Stochastic Minimum Vertex Cover in General Graphs: a 3/2-Approximation

Mahsa Derakhshan, Naveen Durvasula, and Nika Haghtalab

University of California, Berkeley

Abstract

We study the stochastic vertex cover problem. In this problem, $G = (V, E)$ is an arbitrary known graph and $\mathcal{G}^\star$ is an unknown random subgraph of G where each edge e is realized independently with probability p. Edges of $\mathcal{G}^\star$ can only be verified using edge queries. The goal in this problem is to find a minimum vertex cover of $\mathcal{G}^\star$ using a small number of queries.

Our main result is designing an algorithm that returns a vertex cover of $\mathcal{G}^\star$ with size at most $(3/2+\varepsilon)$ times the expected size of the minimum vertex cover, using only $O(n/\varepsilon p)$ non-adaptive queries. This improves over the best-known 2-approximation algorithm by Behnezhad, Blum and Derakhshan [SODA'22] who also show that $\Omega(n/p)$ queries are necessary to achieve any constant approximation. By the Unique Games Conjecture, improving over 2-approximation is not possible in polynomial time. To get around this, we assume that we have access to an MVC oracle and design an oracle-efficient algorithm.

Our guarantees also extend to instances where edge realizations are not fully independent. We complement this upperbound with a tight 3/2-approximation lower bound for stochastic graphs whose edges realizations demonstrate mild correlations.

1 Introduction

In the *stochastic vertex cover* problem, we are given an arbitrary base graph $G = (V, E)$ with n vertices but we do not know which edges in E actually exist. Rather each edge $e \in E$ is realized independently with a given existence probability $p_e \in (0, 1]$, forming a subgraph $\mathcal{G}^\star$. Our goal is to find a minimum vertex cover of $\mathcal{G}^\star$. While $\mathcal{G}^\star$ is unknown, one can verify its edge set by querying edges $e \in E$. Of interest, then, are algorithms that *query a small subset of edges* and, based on the outcome of these queries, find a *near-optimal vertex cover* of $\mathcal{G}^\star$. How small should the set of queried edged be? The gold standard in these problems is to non-adaptively issue a number of queries that is linear in the number of vertices and polynomial in inverse probability $p = \min_{e \in E} p_e$.

While these stochastic settings are primarily concerned with information theoretical questions, most positive results have focused on problems whose non-stochastic counterparts admit computationally-efficient algorithms. Instances include, minimum spanning tree [GV04, GV06], all pairs shortest paths [Von07], maximum matching [BDH+15, BDH+20b, AKL16, AKL17, BR18, YM18, BDF+19, AB19, BFHR19, BDH20a, BD20], 2-approximate minimum vertex cover, and bipartite minimum vertex cover [BBD22]. This emphasis on efficiently solvable problems is not accidental. By and large, structurally-simple properties and heuristics that had long played a key role in understanding and designing computationally efficient algorithms have been used to guide an algorithm in its choice of queries, e.g., the Tutte-Berge witness sets [AKL16], short augmenting paths [BDH+15], local computation [BDH20a], and greedy heuristics [BBD22].

On the other hand, for vertex cover beyond a 2-approximation, and other computationally hard regimes, lack of understanding regarding structural properties has also been a barrier towards solving the stochastic variants of the problems. A natural question here is whether it is possible to obtain any positive results for problems that lack these structure? In this work, we consider this question for the minimum vertex cover problem.

Question 1. *Can we achieve a better than 2-approximation for the stochastic minimum vertex cover problem despite the lack of computationally efficient algorithms for this problem? Can we do so efficiently, given access to an oracle for computing the minimum vertex cover of any graph?*

Our paper answers both of these questions in the affirmative. At a high level, we introduce an algorithm that returns a vertex cover of $\mathcal{G}^\star$ with probability 1 whose size is at most $3/2 + \varepsilon$, times the expected minimum vertex cover of $\mathcal{G}^\star$, for any desirably small ε. Moreover, our algorithm can be efficiently implemented using a polynomial number of calls to a minimum vertex cover oracle. The following theorem, which is formally stated in Section 5, presents our main result.

Theorem 1.1 (Upper-bound)**.** *For any $\varepsilon \in (0, 0.1)$, there is an algorithm (namely Algorithm 3) that returns a $(3/2 + \varepsilon)$-approximate solution for the stochastic minimum vertex cover problem using $O(n/\varepsilon p)$ queries. Moreover, this algorithm is oracle-efficient.*

The number of queries used in this algorithm is asymptotically optimal since [BBD22] show that $\Omega(n/p)$ queries are necessary to achieve any constant approximation ratio. Interestingly, the approximation guarantees of Theorem 1.1 continue to hold even if edges of $\mathcal{G}^\star$ are correlated. (See Section 7.) This allows us to handle *mild correlations* in the realization of $\mathcal{G}^\star$. That is, even if $O(n)$ edges are allowed to be realized in a correlated way, we can still achieve a $(3/2 + \varepsilon)$-approximate solution using only $O(n/\varepsilon p)$ queries. Our next result shows that for such mildly correlated processes, a 3/2-approximation is the best one can hope to get when using $O(n/\varepsilon p)$ queries. The following theorem, which is formally stated in Section 7, presents our lower bound.

Theorem 1.2 (Informal Lower Bound)**.** *There is a stochastic process for generating $\mathcal{G}^\star$ with $O(n)$*

correlated edges[1]*, such that any algorithm that returns a vertex cover of* $\mathcal{G}^\star$ *using only* $O(n/\varepsilon p)$ *queries, must have an approximation ratio of at least* $(3/2-\varepsilon)$ *with probability* $1-o(1)$.

Theorems 1.1 and 1.2 together demonstrate that our results are tight, as long as there is mild correlations between the edges of $\mathcal{G}^\star$. This shows that to further go beyond the 3/2-approximation one must fully leverage independence across all edges. We hope that this tight characterization of what is achievable for mildly correlated graphs may guide the design of new approaches that fully leverage independence to achieve a $(1+\varepsilon)$-approximate minimum vertex cover. Indeed, a similar characterization of what is achievable for correlated realization was given for the stochastic matching problem by [AKL16]. This later inspired [BDH20a] to utilize $(1-\varepsilon)$-approximate matching algorithms designed in the LOCAL model of computation [Gha19] to fully benefit from independence and find $(1-\varepsilon)$-approximate stochastic matching. This subsequently resulted in a $(1+\varepsilon)-$approximate stochastic vertex cover for bipartite graphs [BBD22]. The absence of better than 2-approximation algorithms for minimum vertex cover in the LOCAL model may be seen as an obstacle in breaking the 3/2 barrier for this problem.

Algorithm Design Overview. To achieve a 3/2-approximation, we start with two approaches to solving the stochastic minimum vertex cover problem that give 2-approximations. Our final algorithm is the result of carefully combining insights from these two approaches.

All of the algorithms in this work follow the same blueprint: We consider a set $P \subseteq V$ and the induced subgraph $H = G[V \setminus P]$. We then query all the edges of H to realize $\mathcal{H}^\star$ and take its vertex cover M. Our algorithm then returns $S = P \cup M$. We note that S is a vertex cover of $\mathcal{G}^\star$, since any edge of $\mathcal{G}^\star$ that is not covered by P is covered by M. Our algorithms and their guarantees only differ in their choice of P.

We give two 2-approximation algorithms that employ different principles in their choice of P. Algorithm 1 hallucinates a random subgraph of G, namely $\mathcal{G}_1$, and uses P that is a minimum vertex cover of $\mathcal{G}_1$. On the other hand, Algorithm 2 estimates the probability that any vertex $v \in V$ would belong to the minimum vertex cover of $\mathcal{G}^\star$, denoted by c_v, and uses $P = \{v \mid c_v > 1/2\}$. While both of these algorithms achieve a 2-approximation in the worst-case, their performance guarantees differs based on the distribution of c_v's. In particular, both of these algorithms over-include some vertices — i.e., include a vertex that does not belong to the minimum vertex cover — but they differ in the type of vertices they over-include. As our analysis shows, the first algorithm significantly over-includes vertices that have a very small c_v, but the second algorithm only over-includes vertices with $c_v > \frac{1}{2}$.

Our 3/2-approximation algorithm (Algorithm 3) combines these two insights to define the set $P = P_1 \cup P_2$. It first chooses τ that carefully balances the contribution of vertices with $c_v > \tau$ to the expected size of the minimum vertex cover. For vertices whose $c_v \in [1-\tau-\varepsilon, \tau]$, we use the style of Algorithm 1 and only include them in P_1 if they also belong to a minimum vertex cover of a hallucinated random subgraph. For the set of vertices with $c_v > \tau$, we use the style of Algorithm 2 and include all of them in P_2. Our analysis carefully balances out the over-inclusion of vertices to achieve a 3/2-approximation.

