac{1}{n^{1-c}}$. $\qquad\square$

4 Constraint Satisfaction Problem and Percolation

In this section we deal with percolation on Constraint Satisfaction Problems (CSP). An instance Φ of Boolean k-CSP is a formula consisting of a collection of clauses $C_1, ..., C_m$ over n Boolean variables $x_1, ..., x_n$, where each clause is associated with some k-ary predicate $f : \{0,1\}^k \to \{0,1\}$ over k variables $x_{i_1}, \ldots, x_{i_k}$. An instance Φ is said to be simple of all clauses in Φ are distinct. Given an assignment $\sigma : \{x_1, ..., x_n\} \to \{0,1\}$ we say that the constraint C on the variables $x_{i_1}, \ldots, x_{i_k}$ is a satisfied by σ if $f_C(\sigma(x_{i_1}, ..., \sigma(x_{i_k})) = 1$, where f_C is the predicate corresponding to C. Given a formula Φ, and an assignment σ to its variables the value of Φ with respect to the assignment σ, denoted by $val_\sigma(\Phi)$, is fraction of constraints of Φ satisfied by σ. The value of Φ is defined as $val(\Phi) = \max_\sigma val_\sigma(\Phi)$. If $val(\Phi) = 1$ we say that Φ is satisfiable.

We are typically interested in CSP where constraints belong to some fixed family of predicates $\mathcal{F}$. For example, in the k-SAT problem, the constraints are all of the form $f(z_1, \ldots, z_k) = \bigvee_{i=1}^{k}(z_i = b_i)$, for $b_1, \ldots, b_k \in \{0,1\}$. We assume that k, the arity of the constraints, is some fixed constant that does not depend on the number of variables n.

These definitions give rise to the following optimization problem. Given a CSP instance Φ find an assignment that maximizes the value of Φ. We refer to this maximization problem as Max-CSP-$\mathcal{F}$, where $\mathcal{F}$ denotes the family of predicates constraints are taken from. For $0 < s < c \leq 1$, let Gap-CSP-$\mathcal{F}(c, s)$ be the promise problem whose YES-instances are formulas Φ such that $val(\Phi) \geq c$, and NO-instances are formulas Φ such that $val(\Phi) \leq s$. Here we assume the constraints of CSP instances are restricted to be in some family $\mathcal{F}$.

We study two models of percolation on instances of CSP. In *clause percolation* given an instance Φ of CSP its clause percolation is a random formula Φ_p^c over the same set of variables, that is obtained from Φ by keeping each clause of Φ independently with probability p.

In *variable percolation* given an instance Φ of CSP the variable percolation is a random formula Φ_p^v whose set of variables is a subset S of the variables of Φ, where each variable of Φ is in S independently with probability $p \in (0, 1)$ and the clauses of Φ_p^c are all clauses of Φ induced by S. In other words, a clause C of Φ survives if and only if all variables from C the percolation process.

Clause percolation In this section we show that for Constraint Satisfaction Problem with a k-ary constraints, the problem of approximating the optimal value on percolated instances is essentially as hard as approximating it on a worst-case instance as long as $p > \frac{1}{n^{k-1-\delta}}$ for any constant $\delta > 0$.

Theorem 4.1. *Let $\varepsilon, \delta \in (0,1)$ be fixed constants. There is a polynomial time reduction such that given a simple unweighted instance Φ outputs a simple unweighted instance Ψ on N variables with the same constraints, such that $val(\Psi) = val(\Phi)$, and furthermore for any $p > \frac{1}{n^{k-1-\delta}}$ the following holds.*

1. *If $val(\Phi) = 1$, then $val(\Psi_p^c) = 1$ with probability 1.*

2. *If $val(\Phi) < 1$, then with high probability $|val(\Psi_p^c) - val(\Phi)| < \varepsilon$.*

Theorem 4.1 immediately implies the following corollary.

Corollary 4.2. *Let $\mathcal{F}$ be a collection of Boolean constraints of arity k, and suppose that for some $0 < s < c \le 1$ the problem Gap-CSP-$\mathcal{F}(c,s)$ is $\mathcal{NP}$-hard. Then Gap-CSP-$\mathcal{F}(c-\varepsilon, s+\varepsilon)$ is $\mathcal{NP}$-hard under a robust reduction with respect to clause percolation with any parameter $p > \frac{1}{n^{k-1-\delta}}$, where n denotes the number of variables in a given formula, and $\varepsilon, \delta > 0$ are arbitrary constants.*

As a particular application, we have the following result regarding the hardness of approximating 3-SAT on clause-percolated instances.