[1]For a formal definition of a stochastic process with $O(n)$ correlated edges, see Definition 7.1

2 Notations

We work with a known arbitrary graph $G = (V, E)$ and existence probability $p_e \in (0, 1]$ for each $e \in E$. We consider a random subgraph $\mathcal{G}^\star$ in which every edge $e \in E$ is realized with probability p_e, independently. We denote $p = \min_{e \in E} p_e$.

Let MVC denote a function that given any input graph outputs a minimum vertex cover of that graph. We may also refer to this as the *minimum vertex cover* oracle. We define $\text{OPT} = \text{MVC}(\mathcal{G}^\star)$ to be the optimal solution of our problem. Note that OPT is a random variable since $\mathcal{G}^\star$ itself is a random realization of G. We also let $\textsc{opt} = \mathbb{E}[|\text{OPT}|]$ be the expected size of this optimal solution. Moreover, for any vertex $v \in V$, we define

$$c_v = \Pr[v \in \text{OPT}],$$

which is the probability that v joins the optimal solution. This implies $\sum_{v \in V} c_v = \textsc{opt}$. Similarly, for any edge $e = (u, v)$, we let c_e be the probability that this edge is covered by OPT i.e.,

$$c_{(u,v)} = \Pr[u \in \text{OPT or } v \in \text{OPT}].$$

When c_vs and c_es are not known in advance, we use a polynomial number of calls to a minimum vertex cover oracle to estimate them within arbitrary accuracy. See Section 6 for more details regarding these estimates.

3 Warm Up – Beating 2-Approximation

In this section, we start by discussing two simplified variants of our 3/2-approximate algorithm. Both of these algorithms have the worst case approximation ratio of 2. However, their performance varies for different instances of the problem. One of them has a better performance if a large portion of OPT comes from vertices with smaller c_v's while the other one prefers a large portion of OPT to be from vertices with larger c_v's. After discussing these two algorithms, we will show how, due to their opposing nature, running the best of the two algorithms beats the 2-approximation ratio. Finally, we explore how this observation inspires the design of our 3/2-approximation algorithm.

All the algorithms we design in this paper follow a similar framework. In all of them, we first pick a subset of vertices P and commit to adding them to the final vertex cover. As a result, we only need to query the edges not covered by these vertices. We denote this subgraph by H. Formally, $H = G[V \setminus P]$ is the subgraph induced in G by $V \setminus P$. After querying H, we find a vertex cover of its realized edges which we denote by M. Finally, we output $P \cup M$. Our algorithms mainly differ in their choice of P. See Figure 1 for an illustration of our framework.

We make the following observation about our algorithm framework.

Observation 3.1. *Let $P \subseteq V$ and $H = G[V \setminus P]$ be the induced subgraph on $V \setminus P$. Let $\mathcal{H}^\star$ be the realization of the edges of H and M be a vertex cover of $\mathcal{H}^\star$. Then, $P \cup M$ is a vertex cover of $\mathcal{G}^\star$.*

The simplest way of picking M is for it to be a minimum vertex cover of all the realized edges of H. However, sometimes, for the sake of analysis we require M to be a vertex cover of H satisfying a certain property. In particular, we want $\Pr[v \in M] = c_v$. We achieve this by letting M be a minimum vertex cover of all the realized edges of H and a hallucination of $G \setminus H$. We will explain this in more detail in the following algorithm.

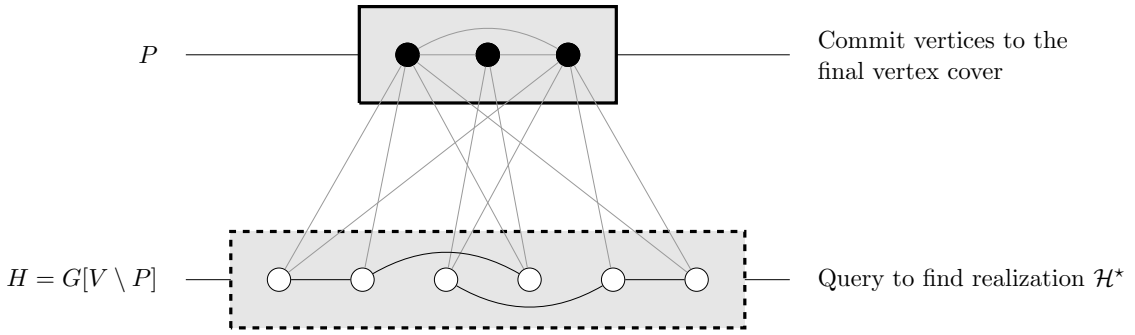

Figure 1: **A commit, then query approach.** After committing vertices of P to the final vertex cover, we query the subgraph of edges not covered by P which we denote by H. This is the subgraph of G induced by $V \setminus P$. The output of our algorithm is $P \cup M$ where M is a vertex cover of $\mathcal{H}^\star$.

The first 2-approximate algorithm. To construct the subset P, this algorithm (formally stated as Algorithm 1) hallucinates a random realization of G and lets P be its minimum vertex cover. Following the aforementioned framework, the next step is to find M: a vertex cover of $\mathcal{H}^\star$. In order to do that, the algorithm hallucinates another realization of G by by including any edge $e \in G \setminus H$ with probability p_e, independently, and including edges of $\mathcal{H}^\star$. Finally, it finds an MVC of this realization denoted by M, and outputs $P \cup M$.

Algorithm 1. A 2-approximation stochastic vertex cover algorithm

1 Let $\mathcal{G}_1$ be a random realization of G containing any edge $e \in G$ independently w.p. p_e.
2 $P \leftarrow \text{MVC}(\mathcal{G}_1)$
3 Let H be the subgraph induced in G by $V \setminus P$.
4 Query subgraph H and let $\mathcal{H}^\star$ be its realization.
5 Let $\mathcal{G}_2$ be a subgraph of G containing all the edges in $\mathcal{H}^\star$ and any edge $e \in G \setminus H$ independently w.p. p_e.
6 $M \leftarrow \text{MVC}(\mathcal{G}_2)$
7 Return $P \cup M$

We will first prove that this algorithm queries only $O(n/p)$ edges, that is $|H| = O(n/p)$. This is due to the fact that, by definition, none of the edges of H are realized in $\mathcal{G}_1$ since otherwise, one of its vertices should be in vertex cover P (which is a contradiction). For any subgraph with more than n/p edges the probability of none of its edges being in $\mathcal{G}_1$ is at most $(1-p)^{n/p}$. Moreover since H is an induced subgraph of G, there are at most 2^n possibilities for it. By an application of union bound, we see that w.h.p., H has at most n/p edges. That is:

$$\Pr[|H| \leq n/p] \geq 1 - 2^n(1-p)^{n/p} \geq 1 - \frac{2^n}{e^n} = 1 - (2/e)^n.$$

We state this observation below for future reference.

Observation 3.2. *Let $\mathcal{G}$ be a random realization of G containing any edge $e \in G$ independently with probability p_e, and let M be a minimum vertex cover of $\mathcal{G}$. The number of edges in G not covered by M is $O(n/p)$.*

As mentioned earlier, it is only for the sake of analysis that we do not simply let M be an arbitrary minimum vertex cover of $\mathcal{H}^\star$. Instead, we use $M = \text{MVC}(\mathcal{G}_2)$ since it satisfies that

$\Pr[v \in M] = c_v$. This is due to the fact that $\mathcal{G}_2$ and $\mathcal{G}^\star$ are drawn from the same distribution, i.e., $\mathcal{G}_2$ contains any edge e independently with probability p_e. Below, we state this observation formally.

Observation 3.3. *Let H be a subgraph of G and $\mathcal{H}^\star$ be its actual realization. Moreover, we define $\mathcal{G}_2$ to be a subgraph of G containing all the edges in $\mathcal{H}^\star$ and any edge $e \in G \setminus H$ independently with probability p_e. If M is a minimum vertex cover of $\mathcal{G}_2$ it satisfies $\Pr[v \in M] = c_v$.*

As a result of this observation and the fact that P also comes from the same distribution as OPT, we get $\mathbb{E}[|P \cup M|] \leq 2\text{OPT}$ which implies that Algorithm 1 is a 2-approximation. However, we claim that depending on the the way c_v's are distributed this algorithm may result in a better than 2 approximation ratio. First of all, observe that $\mathcal{G}_1$ and $\mathcal{G}_2$ are two independent random variables. The only way in which $\mathcal{G}_1$ impacts the construction of $\mathcal{G}_2$ is in determining which of its edges come from the actual realization and which ones are hallucinated. Nonetheless, $\mathcal{G}_2$ contains any edge e with probability p_e independently from other edges and from $\mathcal{G}_1$. This implies that $P = \text{MVC}(\mathcal{G}_1)$ and $M = \text{MVC}(\mathcal{G}_2)$ are also two independent random variables. Therefore, any vertex v joins P and M independently with probability c_v which means

$$\Pr[v \in P \cup M] = 1 - (1 - c_v)^2 = c_v(2 - c_v). \tag{1}$$

Since $c_v(2-c_v) \approx 2c_v$ for $c_v \to 0$, the worst case scenario for this algorithm is when all vertices have very small c_vs. On the other hand, if for all the vertices we have $c_v \geq 0.5$ then, $c_v(2 - c_v) \leq 1.5c_v$ which results in an approximation ratio of 3/2 (our desired bound). Having established this, we will next discuss another 2-approximation algorithm with an opposite nature. This algorithm has a better performance if a large portion of the optimal solution comes from vertices with small c_vs.

The second 2-approximate algorithm. This algorithm (formally stated as Algorithm 2) is not exactly a 2-approximation. Instead, it finds a $(2 + O(\varepsilon))$-approximate vertex cover using $O(n/\varepsilon p)$ queries for any $\varepsilon \in (0, 0.1)$. In this algorithm, we set $P = \{v \in V : c_v \geq 0.5 - \varepsilon\}$. The rest of the algorithm follows our standard framework similar to Algorithm 1.

Algorithm 2. A $(2 + O(\varepsilon))$-approximation stochastic vertex cover algorithm

1. Define $P = \{v \in V : c_v \geq 0.5 - \varepsilon\}$.
2. Let H be the subgraph induced in G by $V \setminus P$.
3. Query subgraph H and let $\mathcal{H}^\star$ be its realization.
4. Let $\mathcal{G}$ be a subgraph of G containing all the edges in $\mathcal{H}^\star$ and any edge $e \in G \setminus H$ independently w.p. p_e.
5. $M \leftarrow \text{MVC}(\mathcal{G})$
6. Return $P \cup M$

We will first discuss why the above mentioned algorithm queries only $O(n/\varepsilon p)$ edges. To put Observation 3.2 differently, the expected number of edges in G not covered by OPT, or equivalently $\sum_e (1 - c_e)$, is $O(n/p)$. Using Markov's inequality, this implies that the number of edges with $c_e < 1-\varepsilon$ is upper-bounded by $O(n/\varepsilon p)$. Since both end-points of any edge $e \in H$ have $c_v < 0.5-\varepsilon$, all these edges have $c_e \leq 1 - 2\varepsilon$. Hence, there are at most $O(n/\varepsilon p)$ of them.

We will next prove that Algorithm 2 is a $(2+O(\varepsilon))$-approximation. Clearly, for any vertex $v \in P$ we have $\Pr[v \in P \cup M] = 1$. Moreover, any $v \notin P$ joins M with probability c_v (by Observation 3.3)

which implies $\Pr[v \in P \cup M] = c_v$. As a result

$$\mathbb{E}[|P \cup M|] = \sum_{v \in P} \Pr[v \in P \cup M] + \sum_{v \notin P} \Pr[v \in P \cup M] = \mathbb{E}[|P|] + \sum_{v \notin P} c_v. \tag{2}$$

Let us define α to be the fraction of the optimal solution not in P. That is

$$\alpha = \frac{\sum_{v \notin P} c_v}{\text{OPT}}.$$

Since, by definition, any vertex $v \in P$ satisfies $c_v \geq 0.5 - \varepsilon$ we have

$$\mathbb{E}[|P|] \times (0.5 - \varepsilon) \leq \sum_{v \in P} c_v = \text{OPT} - \sum_{v \notin P} c_v = (1 - \alpha)\text{OPT},$$

which gives us

$$\mathbb{E}[|P|] \leq \frac{(1 - \alpha)\text{OPT}}{0.5 - \varepsilon}.$$

Combining this with (2) gives us

$$\mathbb{E}[|P \cup M| = \mathbb{E}[|P|] + \sum_{v \notin P} c_v \leq \frac{(1 - \alpha)\text{OPT}}{0.5 - \varepsilon} + \alpha\text{OPT} = \text{OPT}\frac{1 - \alpha(0.5 + \varepsilon)}{0.5 - \varepsilon} = \text{OPT}(2 - \alpha + O(\varepsilon)), \tag{3}$$

and implies the $(2 + O(\varepsilon))$-approximation ratio. Observe that this bound is tight only when $\alpha = 0$. If for an instance of the problem, a large number of vertices have $c_v < 0.5 - \varepsilon$ and as a result α is large, this algorithm achieves a better approximation ratio.

3.1 Beating 2-approximation

As discussed above, Algorithm 2 has a better performance when a large portion of OPT comes from vertices with $c_v < 0.5 - \varepsilon$ while Algorithm 1 is almost the opposite. Therefore, an idea for beating the 2-approximation ratio is to run the best of these two. We will prove that doing so achieves an approximation ratio of $5/3 + O(\varepsilon)$. Given parameter $\varepsilon \in (0, 0.1)$, let us recall the definition of α as

$$\alpha = \frac{\sum_{v : c_v < 0.5 - \varepsilon} c_v}{\text{OPT}}. \tag{4}$$

Moreover, let S_1 and S_2 be the solutions outputted by Algorithm 1 and Algorithm 2 respectively. As an upper-bound for $|S_1|$ we have

$$\mathbb{E}[|S_1|] = \sum_{v \in V} \Pr[v \in S_1] \overset{(1)}{=} \sum_{v \in V} c_v(2 - c_v) \leq \sum_{v : c_v \geq 0.5 - \varepsilon} (1.5 + \varepsilon) \cdot c_v + \sum_{v : c_v < 0.5 - \varepsilon} 2 \cdot c_v$$
$$\overset{(4)}{=} (1.5 + \varepsilon)(1 - \alpha)\text{OPT} + 2\alpha\text{OPT} = \text{OPT}(1.5 + 0.5\alpha + O(\varepsilon)).$$

Moreover, by (3) we have

$$\mathbb{E}[|S_2|] = \text{OPT}(2 - \alpha + O(\varepsilon))$$

Therefore, the approximation ratio achieved by running the best of these two algorithms is upper-bounded by

$$\max\left[(2 - \alpha + O(\varepsilon)), (1.5 + 0.5\alpha + O(\varepsilon))\right]$$

We observe that this term is minimized for $\alpha = 1/3$ which results in an approximation ratio of $5/3 + O(\varepsilon)$. This analysis is tight since both algorithms achieve this approximation ratio when $2/3$ of OPT comes from vertices with $c_v = 0.5$ and the rest from vertices with $c_v \to 0$.

4 The $(3/2+\varepsilon)$-Approximation Algorithm

Inspired by the above algorithms, in this section, we design a 3/2-approximation algorithm. Throughout this section, we assume that c_vs are known in advance. We relax this assumption in Section 6 by directly estimating c_vs to an arbitrary desirable accuracy with polynomial number of calls to a minimum vertex cover oracle.

Similar to Algorithm 1 and Algorithm 2, this algorithm first picks a subset of vertices P and commits to including them in the final solution and then queries the edges not covered by them, i.e., $H = G[V \setminus P]$. The algorithm first picks a threshold τ which may vary for different instances. Based on this threshold and c_v of the vertices it commits to including the set $P = P_1 \cup P_2$. For vertices whose $c_v \in [1-\tau-\varepsilon, \tau]$, we use the style of Algorithm 1 and only include them in P_1 if they also belong to a minimum vertex cover of a hallucinated random subgraph. For the set of vertices with $c_v > \tau$, we use the style of Algorithm 2 and include all of them in P_2.

Consider a solution S for a given instance of the problem. If we claim that S is an $(3/2+\varepsilon)$-approximate solution, we need to show

$$\sum_{v\in V} \Pr[v \in S] \leq (3/2+\varepsilon)\cdot \textsc{opt} = (3/2+\varepsilon)\sum_{v\in V} c_v.$$

In other words, S should satisfy

$$\sum_{v\in V} \big((3/2+\varepsilon)\cdot c_v - \Pr[v\in S]\big) \leq 0.$$

Inspired by this, for any vertex v, we define the budget of this vertex as

$$b_v = \max\Big((3/2+\varepsilon)\cdot c_v - \Pr[v\in S],\, 0\Big) \tag{5}$$

and its cost as

$$\sigma_v = \max\Big(\Pr[v\in S] - (3/2+\varepsilon)\cdot c_v,\, 0\Big) \tag{6}$$

Proving that S is an 3/2-approximate solution is equivalent to showing that we can use the budget of vertices with $b_v > 0$, to pay the cost of the vertices with $\sigma_v > 0$. Formally,

Claim 4.1. *Let S be a vertex cover of $\mathcal{G}^\star$. Also, for any vertex $v \in V$, consider b_v and σ_v defined respectively in (5) and (6). If S satisfies*

$$\sum_{v\in V} b_v - \sum_{v\in V} \sigma_v \geq 0,$$

then $\mathbb{E}[|S|] \leq (3/2+\varepsilon)\textsc{opt}$.