Theorem 4.3. *Let $\varepsilon, \delta \in (0,1)$ be fixed constants. Then, unless $\mathcal{NP} \subseteq \text{co}\mathcal{RP}$, there is no polynomial time algorithm that when given a satisfiable instance Φ of 3-SAT finds an assignment σ to Φ_p^c such that $val_\sigma(\Phi_p^c) > 7/8 + \varepsilon$ for all $p > \frac{1}{n^{2-\delta}}$.*

Proof. By the result of Håstad [22] for any constant $\varepsilon > 0$ given a weighted 3-SAT instance ϕ it is $\mathcal{NP}$-hard to distinguish between the case that that ϕ is satisfiable, and the case that $val(\phi) < 7/8 + \varepsilon$. Combining this result with the result of [12] we get that the same problem is $\mathcal{NP}$-hard also for unweighted simple instances. The proof follows by applying Theorem 4.1. $\quad\square$

We now return to the proof of Theorem 4.1.

Proof of Theorem 4.1. We start with the following lemma.

Lemma 4.4. *Let Φ be a simple unweighted a k-CSP instance with n variables and m clauses, and let $p > \frac{Cn}{\varepsilon^2 m}$ for some $\varepsilon \in (0,1)$ and some absolute constant $C > 0$. Then,*

1. *If $val(\Phi) = 1$, then $val(\Phi_p^c) = 1$.*

2. *$val(\Phi) < 1$, with high probability $|val(\Phi_p^c) - val(\Phi)| < \varepsilon$.*

Proof. The first item is clear, as any assignment that satisfies Φ will also satisfy Φ_p^c. For the second item, denote by m' the number of clauses in Φ_p^c. By concentration bounds we have

$$\Pr[|m' - pm| > \varepsilon pm] < e^{-\Omega(\varepsilon^2 pm)} < e^{-\Omega(Cn)},$$

where $\Omega(\cdot)$ hides some absolute constant. Fix an assignment σ to the variables of Φ, and let $s = val_\sigma(\Phi)$. Then, the number of clauses in Φ satisfied by σ is sm. Denote by S_σ the number clauses in Φ_p^c satisfied by σ. Since we pick each clause with probability p independently, we have

$$\Pr[|S_\sigma - spm| > \varepsilon pm] < e^{-\Omega(\varepsilon^2 pm)} < e^{-\Omega(Cn)},$$

and hence

$$
\begin{aligned}
\Pr[|val_\sigma(\Phi_p^c) - s| > \varepsilon] &= \Pr[|S_\sigma - sm'| > \varepsilon m'] \\
&\leq \Pr[|m' - pm| > \varepsilon pm/2] + \Pr[|S_\sigma - spm| > \varepsilon pm/2] \\
&\leq 2e^{-\Omega(Cn)},
\end{aligned}
$$

where $\Omega(\cdot)$ hides some absolute constant.

Suppose now that that $val(\Phi) = s$. If σ is an optimal assignment to Φ, i.e., $val_\sigma(\Phi) = s$, then we immediately have by the argument above that $val_\sigma(\Phi_p^c) > s - \varepsilon$ with high probability. On the other hand, for any assignment σ' it holds that $\Pr[val_{\sigma'}(\Phi_p^c) > s + \varepsilon] < e^{-\Omega(Cn)}$ for some sufficiently large $C > 0$, and by taking union bound over all assignments σ we get

$$
\Pr[val(\Phi_p^c) > s + \varepsilon] < \Pr[\exists \sigma' \text{ such that } val_{\sigma'}(\Phi_p^c) > s + \varepsilon] < c^n
$$

for some absolute constant $c < 1$. $\qquad\square$

We note that we assume in the proof above that s is a constant independent of n. This assumption is justified as $s \geq \frac{1}{2^k}$, and we assume that k is independent of n.

Next, we show a polynomial time reduction such that given a Max-CSP-$\mathcal{F}$ instance Φ outputs a Max-CSP-$\mathcal{F}$ instance Ψ with N variables and $N^{k-\varepsilon}$ clauses such that $val(\Psi) = val(\Phi)$. We use similar ideas to those used in [12] in proving that unweighted instances of CSP problems are as hard to approximate as in the weighted case.