Proof. Since for any vertex $v \in V$, we have

$$\begin{aligned} b_v - \sigma_v &= \max\Big((1.5+\varepsilon)\cdot c_v - \Pr[v\in S],\, 0\Big) - \max\Big(\Pr[v\in S] - (1.5+\varepsilon)\cdot c_v,\, 0\Big) \\ &= (1.5+\varepsilon)\cdot c_v - \Pr[v\in S], \end{aligned}$$

We get

$$\sum_{v\in V}(b_v-\sigma_v) = \sum_{v\in V}\Big((1.5+\varepsilon)\cdot c_v - \Pr[v\in S]\Big) = \sum_{v\in V}(1.5+\varepsilon)\cdot c_v - \sum_{v\in V}\Pr[v\in S]$$

$$= (1.5+\varepsilon)\text{OPT} + \mathbb{E}[|S|].$$

Therefore, inequality $\sum_{v\in V}(b_v - \sigma_v) \geq 0$ in the statement of this claim also implies

$$(1.5+\varepsilon)\text{OPT} + \mathbb{E}[|S|] \geq 0,$$

and as a result we have $\mathbb{E}[|S|] \leq (1.5+\varepsilon)\text{OPT}$, completing the proof of this claim. □

We observe that if S is found by Algorithm 1, then vertices with $c_v < (0.5-\varepsilon)$ have a positive cost. On the other hand, if S is found by Algorithm 2, all these vertices have a positive budget. Based on this observation, we want a threshold τ such that:

- If Algorithm 1 is run on $\{v \in V : c_v \in [1-\tau-\varepsilon, \tau]\}$, then we can pay the cost of vertices with $c_v < (0.5-\varepsilon)$ in this set using the budget of the ones with $c_v > (0.5-\varepsilon)$.
- If Algorithm 2 is run on the rest of the vertices, we can use the budget of the vertices with $c_v < 1-\tau-\varepsilon$ to pay the cost of the vertices with $c_v > \tau$.

We claim that setting τ to be the smallest number in $[0.5, 1]$ satisfying

$$\sum_{v:c_v>\tau} c_v \leq \sum_{v:c_v<1-\tau-\varepsilon} c_v$$

gives us these properties. However, clearly, we cannot just run two separate algorithms on these two subsets of vertices since there can potentially be a large number of edges between them. Therefore, we need to prove that following this intuition does not force us to query a large number of edges. We formally state our 3/2-approximate algorithm below as Algorithm 3. Later, in Section 5, we prove that for any $\varepsilon \in (0, 0.1)$ this algorithm outputs a $(3/2+\varepsilon)$-approximate vertex cover using only $O(n/\varepsilon p)$ queries.

Algorithm 3. Our 3/2-approximation algorithm.

1 Let $\mathcal{G}_1$ be a random realization of G containing any edge $e \in G$ independently with probability p_e.
2 $C \leftarrow \text{MVC}(\mathcal{G}_1)$.
3 Let τ be the smallest number in $[0.5, 1]$ such that $\sum_{v:c_v>\tau} c_v \leq \sum_{v:c_v<1-\tau-\varepsilon} c_v$.
4 $P \leftarrow \{v \in V : c_v > \tau\} \cup \{v \in V : c_v \in [1-\tau-\varepsilon, \tau] \text{ and } v \in C\}$.
5 Let H be the subgraph induced in G by $V \setminus P$.
6 Query edges in H and let $\mathcal{H}^\star$ be its realization
7 **return** $P \cup \text{MVC}(\mathcal{H}^\star)$.

First, since this algorithm follows our standard framework, by Observation 3.1 it outputs a vertex cover of $\mathcal{G}^\star$. Note that in this algorithm, any vertex in set $\{v \in V : c_v \in [1-\tau-\varepsilon, \tau]\}$ joins set P iff it is in vertex cover C, This is similar to the way Algorithm 1 constructs P. Therefore for any vertex v in this set, the probability of v joining P in Algorithm 3 is the same as that of Algorithm 1. On the other hand, from the rest of the vertices, i.e., $\{v \in V : c_v < 1-\tau-\varepsilon \text{ or } \tau < c_v]$, P includes any vertex with $\tau < c_v$. These are the only vertices in this set with $c_v \geq 0.5-\varepsilon$. Therefore, this is similar to the way Algorithm 1 constructs set P.

5 The Analysis

In the following lemma we prove that the number of edges queried by our algorithm is $O(n/\varepsilon p)$.

Lemma 5.1. *Subgraph H from Algorithm 3 satisfies $\mathbb{E}[|H|] = O(\frac{n}{\varepsilon p})$.*

Proof. Consider an edge $e = (u, v)$. We will first show that if $e \in H$, then either $c_e \leq 1 - \varepsilon$ or it is not covered in C. Assume w.o.l.g. that $c_v \geq c_u$. Since $e \in H$, we know that $v \notin P$ and $u \notin P$ which implies $c_v \leq \tau$ since otherwise v joins P. We will prove our claim by considering all possible values of c_u.

- $c_u \geq 1 - \tau - \varepsilon$: Since we know $c_v \geq c_u$ and $c_v \leq \tau$, in this case both c_u and c_v are in $[1-\tau-\varepsilon, \tau]$. Thus, e joins H iff both its end-points are not in C which means e is not covered in C.
- $c_u < 1 - \tau - \varepsilon$: Since we know $c_v \leq \tau$, this implies $c_v + c_u \leq 1 - \varepsilon$. Moreover, since for any edge $c_v + c_u \geq c_e$ holds, we get $c_e \leq 1 - \varepsilon$.

As a result, we have

$$|H| \leq |\{e : c_e \leq 1 - \varepsilon\}| + |\{e : e \text{ not covered by } C\}|.$$

Observation 3.2, states that the number of edges not covered by an MVC of a random realization of G is $O(n/p)$. This directly implies

$$\mathbb{E}[|\{e : e \text{ not covered by } C\}|] = O(n/p).$$

Since OPT itself is an MVC of a random realization of G, Observation 3.2 also implies

$$\mathbb{E}|\{e : e \text{ not covered by OPT}\}|] = \sum_e (1 - c_e) = O(n/p).$$

Using Markov's inequality, this gives us

$$\mathbb{E}[|\{e : c_e \leq 1 - \varepsilon\}|] = \mathbb{E}[|\{e : 1 - c_e > \varepsilon\}|] \leq \left(\sum_{e \in G} (1 - c_e)\right)/\varepsilon = O(n/\varepsilon p).$$

Putting these inequalities together, we conclude

$$\mathbb{E}[|H|] \leq \mathbb{E}[|\{e : c_e \leq 1 - \varepsilon\}| + |\{e : e \text{ not covered by } C\}|] = O(n/\varepsilon p),$$

completing the proof of this claim. □

As we mentioned before, for the sake of analysis, we need for the vertex cover of $\mathcal{H}^\star$ to include any vertex v with probability c_v. In order to achieve this, we need to make a slight change to the algorithm. We explain the modified algorithm below.

Algorithm 4. An algorithm used only for analysis.

1. Consider subgraph H and subset of vertices P from Algorithm 3
2. Query subgroup H and let $\mathcal{H}^\star$ be its realization.
3. Let $\bar{H} \leftarrow G \setminus H$
4. Let $\bar{\mathcal{H}}$ be a random realization of $\bar{H}$ containing each of its edges e independently with probability p_e.
5. $M \leftarrow \text{MVC}(\mathcal{H}^\star \cup \bar{\mathcal{H}})$.
6. **return** $M \cup P$.

In the rest of the paper we will prove our desired approximation ratio for Algorithm 4 instead of Algorithm 3. However, for that to work we first need the following observation.

Observation 5.2. *Let S_1 and S_2 be respectively the outputs of Algorithm 3 and Algorithm 4. We have $\mathbb{E}[|S_1|] \leq \mathbb{E}[|S_2|]$.*

Proof. Note that both S_1 and S_2 contain P and a vertex cover of $\mathcal{H}^\star$. Since the vertex cover of $\mathcal{H}^\star$ in Algorithm 3 is the smallest one, the output of this algorithm is not larger than that of Algorithm 4. □

We are now ready to prove the main lemma about the size of the solution outputted by Algorithm 4. The lemma is sated below.