Lemma 4.5. *For any $\delta \in (0,1)$ there is a polynomial time reduction such that given a simple unweighted Max-CSP-$\mathcal{F}$ instance Φ outputs a simple Max-CSP-$\mathcal{F}$ instance Ψ with the same constraint with n variables and at least $n^{k-\delta}$ clauses such that $val(\Psi) = val(\Phi)$.*

Proof. The reduction works as follows. Let R be a parameter to be chosen later. Given an instance Φ of k-CSP with M clauses over the variables $x_1, \ldots, x_N$ the reduction creates the following instance Ψ. For each variable x_i of Φ, the instance Ψ will have a set of R corresponding variables $X_i = \{x_{i,j} : j \in [R]\}$, where we think of each variable in X_i as a copy of x_i. For each clause C of Φ we add to Ψ a *cloud* of R^k corresponding clauses, by taking all possible combinations of the variables from the corresponding X_i's. We call the set of R^k clauses corresponding to C the *cloud* of C. That is, Ψ has $n = NR$ variables and $m = M \cdot R^k$ clauses. Therefore, if $R > N^{k/\delta}$, then $m > n^{k-\delta}$.

Next we claim that $val(\Phi) = val(\Psi)$. Clearly, we have $val(\Phi) \leq val(\Psi)$, as any assignment $\sigma : \{x_1, \ldots, x_N\} \in \{0,1\}$ to Φ can be extended to the assignment τ to Ψ by letting $\tau(x_{i,j}) = \sigma(x_i)$ for all $i \in [N], j \in [R]$.

In the other direction, let τ be an assignment to the variables of Ψ.[1] For each $i \in [N]$ let $p_i^1 = \frac{|\{j \in [R] : \tau(x_{i,j}) = 1\}|}{R}$ be the fraction of $x_{i,j}$'s that are assigned the value 1, and let $p_i^0 = 1 - p_i^1$. be the fraction of $x_{i,j}$'s that are assigned the value 0. Construct a assignment σ to the variables of Φ randomly, by setting $\sigma(x_i) = 1$ with probability p_i^1, and setting $\sigma(x_i) = 0$ with probability p_i^0 independently. Equivalently we choose one of the R copies of x_i in Ψ uniformly at random and assign to x_i the value assigned by τ to the variable chosen. A moment of thought reveals that for each clause C of Φ, the probability that σ satisfies C is equal to the fraction of the clauses in Ψ in the cloud corresponding to C that are satisfied by τ. Denote by $SAT_\sigma(C_i)$ the number of clauses

---

[1] Note that if for each $i \in [N]$ the assignment τ gave the same value to all variables in X_i, this would naturally induce a corresponding assignment to Φ. However, this need not be the case in general.

that are satisfied by σ in the cloud corresponding to C_i. Since each clause of Φ corresponds to the same number of clauses in Ψ, it follows that the expected value of Φ under the assignment σ is

$$
\begin{aligned}
\mathbf{E}[val_\sigma(\Phi)] &= \frac{1}{M} \sum_{i=1}^{M} \Pr[\sigma \text{ satisfies } C_i] \\
&= \frac{1}{M} \sum_{i=1}^{M} \frac{SAT_\sigma(C_i)}{R^3} \\
&= val_\tau(\Psi).
\end{aligned}
$$

Hence, there exists an assignment σ to the variables of Φ such that $val_\sigma(\Phi) \geq val_\tau(\Psi)$, and thus $val(\Phi) \geq val(\Psi)$, as required. $\qquad\square$

Theorem 4.1 follows immediately from Lemmas 4.4 and 4.5.

$\square$

We observe that it is unlikely that Lemma 4.5 could be generalized to Max-CSP-$\mathcal{F}$ instances with-arity k and $\Omega(n^k)$ constraints. For example, the value of a 3-SAT formula with $\Omega(n^3)$ clauses, admits $1 - \delta$ approximation for every $\delta \in (0, 1)$ in polynomial time [5].

Variable percolation Next we show that Max-CSP-$\mathcal{F}$ is also hard under variable percolation. We prove below that for p that is no too small, with high probability Max-CSP-$\mathcal{F}$ is hard to approximate on percolated instances within the same factor as in the worst-case setting.

Theorem 4.6. *Let $\varepsilon, \delta > 0$ be fixed constants. There is a polynomial time reduction such that given a simple unweighted instance Φ outputs a simple unweighted instance Ψ on n variables with the same constraints, such that $val(\Psi) = val(\Phi)$, and furthermore for any $p > \frac{1}{n^{1-\delta}}$ the following holds.*

1. *If $val(\Phi) = 1$, then $val(\Psi_p^v) = 1$ with probability 1.*

2. *If $val(\Phi) < 1$, then with high probability $|val(\Psi_p^v) - val(\Phi)| < \varepsilon$.*

The following corollary is the analogue of Corollary 4.2 for variable percolation.

Corollary 4.7. *Let $\mathcal{F}$ be a collection of Boolean constraints of arity k, and suppose that for some $0 < s < c \leq 1$ the problem Gap-CSP-$\mathcal{F}(c, s)$ is $\mathcal{NP}$-hard. Then Gap-CSP-$\mathcal{F}(c-\varepsilon, s+\varepsilon)$ is $\mathcal{NP}$-hard under a robust reduction with respect to vertex percolation with any parameter $p > \frac{1}{n^{1-\delta}}$, where n denotes the number of variables in a given formula, and $\varepsilon, \delta > 0$ are arbitrary constants.*

Proof of Theorem 4.6. The reduction is the same reduction as in the proof of Theorem 4.1. Namely, given a simple unweighted instance Φ with N variables and M clauses the reduction replaces each variable x_i of Φ, with a set of R corresponding variables $X_i = \{x_{i,j} : j \in [R]\}$, and replaces each clause of Φ with a *cloud* of R^k corresponding clauses, by taking all possible combinations of the variables from the corresponding X_i's. That is, the output of the reduction Ψ has $n = NR$ variables and $m = M \cdot R^k$ clauses. We choose $R = N^{1/c}$, where $c = \frac{\log(pn)}{\log(n)} \in (\delta, 1)$ so that $\sqrt{\frac{\log(N)}{pR}} < \frac{1}{N^{c/2}}$.

For each $i \in [N]$ let X_i' be variables from X_i that remain in Ψ_p^v after variable percolation. By concentration, it follows that for $p > \frac{1}{N^{1-\delta}}$ with high probability $||X_i'| - pR| < O(\sqrt{pR \log(n)})$ for all

$i \in [N]$. We assume from now on that this is indeed the case. For a constraint C_i of Φ let $x_{i_1}, \ldots, x_{i_k}$ be the variables that participate in C_i. Then, the number of clauses in the cloud corresponding to C_i in Ψ_p^v is equal to $|X'_{i_1}| \cdots |X'_{i_k}|$, and the total number of clauses in Ψ_p^v is $\sum_{i=1}^{M} |X'_{i_1}| \cdots |X'_{i_k}|$.

By Lemma 4.5 we have $val(\Psi) = val(\Phi)$. In particular, if Φ is satisfiable, then so if Ψ, as any assignment that satisfies Ψ also satisfies any subformula of Ψ, which implies that Ψ_p^v is also satisfiable with probability 1.

Suppose now that $val(\Phi) < 1$. We claim that with high probability $|val(\Psi_p^v) - val(\Phi)| < \varepsilon$.

To prove that $val(\Psi_p^v) \geq val(\Phi) - \varepsilon$, let σ be an optimal assignment to Φ. Extend σ to an assignment τ to Ψ_p^v by letting $\tau(x_{i,j}) = \sigma(x_i)$ for all $1 \leq i \leq R$. Note that for each constraint C_i of Φ if C_i is satisfied by σ, then in Ψ_p^v all clauses in the corresponding cloud are satisfied, and otherwise no clause in the corresponding cloud is satisfied. Denoting by $SAT_\tau(C_i)$ the number of clauses that are satisfied by τ in the cloud corresponding to C_i we have

$$val_\tau(\Psi_p^v) = \frac{\sum_{i=1}^{M} SAT_\tau(C_i)}{\sum_{i=1}^{M} |X'_{i_1}| \cdots |X'_{i_k}|} \geq \frac{val(\Phi)M \cdot (pR - \sqrt{pR\log(N)})^k}{M(pR + \sqrt{pR\log(N)})^k} \geq val(\Phi) - O\left(\sqrt{\frac{\log(N)}{pR}}\right).$$