Lemma 5.3. *Let S be the the output of Algorithm 4. We have*

$$E[|S|] \leq (3/2 + \varepsilon)\textsc{opt}.$$

Proof. By Claim 4.1, to prove this lemma it suffices to show

$$\sum_{v \in V} b_v - \sum_{v \in V} \sigma_v \geq 0$$

where b_v and σ_v, the budget and cost of vertex v are respectively defined in (5) and (6). To prove this, we divide the vertices of our graph V to three disjoint subsets V_1, V_2, and V_3 and prove this equation for them separately. That is, for any V_i we prove

$$\sum_{v \in V_i} b_v - \sum_{v \in V_i} \sigma_v \geq 0. \tag{7}$$

We define these subsets as follows:

- $V_1 = \{v : 0.5 - \varepsilon \leq c_v \leq 0.5\}$.
- $V_2 = \{v : c_v > \tau\} \cup \{v : c_v < 1 - \tau - \varepsilon\}$.
- $V_3 = \{v : 0.5 < c_v \leq \tau\} \cup \{v : 1 - \tau - \varepsilon \leq c_v < 0.5 - \varepsilon\}$. See a visualisation of these sets in Figure 2.

We will prove Equation (7) for subsets V_1, V_2 and V_3 respectively in Lemma 5.5, Lemma 5.6, and Lemma 5.7.

Since these three subsets are disjoint and satisfy $V = V_1 \cup V_2 \cup V_3$, we get

$$\sum_{v \in V} b_v - \sum_{v \in V} \sigma_v = \sum_i \left(\sum_{v \in V_i} b_v - \sum_{v \in V_i} \sigma_v \right) \geq 0$$

completing the proof of this lemma. □

Before stating the three aforementioned lemmas formally, we need the following claim which we will use to prove them.

Claim 5.4. *Consider τ defined in Algorithm 3, and let S be the output of Algorithm 4. For any vertex v with $c_v \in [1 - \tau - \varepsilon, \tau]$, we have $\Pr[v \in S] = c_v(2 - c_v)$.*

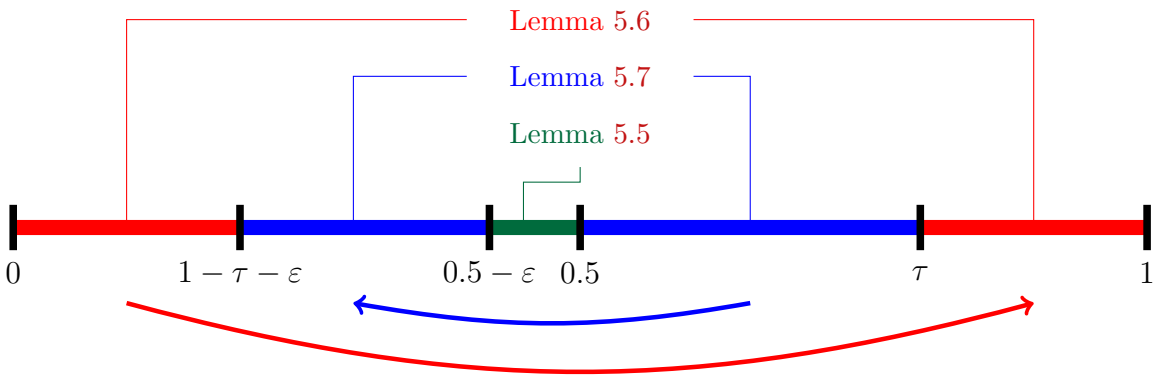

Figure 2: **Managing vertex costs for different values of c_v.** For set V_1 (the green area), we prove in Lemma 5.5 that the vertices in this set have no cost. For set V_2 (the red area), in Lemma 5.6 we use the budget of the vertices in $\{v : c_v < 1-\tau-\varepsilon\}$ to pay the cost of the vertices in $\{v : c_v > \tau\}$. Finally, for set V_3 (the blue area), in Lemma 5.7 we use the budget of vertices in $\{v : 0.5 < c_v \leq \tau\}$ to pay the cost of the vertices in $\{v : 1-\tau-\varepsilon \leq c_v < 0.5-\varepsilon\}$.

Proof. Consider M and P from Algorithm 4. Recall that the algorithm outputs $M \cup P$. That is $S = M \cup P$. Set P itself is defined in Algorithm 3 as

$$P = \{v \in V : c_v > \tau\} \cup \{v \in V : c_v \in [1-\tau-\varepsilon, \tau] \text{ and } v \in C\},$$

where C is a minimum vertex cover of a random realization of G. This implies that for any v with $c_v \in [1-\tau-\varepsilon, \tau]$, we have

$$\Pr[v \in S] = \Pr[v \in C \cup M].$$

Since M and C are minimum vertex covers of two independent realizations of G, we get

$$\Pr[v \in M] = \Pr[v \in C \cup M] = \Pr[v \in C] + \Pr[v \in M] - \Pr[v \in M] \times \Pr[v \in C] = 2c_v - c_v^2.$$

This concludes the proof of this claim. □

Lemma 5.5. *Consider τ from Algorithm 3, and define $V_1 = \{v : 0.5-\varepsilon \leq c_v \leq 0.5\}$. We have*

$$\sum_{v \in V_1} b_v - \sum_{v \in V_1} \sigma_v \geq 0,$$

where b_v and σ_v, the budget and cost of vertex v are defined in (5) and (6) with respect to the solution S outputted by Algorithm 4.

Proof. By Claim 5.4, for any vertex $v \in V_1$, we have $\Pr[v \in S] = c_v(2-c_v)$. Combining this with the fact that any vertex in V_1 satisfies $c_v > 0.5-\varepsilon$, we get

$$\Pr[v \in S] \leq c_v(2-0.5+\varepsilon) = c_v(1.5+\varepsilon).$$

This means that for any vertex $v \in V_1$, we have $b_v = (1.5+\varepsilon)c_v - \Pr[v \in S] \geq 0$ and $\sigma_v = 0$, and as a result

$$\sum_{v \in V_1} b_v - \sum_{v \in V_1} \sigma_v \geq 0,$$

completing the proof of this claim. □

Lemma 5.6. *Consider τ defined in Algorithm 3, and let $V_2 = \{v : c_v > \tau\} \cup \{v : c_v < 1 - \tau - \varepsilon\}$. This set satisfies*

$$\sum_{v \in V_2} b_v - \sum_{v \in V_2} \sigma_v \geq 0,$$

where b_v and σ_v, the budget and cost of vertex v are defined in (5) and (6) with respect to the solution S outputted by Algorithm 4.

Proof. Let us start by defining

$$A = \{v : c_v > \tau\} \quad \text{and} \quad B = \{v : c_v < 1 - \tau - \varepsilon\},$$

where $V_2 = A \cup B$. Consider sets P and M from Algorithm 4 which form its output. That is $S = M \cup P$. Note that by definition of P we have $A \in P$ and as a result $\Pr[v \in S] = 1$ for any $v \in A$. Therefore,

$$b_v - \sigma_v \overset{(6),(5)}{=} \max\Big((1.5+\varepsilon)c_v - \Pr[v \in S], 0\Big) - \max\Big(\Pr[v \in S] - (1.5+\varepsilon)c_v, 0\Big) = (1.5+\varepsilon)c_v - 1.$$

Moreover, we have $B \cap P = \emptyset$ and by Observation 3.3, for any $u \in B$ we have $\Pr[u \in M] = c_u$. This implies $\Pr[u \in S] = c_u$ and as a result the budget of vertex u is

$$b_u = \max\Big((1.5+\varepsilon)c_u - \Pr[v \in P \cup M],\, 0\Big) = (1.5+\varepsilon)c_u - c_u = c_u(0.5+\varepsilon),$$

and $\sigma_u = 0$. Putting these together gives us

$$\begin{aligned}\sum_{v \in V_2}(b_v - \sigma_v) &= \sum_{v \in A}(b_v - \sigma_v) + \sum_{u \in B}(b_u - \sigma_u) = \sum_{v \in A}\big((1.5+\varepsilon)c_v - 1\big) + \sum_{u \in B}(0.5+\varepsilon)c_u \\ &= (1.5+\varepsilon)\sum_{v \in A} c_v - |A| + (0.5+\varepsilon)\sum_{u \in B} c_u.\end{aligned}$$

Note that by definition of τ, we have $\sum_{v \in A} c_v \leq \sum_{u \in B} c_u$. Thus, we can write

$$\begin{aligned}\sum_{v \in V_2}(b_v - \sigma_v) &= (1.5+\varepsilon)\sum_{v \in A} c_v - |A| + (0.5+\varepsilon)\sum_{u \in B} c_u \\ &\geq (2+2\varepsilon)\sum_{v \in A} c_v - |A|.\end{aligned}$$

Since for any vertex $v \in A$ we have $c_v > \tau \geq 0.5$, this implies

$$\sum_{v \in V_2}(b_v - \sigma_v) \geq (2+2\varepsilon)\sum_{v \in A} c_v - |A| \geq (2+2\varepsilon)\sum_{v \in A} 0.5 - |A| \geq \varepsilon|A| \geq 0.$$

This concludes the proof. □

Lemma 5.7. *Consider τ defined in Algorithm 3, and let*

$$V_3 = \{v : 0.5 < c_v \leq \tau\} \cup \{v : 1 - \tau - \varepsilon \leq c_v < 0.5 - \varepsilon\},$$

This set satisfies

$$\sum_{v \in V_3} b_v - \sum_{v \in V_3} \sigma_v \geq 0,$$

where b_v and σ_v, the budget and cost of vertex v are defined in (5) and (6) with respect to the solution S outputted by Algorithm 4.