By the choice of R we get for large enough N

$$val_\tau(\Psi_p^v) \geq val(\Phi) - O\left(\frac{1}{N^{c/2}}\right) \geq val(\Phi) - \varepsilon.$$

Next, we prove that $val(\Phi) \geq val(\Psi_p^v) - \varepsilon$. Given an assignment τ to the variables of Ψ_p^v we decode it into an assignment to Φ using the same decoding as in the proof of Lemma 4.5. Namely, we choose a random assignment σ to the variables of Φ by setting $\sigma(x_i) = 1$ with probability p_i^1 and $\sigma(x_i) = 0$ with probability p_i^0 independently between i's, where $p_i^1 = \frac{|\{x_{i,j} \in X'_i : \tau(x_{i,j}) = 1\}|}{|X'_i|}$, and $p_i^0 = 1 - p_i^1$. Let C'_i be the set of clauses in C_i that belong to Ψ_p^v. Let $SAT_\tau(C'_i)$ the number of clauses that are satisfied by τ in C'_i, it follows that the expected value of Φ under the assignment σ is

$$\mathbf{E}[val_\sigma(\Phi)] = \frac{1}{M} \sum_{i=1}^{M} \Pr[\sigma \text{ satisfies } C'_i] = \frac{1}{M} \sum_{i=1}^{M} \frac{SAT_\tau(C'_i)}{|X'_{i_1}| \cdots |X'_{i_k}|}. \tag{2}$$

On the other hand we have

$$val_\tau(\Psi_p^v) = \frac{\sum_{i=1}^{M} SAT_\tau(C'_i)}{\sum_{i=1}^{M} |X'_{i_1}| \cdots |X'_{i_k}|}. \tag{3}$$

Now, using the assumption that for all $i \in [n]$ it holds that $||X'_i| - pR| < \sqrt{pR\log(n)}$, we get that both (2) and (3) are between $\frac{\sum_{i=1}^{M} SAT_\tau(C'_i)}{M(pR+\sqrt{pR\log(N)})^k}$ and $\frac{\sum_{i=1}^{M} SAT_\tau(C'_i)}{M(pR-\sqrt{pR\log(N)})^k}$. A simple computation reveals that the difference between the two quantities is at most $O\left(\sqrt{\frac{\log N}{pR}}\right)$, and hence

$$\mathbf{E}[val_\sigma(\Phi)] \geq val_\tau(\Psi_p^v) - O\left(\sqrt{\frac{\log(N)}{pR}}\right) \geq val_\tau(\Psi_p^v) - O\left(\frac{1}{N^{c/2}}\right) \geq val_\tau(\Psi_p^v) - \varepsilon.$$

This completes the proof of Theorem 4.6. □

5   The Subset Sum Problem and Percolation

In this section we consider the subset-sum problem, and its percolated version. In the subset-sum problem we are given a set items $\{a_i\}_{i=1}^n$ which are positive integers, and a target integer S. The goal is to decide whether there is a subset of a_i's whose sum is S.

Given an instance $I = (\{a_i\}_{i=1}^n; S)$ of the subset sum problem, we define a percolation on I with probability p to be a random instance I_p, where each item a_i is included in I_p with probability p independently, with the target of I_p being the same as the target of I.

It is known that subset sum is $\mathcal{NP}$-hard. Below we prove hardness of the percolated version of the subset sum problem.

Theorem 5.1. *Rhe Subset-Sum problem is $\mathcal{NP}$-hard under robust reduction with respect to percolation with parameter p for any $p > \frac{1}{n^{1/2-\varepsilon}}$, where n is the number of items in a given instance, and $\varepsilon > 0$ is any fixed constant.*

Proof. In order to prove the theorem, we show a reduction that given an instance $I = (\{a_i\}_{i=1}^N; S)$ of the subset-sum problem with all $a_i > 0$, produces an instance I' on n variables such that the following two properties are satisfied.

- If $I \in$ Subset-Sum, then $I' \in$ Subset-Sum, and furthermore, with high probability $I'_p \in$ Subset-Sum.

- If $I \notin$ Subset-Sum, then $I' \notin$ Subset-Sum, and hence $I'_p \notin$ Subset-Sum with probability 1.