Due to having a detailed and lengthy proof, we designate Section 5.1 to the proof of this lemma. Below, we restate our main theorem and give a formal proof for the approximation ratio and the number of queries that our algorithm requires. Later in Section 6, we explain how we can get the same bounds using only $(n \log n/\varepsilon^2)$ calls to the MVC oracle.

Theorem 1.1 (restated). *For any $\varepsilon \in (0, 0.1)$, Algorithm 3 finds a vertex cover of $\mathcal{G}^\star$ with the expected size of at most $(1.5+\varepsilon)\text{OPT}$ by querying $O(n/\varepsilon p)$ total edges.*

Proof. Due to Lemma 5.1, we know that Algorithm 3 only requires $O(n/\varepsilon p)$ queries. Let S be the solution outputted by Algorithm 3. By Observation 3.1, S is a vertex cover of $\mathcal{G}^\star$ and by Observation 5.2 its expected size is upper-bounded by the output of Algorithm 4. In Lemma 5.3, we prove that the output of Algorithm 4 is upper-bounded by $(1.5+\varepsilon)\text{OPT}$. Putting these together implies that S is a vertex cover of $\mathcal{G}^\star$ with the expected size of at most $(1.5+\varepsilon)\text{OPT}$. □

5.1 Proof of Lemma 5.7

Let us define subsets

$$A = \{v : 0.5 < c_v \le \tau\} \quad \text{and} \quad B = \{v : 1 - \tau - \varepsilon \le c_v < 0.5 - \varepsilon\},$$

where $V_3 = A \cup B$. By Claim 5.4, for any vertex $v \in V_3$, we have $\Pr[v \in S] = c_v(2 - c_v)$, thus

$$\Pr[v \in S] - (1.5+\varepsilon)c_v = c_v(2-c_v) - (1.5+\varepsilon)c_v = (0.5-\varepsilon)c_v - c_v^2.$$

Observe that for vertices in $v \in A$ this term is non-positive, therefore this vertex has a zero cost and a budget of

$$b_v = c_v^2 - (0.5-\varepsilon)\cdot c_v.$$

On the other hand since $(0.5-\varepsilon)c_u - c_u^2 > 0$ holds for any vertex $u \in B$, this vertex has a zero budget and a cost of

$$\sigma_u = (0.5-\varepsilon)\cdot c_u - c_u^2$$

Now, we need to we will show that we can use the budget of vertices in A to pay the cost of the vertices in B. Proving that this is possible heavily relies on the way threshold τ is chosen.

To complete the proof of this lemma, we need the two following claims.

Claim 5.8. *For any pair of vertices $u \in B$ and $v \in A$, if $c_u \ge 1 - c_v - \varepsilon$, then*

$$b_v - \sigma_u \ge (\sigma_u/c_u)(c_v - c_u).$$

Proof. We start by proving $\sigma_u/c_u \le b_v/c_v$ as follows.

$$\begin{aligned}
\sigma_u/c_u &= \frac{(0.5-\varepsilon)\cdot c_u - c_u^2}{c_u} = 0.5 - \varepsilon - c_u \\
&\le 0.5 - \varepsilon - (1 - c_v - \varepsilon) = c_v - 0.5 = \frac{c_v^2 - (0.5-\varepsilon)\cdot c_v}{c_v} - \varepsilon \\
&\le b_v/c_v - \varepsilon \\
&< b_v/c_v. \qquad (8)
\end{aligned}$$

Thus, we can write

$$b_v - \sigma_u = c_v(b_v/c_v) - c_u(\sigma_u/c_u) \overset{(8)}{\geq} (\sigma_u/c_u)(c_v - c_u),$$

completing the proof of this claim. □

Claim 5.9. *Let us sort the vertices in B in the increasing order of their c_u with u_i denoting the i-th vertex. We claim that for any $i \in |B|$ it is possible to pay the cost of the vertices in $B_i = \{u \in B : c_u \leq c_{u_i}\}$ with the budget of vertices in $A_i = \{v \in A : c_v \geq 1 - c_{u_i} - \varepsilon\}$. That is*

$$\sum_{v \in A_i} b_v - \sum_{u \in B_i} \sigma_u \geq 0. \tag{9}$$

Proof. In order to prove this claim, we prove the following stronger inequality via induction.

$$\sum_{v \in A_i} b_v - \sum_{u \in B_i} \sigma_u \geq (\sigma_{u_i}/c_{u_i}) \left(\sum_{v \in A_i} c_v - \sum_{u \in B_i} c_u \right). \tag{10}$$

Doing so proves this claim since by definition of τ, for any $i \in |B|$ we have

$$\sum_{v \in A_i} c_v - \sum_{u \in B_i} c_u \geq 0. \tag{11}$$

As the base case of $i = 1$, we need to prove that Equation 9 holds for $A_1 = \{v \in A : c_v \geq 1 - c_{u_1} - \varepsilon\}$ and $B_1 = \{u_1\}$. That is

$$\sum_{v \in A_1} b_v - \sigma_{u_1} \geq (\sigma_{u_1}/c_{u_1}) \left(\sum_{v \in A_1} c_v - c_{u_1} \right). \tag{12}$$

Since for any $v \in A_1$, we have $c_v \geq 1 - c_{u_1} - \varepsilon$, this inequality follows from Claim 5.8 proving our base case. Now, as the induction step, we will prove Equation (9) for $i = j$ assuming that it holds for $i = j - 1$. We can write

$$\begin{aligned}
\sum_{v \in A_j} b_v - \sum_{u \in B_j} \sigma_u &= \sum_{v \in A_{j-1}} b_v + \sum_{v \in B_j \setminus B_{j-1}} b_v - \sum_{u \in B_{j-1}} \sigma_u - \sigma_{u_j} \\
&\geq (\sigma_{u_{j-1}}/c_{u_{j-1}}) \left(\sum_{v \in A_{j-1}} c_v - \sum_{u \in B_{j-1}} c_u \right) + \left(\sum_{v \in B_j \setminus B_{j-1}} b_v - \sigma_{u_j} \right) \\
&\geq (\sigma_{u_j}/c_{u_j}) \left(\sum_{v \in A_{j-1}} c_v - \sum_{u \in B_{j-1}} c_u \right) + \left(\sum_{v \in B_j \setminus B_{j-1}} b_v - \sigma_{u_j} \right),
\end{aligned}$$

where the second inequality is due to the induction hypothesis and the last one is due to

$$\sigma_{u_{j-1}}/c_{u_{j-1}} \geq \sigma_{u_j}/c_{u_j}.$$

To prove the induction step, it suffices to show that the following holds.

$$\sum_{v \in B_j \setminus B_{j-1}} b_v - \sigma_{u_j} \geq \left(\sigma_{u_j}/c_{u_j}\right) \left(\sum_{v \in B_j \setminus B_{j-1}} c_v - c_{u_j} \right).$$

Note that this is a general version of Equation (12) above. Observe that by definition, for any $u \in B_j \setminus B_{j-1}$ we have $c_v \geq 1 - c_{u_j} - \varepsilon$. Thus, by Claim 5.8, we get

$$\sum_{v \in B_j \setminus B_{j-1}} b_v - \sigma_{u_j} \geq \sum_{v \in B_j \setminus B_{j-1}} (\sigma_{u_j}/c_{u_j})(c_v - c_{u_j}) \geq \left(\sigma_{u_j}/c_{u_j}\right) \left(\sum_{v \in B_j \setminus B_{j-1}} c_v - c_{u_j} \right).$$

This concludes the induction step and the proof of this lemma. □

Observe that proving Claim 5.9 also completes the proof of Lemma 5.7. Let $x = |B|$. Correctness of Claim 5.9 for $i = x$ implies

$$\sum_{v \in A} b_v - \sum_{u \in B} \sigma_u \geq 0.$$

Since vertices in A have a zero cost, this also implies

$$\sum_{v \in V_3} (b_v - \sigma_u) \geq 0,$$

completing the proof of Lemma 5.7. □

6 Limiting the Number of Calls to the MVC Oracle

In this section, we discuss how we can implement Algorithm 3 using only a polynomial number of calls to the MVC oracle. In Algorithm 3, we need to know c_v of the vertices in V. We can compute them exactly if we do not limit the number of calls to the MVC oracle. However, we claim that it is possible to use an estimated value of these parameters in Algorithm 3 and still get the same bounds. In the algorithm below we first find an estimate for any c_v and then feed them to Algorithm 3.