Let us assume that the number of items in I is even. (If N is odd, then, add an item to I that is equal to zero). Let R be a parameter to be chosen later, let $N' = \lceil \log_2(\sum_i a_i) \rceil$, and for $i = 1, \ldots, n$ let $M_i = 2^{C'(N'+i)}$ for a large enough constant C'. For each $i \in [N]$ define the following set

$$J_i = \{M_i + a_i \cdot N^3 + k : k \in \{-R, \ldots, R\}\} \quad \text{and} \quad J'_i = \{M_i + k : k \in \{-R, \ldots, R\}\}$$

Consider now the instance

$$I' = (\cup_{i \in [N]} (J_i \cup J'_i); S'),$$

where $S' = S \cdot N^3 + \sum_{i=1}^N M_i$. Clearly this is a polynomial reduction that outputs a Subset-Sum instance with $n = 2NR$ items.

We show first that $I \in$ Subset-Sum if and only if $I' \in$ Subset-Sum. Indeed, suppose that for some subset $T \subseteq [N]$ it holds that $\sum_{i \in T} a_i = S$. Consider the following subset of items of I'. For each $i \in T$ take the item from J_i that corresponds to $k = 0$, and for $i \in [N] \setminus T$ take the item from J'_i that corresponds to $k = 0$. Then, by taking these items we are getting

$$\sum_{i \in T}(a_i \cdot N^3 + M_i) + \sum_{i \in [N] \setminus T} M_i = S'.$$

In the other direction, suppose that $I' \in \mathbb{N}$. Then, there is some subset $T' \subseteq [N] \times \{0, 1\} \times \{-R, \ldots, R\}$ such that

$$\sum_{(i,t,k) \in T'} (M_i + a_i \cdot N^3 \cdot t + k) = S' = S \cdot N^3 + \sum_{i \in [N]} M_i.$$

Note that by the choices of M_i (namely because M_i's are much larger than a_i's and R) for each $i \in [N]$ there is a unique $t_i \in \{0, 1\}$ and a unique $k_i \in \{-R, \ldots, R\}$ such that $(i, t_i, k_i) \in T'$. Therefore, since $\sum_{i=1}^{N} |k_i| \leq ND < N^3$, it follows that $\sum_{i \in [N]} k_i = 0$, and hence by defining $T = \{i \in [N] : t_i = 1\}$ we get that $\sum_{i \in T} a_i = S$, and so $I \in$ Subset-Sum.

Next, we claim that the reduction above is in fact robust. Indeed, consider the percolated instance I'_p for some $p \in (0, 1]$. Note that if $I \notin$ Subset-Sum, then neither is I', and hence $I'_p \notin$ Subset-Sum with probability 1. It remains to show that if $I \in$ Subset-Sum, then with high probability $I'_p \in$ Subset-Sum. The proof relies on the following claim.

Claim 5.2. *Let $N \in \mathbb{N}$ be even, and let $R \in \mathbb{N}$. Let $A_1, \ldots, A_n \subseteq \{-R, \ldots, R\}$ be random sets chosen by letting each $k \in \{-R, \ldots, R\}$ to be in A_i with probability p independently of each other. Then, with probability $\geq 1 - N/2 \cdot (1 - p^2)^{2R}$ for each $i \in [n]$ there is $k_i \in A_i$ such that $\sum_{i=1}^{N} k_i = 0$.*

Proof. Note that for each odd $i \in [N]$, the probability for a fixed element $x \in \{-R, \ldots, R\}$ that both $x \in A_i$ and $-x \in A_{i+1}$ hold is p^2. Therefore,

$$\Pr[\exists k \in \{-R, \ldots, R\} : k \in A_i \text{ and } -k \in A_{i+1}] = 1 - (1 - p^2)^{2R+1}.$$

Hence, by taking the union bound over all pairs $(i, i+1)$ with odd values of i we get that with probability at least $1 - N/2 \cdot (1 - p^2)^{2R}$, for all odd i's there is $k_i \in A_i$ such that $-k_i \in A_{i+1}$. $\square$