Algorithm 5. Oracle-efficient 3/2-approximation algorithm

1 Draw $t = \frac{n^2}{8\varepsilon^2} \ln(2n/\delta)$ realizations of G and denote them by $\mathcal{G}_1, \ldots, \mathcal{G}_t$.
2 For any $i \in [t]$, let $C_i = \text{MVC}(\mathcal{G}_1)$.
3 For any vertex $v \in V$, let $\bar{c}_v$ be be the fraction of C_i's that contain v.
4 Run Algorithm 3 with parameters $\bar{c}_v$s and $\varepsilon' = \varepsilon/2$.

We first show that Algorithm 5 returns $\bar{c}_v$s that are within $\varepsilon/2n$ of the respective c_vs, with high probabity.

Claim 6.1. *With probability $1 - \delta$, for all $v \in V$, we have $|c_v - \bar{c}_v| \leq \varepsilon/2n$.*

Proof. This proof follows from a simple application of the Hoeffding bound. Let $\alpha = \varepsilon/2n$. Let $X_i^v = 1(v \in C_i)$ be an indicator variable for whether $v \in C_i$. Note that X_i^v is a Bernoulli random

variable with $\mathbb{E}[X_i^v] = c_v$ for any i and $v \in V$. Furthermore, $\bar{c}_v = \frac{1}{t}\sum_{i=1}^{t} X_i^v$. Using Hoeffding and union bound, we have

$$\Pr[\exists v \in V \text{ s.t. } |\bar{c}_v - c_v| \geq \alpha] \leq n \cdot \Pr\left[\left|\frac{1}{t}\sum_{i=1}^{t} X_i^v - c_v\right| \geq \alpha\right] \leq 2n\exp\left(-2t\alpha^2\right) \leq \delta,$$

where the last inequality is by the choice of $t = \frac{1}{2\alpha^2}\ln\left(\frac{2n}{\delta}\right)$.

□

Next, we show that Algorithm 5 returns an a minimum vertex cover of $\mathcal{G}^\star$ which w.h.p. has expected size of at most $(3/2+\varepsilon)\textsc{opt}$ and queries only $O(n/p\varepsilon)$.

Theorem 6.2. *For any $\varepsilon \in (0, 0.1)$, Algorithm 5 finds a vertex cover of $\mathcal{G}^\star$ with the expected size of at most $(3/2+\varepsilon)\textsc{opt}$ by querying $O(n/\varepsilon p)$ total edges. Moreover, this algorithm is oracle-efficient.*

Proof sketch. By Claim 6.1, with high probability for all $v \in V$, $|c_v - \bar{c}_v| \leq \varepsilon/2n$. For the rest of the proof, we condition on this high probability event.

We will first give an upper-bound on the number of queries (edges in H for Algorithm 3). Let $\alpha = \varepsilon/2n$. Following the proof of Lemma 5.1, it is easy to verify that if $e = (u, v) \in H$, then either $\bar{c}_v + \bar{c}_u \leq 1 - \varepsilon'$ or it is not covered in C. By our assumption, $|c_v - \bar{c}_v| \leq \alpha$, therefore either e satisfies $c_v + c_u \leq 1 - \varepsilon' + 2\alpha \leq 1 - \varepsilon$ or it is not covered in C. Similar to the proof of Lemma 5.1, this gives us $|H| = O(n/\varepsilon p)$.

We next bound the approximation ratio of Algorithm 5. Consider a hypothetical stochastic process for generating a subgraph $\bar{\mathcal{G}}$ that chooses a $\mathcal{G}_i$ of Algorithm 5 uniformly at random. We note that for this stochastic process, $\bar{c}_v = \Pr[v \in \text{MVC}(\bar{\mathcal{G}})]$. Moreover, let $\overline{\textsc{opt}} = \mathbb{E}[|\text{MVC}(\bar{\mathcal{G}})|]$. Now assume that we run Algorithm 3 with $\bar{c}_v$s. We note that the approximation guarantees of our Algorithm 3 holds for any stochastic process for generating a random subgraph (the independence is only used for bounding the number of queries). Therefore, the outcome of Algorithm 3 is of size at most $(3/2+\varepsilon/2)\overline{\textsc{opt}}$. Moreover, by Claim 6.1 $\textsc{opt} \in [\overline{\textsc{opt}} - \varepsilon/2, \overline{\textsc{opt}} + \varepsilon/2]$, the outcome of Algorithm 4 is at most $(3/2+\varepsilon)\textsc{opt}$. This proves our approximation guarantees.

7 Tightness Under Mild Correlation

In the previous sections, we exhibited an algorithm that gives a $(3/2+\varepsilon)$-approximation for stochastic graphs that have independently realized edges. Indeed, the analysis given in the previous section continues to hold for graphs with a small number of correlated edges. In this section, we show that for such graphs, a $(3/2+\varepsilon)$ approximation factor is tight. That is, we exhibit a stochastic graph with just a few correlated edges and show that any non-adaptive algorithm must have an approximation factor of $(3/2-\varepsilon)$ on this graph with high probability. Our arguments are based on the arguments given in Section 6 of [AKL16].

Definition 7.1 (Mildly Correlated Graph)**.** *We say that an stochastic graph $G = (V, E)$ is mildly correlated if the edge set E can be partitioned into sets E_1 and E_2 such that the following are satisfied:*

- *The edges in E_1 are realized independently from each other: for any $S_1 \subseteq E_1$*

$$\Pr \bigcap_{e \in S_1} \{e \in G_r\} = \prod_{e \in S_1} \Pr[e \in G_r]$$

- *The edges in E_1 are realized independently from those in E_2: for any $S_1 \subseteq E_1$ and $S_2 \subseteq E_2$,*

$$\Pr\left[\bigcap_{e\in S_1}\{e \in G_r\} \;\middle|\; \bigcap_{e\in S_2}\{e \in G_r\}\right] = \prod_{e\in S_1}\Pr[e \in G_r]$$

- *E_2 is small: $|E_2| = O(n)$.*

Notably, in our definition of a mildly correlated graph, the realizations of edges in E_2 may depend on those in E_1: we make no assumptions on probabilities of the form

$$\Pr\left[\bigcap_{e\in S_2}\{e \in G_r\} \;\middle|\; \bigcap_{e\in S_1}\{e \in G_r\}\right],$$

where $S_1 \subseteq E_1$ and $S_2 \subseteq E_2$.

Remark 7.2. *Given any mildly correlated stochastic graph G, and a parameter $\varepsilon \in (0, 0.1)$, Algorithm 3, outputs a vertex cover of $\mathcal{G}^\star$ with expected size of at most $(3/2 + \varepsilon)$ using only $O(n/\varepsilon p)$ queries without knowledge of the edge partitions E_1 and E_2.*

Proof. To see this, we first show that Algorithm 4 queries at most $O(n/\varepsilon p)$ edges. Looking more closely at our argument in Lemma 5.1, our only use of the independence assumption comes from our use of Observation 3.2. Thus, to show that Lemma 5.1 holds in the case of mildly correlated graphs, it suffices to prove that Observation 3.2 holds in this setting as well. Formally, we show that if $\mathcal{G}$ is a random realization of a mildly correlated stochastic graph G, then the number of edges in G not covered by M is at most $O(n/p)$. As before, let $H = G[V \setminus M]$ be the subgraph induced by the complement of the vertex cover M. Again, we must have that none of the edges of H are realized in $\mathcal{G}$ else one of the vertices of H must lie in the vertex cover M. Letting E_1 and E_2 be the subsets of the graph's edge set E as guaranteed by definition 7.1, we define $H_1 = H \cap G[E_1]$ and $H_2 = H \cap G[E_2]$. Notice that as $|E_2| = O(n)$, we find that

$$\Pr[|H| \geq O(n/p)] = P[|H_1| + |H_2| \geq O(n/p)] \leq \Pr[|H_1| \geq O(n/p)] \leq (2/e)^n$$

where in the final inequality, we invoke Observation 3.2 on the stochastic graph $G[E_1]$, which does have independently realized edges. Thus, $|H|$ does remain small even in this setting, and Lemma 5.1 can be applied directly to see that we only sample $O(n/\varepsilon p)$ edges.