Suppose now that $I \in$ Subset-Sum, i.e., for some subset $T \subseteq [N]$ it holds that $\sum_{i \in T} a_i = S$. Note that the percolated instance I'_p is obtained from I' by taking random subsets of J_i and J'_i independently of each other. For $i \in [N]$ define A_i to be the p-percolated subsets of J_i if $i \in T$, and define A_i to be the p-percolated subsets of J'_i if $i \notin T$. Note that if $R > \frac{C \log(N)}{p^2}$, then the conclusion of Claim 5.2 holds with probability at least $1 - 1/N$. Therefore, in the percolated instance I'_p by taking the items of I'_p from A_i's that correspond to $k \in A_i$'s from Claim 5.2 we get

$$\sum_{i \in T}(M_i + a_i \cdot N^3 + k_i) + \sum_{i \in [N] \setminus T}(M_i + k_i) \; = \; (\sum_{i \in [N]} M_i) + (\sum_{i \in T} a_i \cdot N^3) + (\sum_{i \in [N]} k_i)$$
$$= \; (\sum_{i \in [N]} M_i) + S + 0$$
$$= \; S'.$$

Therefore, with high probability $I'_p \in$ subset-sum as required.

Finally, note that the reduction works as long as $R > C\frac{\log(N)}{p^2}$, or equivalently $p > C\sqrt{\frac{\log N}{R}}$. It is easy to verify that for $R = N^{1/c}$, with $c = \frac{\log(p\sqrt{n})}{\log(n)}$, the foregoing reduction is indeed a robust reduction with respect to percolation with parameter $p > \frac{1}{n^{1/2-\varepsilon}}$ for any constant $\varepsilon > 0$, where n is the number of items in the Subset-Sum instance. $\square$

6 Conclusion

We have examined the complexity of percolated instances of several well known $\mathcal{NP}$-hard problems and established the hardness of solving exactly and approximately these problems on such instances.

It might be of interest to study percolated instances of other $\mathcal{NP}$-hard problems that were not considered here.

There are several question arising from this work. For the HamCycle problem it would be interesting to determine whether vertex-percolated instances are hard (in directed or undirected graphs). Currently, we are unable to establish that this problem is hard even if every vertex remains with probability $p < 1$, where p is a constant that does not depend on the size of the graph. It could be the case that there is no reduction from $\mathcal{NP}$ to HamCycle that is robust to vertex percolation. Proving the inexistence of such reductions (if true) could be of interest.

It might also prove worthwhile to determine whether percolated instances of 3-SAT remain hard to solve even if $p = O(1/n^2)$ over n-variable formulas. Several works suggest that finding a satisfying assignment for a random 3-SAT instance with n variables and $C_1 n$ clauses is hard when C_1 is close to the satisfiability threshold [30]. Other works suggest that certifying the unsatisfiability of a random 3-SAT instance with $C_2 n$ clauses, with C_2 being large enough, is difficult as well [15]. These works may serve as evidence that 3-SAT should be hard to solve for $p = O(1/n^2)$.

One of the first algorithms for solving independent sets in the random graph $G(n, p)$ is the Karp-Sipser algorithm [24]. This algorithm works by choosing iteratively a degree one vertex randomly, adding it to the independent set, and removing both the selected vertex and its sole neighbor from the graph. When there are no vertices of degree one, the algorithm terminates. It was proven in [4] that when $p < \frac{e}{n}$ (where e is the base of the natural logarithm, $e = 2.71828\ldots$) the algorithm finds with high probability an optimal independent set. When $p > \frac{e}{n}$, the Karp-Sipser algorithm fails (with high probability) to find a maximum independent set of the graph [24]. Despite extensive research, no algorithm is known to find an optimal independent set in $G(n, p)$ when $p > \frac{e}{n}$.

It is not clear whether the Karp-Sipser algorithm works on random subgraphs of worst-case graphs, as opposed to a random subgraph of the complete graph. This leads to the following problem.

Problem 6.1. *Let $p < \frac{e}{n}$. Is there a polynomial-time algorithm that given an n-vertex graph G finds a maximal independent set in $G_{p,e}$ with high probability?*

Is it true that for $p > \frac{e}{n}$ no algorithm can find a maximal independent set in $G_{p,e}$ for worst case instance G, unless $\mathcal{NP} \subseteq \mathcal{BPP}$?

Acknowledgements

References

[1] D. Achlioptas and A. Coja-Oghlan. Algorithmic barriers from phase transitions. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS Philadelphia, PA, USA*, pages 793–802, 2008.

[2] N. Ailon, M. Charikar, and A. Newman. Aggregating inconsistent information: Ranking and clustering. *J. ACM*, 55(5), 2008.

[3] N. Alon and J. H. Spencer. *The probabilistic method*. 2000.

[4] J. Aronson, A. M. Frieze, and B. Pittel. Maximum matchings in sparse random graphs: Karp-sipser revisited. *Random Struct. Algorithms*, 12(2):111–177, 1998.

[5] S. Arora, D. R. Karger, and M. Karpinski. Polynomial time approximation schemes for dense instances of np-hard problems. *J. Comput. Syst. Sci.*, 58(1):193–210, 1999.

[6] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998.

[7] S. Arora and S. Safra. Probabilistic Checking of Proofs: A New Characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998.

[8] B. Barak, M. Hardt, T. Holenstein, and Steurer D. Subsampling mathematical relaxations and average-case complexity. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA, San Francisco, California, USA*, pages 512–531, 2011.

[9] B. Bollobás. *Random graphs.* Springer, 1998.

[10] A. Z. Broder, A. M. Frieze, and E. Upfal. On the satisfiability and maximum satisfiability of random 3-cnf formulas. In *Proceedings of the Fourth Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms, Austin, Texas*, pages 322–330, 1993.

[11] A. Coja-Oghlan and C. Efthymiou. On independent sets in random graphs. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA, San Francisco, California, USA*, pages 136–144, 2011.

[12] P. Crescenzi, R. Silvestri, and L. Trevisan. On weighted vs unweighted versions of combinatorial optimization problems. *Inf. Comput.*, 167(1):10–26, 2001.

[13] B. C. Dean, M. X. Goemans, and J. Vondrák. Adaptivity and approximation for stochastic packing problems. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA, Vancouver, British Columbia, Canada*, pages 395–404, 2005.

[14] I. Dinur and S. Safra. On the hardness of approximating minimum vertex cover. *Annals of Mathematics*, 162(1):439–485, 2005.

[15] U. Feige. Relations between average case complexity and approximation complexity. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing Montréal, Québec, Canada*, pages 534–543, 2002.

[16] U. Feige and J. Kilian. Zero knowledge and the chromatic number. *Journal of Computer and System Sciences*, 57(2):187 – 199, 1998.

[17] U. Feige and Daniel Reichman. Recoverable values for independent sets. *Random Struct. Algorithms*, 46(1):142–159, 2015.

[18] A. Ferber, G. Kronenberg, and E. Long. Packing, counting and covering hamilton cycles in random directed graphs. 2015. arXiv:1506.00618.

[19] A. M. Frieze and C. McDiarmid. Algorithmic theory of random graphs. *Random Struct. Algorithms*, 10(1-2):5–42, 1997.

[20] D. Gamarnik, D. Goldberg, and T. Weber. PTAS for maximum weight independent set problem with random weights in bounded degree graphs. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA, Austin, Texas, USA*, pages 268–278, 2010.

[21] G. Grimmett. *Percolation*. Springer, 1999.

[22] J. Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, July 2001.

[23] D. R. Karger. Random sampling in cut, flow, and network design problems. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, Montréal, Québec, Canada*, pages 648–657, 1994.

[24] R. M. Karp and M. Sipser. Maximum matchings in sparse random graphs. In *22nd Annual Symposium on Foundations of Computer Science, Nashville, Tennessee, USA*, pages 364–375, 1981.

[25] S. Khot and O. Regev. Vertex cover might be hard to approximate to within 2-epsilon. *J. Comput. Syst. Sci.*, 74(3):335–349, 2008.

[26] J. M. Kleinberg, Y. Rabani, and É. Tardos. Allocating bandwidth for bursty connections. *SIAM J. Comput.*, 30(1):191–217, 2000.

[27] L. Kucera. The greedy coloring is a bad probabilistic algorithm. *J. Algorithms*, 12(4):674–684, 1991.

[28] T. Luczak. Size and connectivity of the k-core of a random graph. *Discrete Mathematics*, 91(1):61–68, 1991.

[29] M. Mezard and A. Montanari. *Information, physics, and computation*. Oxford University Press, 2009.

[30] D. G. Mitchell, B. Selman, and H. J. Levesque. Hard and easy distributions of SAT problems. In *Proceedings of the 10th National Conference on Artificial Intelligence. San Jose, CA.*, pages 459–465, 1992.

[31] B. Rossman. The monotone complexity of k-clique on random graphs. *SIAM J. Comput.*, 43(1):256–279, 2014.

[32] V. V. Vazirani. *Approximation Algorithms*. Springer-Verlag New York, Inc., New York, NY, USA, 2001.