Next, we show that the approximation guarantee continues to hold in this setting. The remaining analysis presented in Section 5 follows with just a simple change to Algorithm 4: when we sample realizations $\bar{\mathcal{H}}$ from $\bar{H}$ in line 4 of the algorithm, we should do so conditioned on the realization $\mathcal{H}^\star$ that we obtain in line 2. We note that our algorithm (Algorithm 3) does not actually require such capability to function, and we only use Algorithm 4 for the purposes of analysis. Once we make this change, we again have that the vertex cover M given in Algorithm 4 is drawn independently from the same distribution as the true minimum vertex cover, which is all that is required for the remaining analysis to follow. Thus, Algorithm 3 gives the desired approximation ratio of $1.5 + \varepsilon$ while only sampling $O(n/\varepsilon p)$ edges. □

Theorem 1.2 (restated). *There exists a mildly correlated stochastic graph G for which every non-adaptive algorithm must have an approximation ratio of at least $1.5-\varepsilon$ with probability $1-o(1)$.*

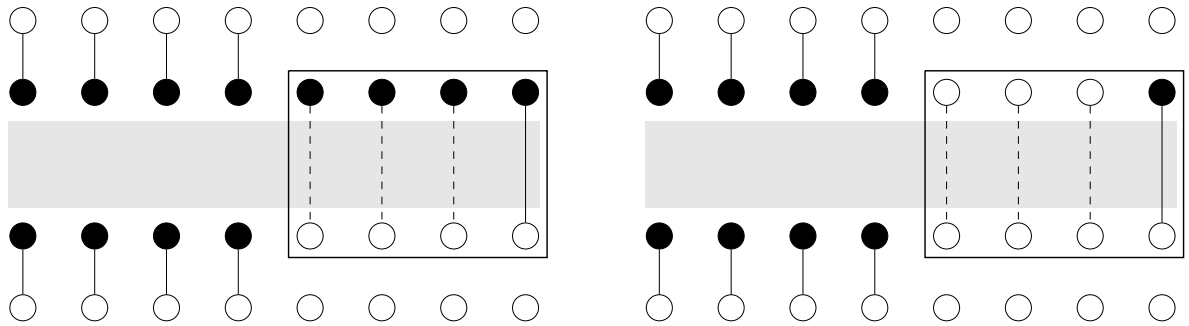

Figure 3: **A graphical depiction of G.** The middle $2n$ vertices (with edges depicted by the gray rectangle) is given by the Ruzsa-Szemerédi graph, and its edges are realized independently. We then select M^* (boxed) uniformly at random from the induced matchings, and realize the corresponding exterior edges for all vertices not in M^*. With high probability, any algorithm that non-adaptively queries only $O(n)$ edges must cover almost every edge of M^* (as depicted on the left, with vertices in the cover shown in black), but only the few edges in M^* that are actually realized must be covered (as depicted on the right).

Proof. We define G as follows. First, let an (r, t)-Ruzsa-Szemerédi graph be a bipartite graph on $2n$ vertices whose edge set may be partitioned into t induced matchings of size r. Such graphs exist for $r = \frac{n}{2} - \varepsilon_1$ and $t = n^{1+\Omega(\log\log n)}$ [GKK12]. We define the base graph of G by starting with such a graph, and then augmenting it by adding one additional vertex and exterior edge for each of the $2n$ vertices in the Rusza-Szemerèdi graph. We then realize all of the edges of the Rusza-Szemerèdi graph independently with $p_e = \varepsilon_2$ for all edges e in the edge set. Next, we select one of these induced matchings $M_1, M_2, \ldots, M_t$ at random, and call it M^*. For each of the $O(n)$ vertices of the Rusza-Szemerèdi graph that do not participate in M^*, we realize its respective exterior edge. It is easy to see that this stochastic graph is mildly correlated, following Definition 7.1 with E_1 denoting the edges of the Ruzsa-Szemerédi graph and E_2 denoting the exterior edges.

To see that every non-adaptive algorithm must have an approximation ratio of at least $1.5 - \varepsilon$, observe that as the algorithm may query at most $O(n) = o(r \cdot t)$ edges, it follows from a simple counting argument that the set of edges that the algorithm queries must contain $o(r)$ edges for all but an $o(1)$ fraction of the matchings $M_1, \ldots, M_t$. Thus, with probability $1 - o(1)$, the algorithm must cover $r - o(r)$ edges in M^*, and must further cover every exterior edge corresponding to vertices not in M^*. As the edges in M^* and the exterior edges do not coincide, it follows that with probability $1 - o(1)$, the size of the vertex cover returned by any non-adapative algorithm must be equal to

$$\underbrace{2(n-r)}_{\#\text{ exterior edges that must be covered}} + \underbrace{r - o(r)}_{\#\text{ edges in } M^* \text{ that must be covered}} = 1.5n + \varepsilon_1 - o(n)$$

However, observe that by the Chernoff bound, the probability that more than $2n\varepsilon_2$ edges in M^* are realized is upper bounded by $o(1)$. Thus, with probability $1 - o(1)$, the size of the minimum vertex cover of G is given by

$$\underbrace{2(n-r)}_{\#\text{ exterior edges that must be covered}} + \underbrace{2n\varepsilon_2}_{\#\text{ edges in } M^* \text{ that must be covered}} = n(1 + 2\varepsilon_2) + \varepsilon_1$$

It thus follows by the union bound that with probability $1 - o(1)$ any such algorithm must have an

approximation ratio of

$$\frac{1.5n + \varepsilon_1 - o(n)}{n(1+2\varepsilon_2) + \varepsilon_1} \to \frac{1.5}{1+2\varepsilon_2} \geq 1.5 - \varepsilon$$

for sufficiently large n, and appropriate choice of ε_2 (note here that this parameter does not depend on n, only ε). □

References

[AB19] Sepehr Assadi and Aaron Bernstein. Towards a Unified Theory of Sparsification for Matching Problems. In *2nd Symposium on Simplicity in Algorithms, SOSA@SODA 2019, January 8-9, 2019 - San Diego, CA, USA*, pages 11:1–11:20, 2019.

[AKL16] Sepehr Assadi, Sanjeev Khanna, and Yang Li. The Stochastic Matching Problem with (Very) Few Queries. In *Proceedings of the 2016 ACM Conference on Economics and Computation, EC '16, Maastricht, The Netherlands, July 24-28, 2016*, pages 43–60, 2016.

[AKL17] Sepehr Assadi, Sanjeev Khanna, and Yang Li. The Stochastic Matching Problem: Beating Half with a Non-Adaptive Algorithm. In *Proceedings of the 2017 ACM Conference on Economics and Computation, EC '17, Cambridge, MA, USA, June 26-30, 2017*, pages 99–116, 2017.

[BBD22] Soheil Behnezhad, Avrim Blum, and Mahsa Derakhshan. Stochastic vertex cover with few queries. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1808–1846. SIAM, 2022.

[BD20] Soheil Behnezhad and Mahsa Derakhshan. Stochastic weight matching: $(1-\varepsilon)$-approximation. In *Foundations of Computer Science (FOCS 20), to appear*, 2020.

[BDF+19] Soheil Behnezhad, Mahsa Derakhshan, Alireza Farhadi, MohammadTaghi Hajiaghayi, and Nima Reyhani. Stochastic Matching on Uniformly Sparse Graphs. In *Algorithmic Game Theory - 12th International Symposium, SAGT 2019, Athens, Greece, September 30 - October 3, 2019, Proceedings*, pages 357–373, 2019.

[BDH+15] Avrim Blum, John P. Dickerson, Nika Haghtalab, Ariel D. Procaccia, Tuomas Sandholm, and Ankit Sharma. Ignorance is Almost Bliss: Near-Optimal Stochastic Matching With Few Queries. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation, EC '15, Portland, OR, USA, June 15-19, 2015*, pages 325–342, 2015.

[BDH20a] Soheil Behnezhad, Mahsa Derakhshan, and MohammadTaghi Hajiaghayi. Stochastic Matching with Few Queries: $(1-\varepsilon)$-approximation. In *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 1111–1124, 2020.

[BDH+20b] Avrim Blum, John P. Dickerson, Nika Haghtalab, Ariel D. Procaccia, Tuomas Sandholm, and Ankit Sharma. Ignorance Is Almost Bliss: Near-Optimal Stochastic Matching with Few Queries. *Operations Research*, 68(1):16–34, 2020.

[BFHR19] Soheil Behnezhad, Alireza Farhadi, MohammadTaghi Hajiaghayi, and Nima Reyhani. Stochastic Matching with Few Queries: New Algorithms and Tools. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2855–2874, 2019.

[BR18] Soheil Behnezhad and Nima Reyhani. Almost Optimal Stochastic Weighted Matching with Few Queries. In *Proceedings of the 2018 ACM Conference on Economics and Computation, Ithaca, NY, USA, June 18-22, 2018*, pages 235–249, 2018.

[Gha19] Mohsen Ghaffari. Distributed maximal independent set using small messages. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 805–820. SIAM, 2019.

[GKK12] Ashish Goel, Michael Kapralov, and Sanjeev Khanna. On the communication and streaming complexity of maximum bipartite matching. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 468–485, 2012.

[GV04] Michel X. Goemans and Jan Vondrák. Covering minimum spanning trees of random subgraphs. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004, New Orleans, Louisiana, USA, January 11-14, 2004*, pages 934–941, 2004.

[GV06] Michel X. Goemans and Jan Vondrák. Covering minimum spanning trees of random subgraphs. *Random Struct. Algorithms*, 29(3):257–276, 2006.

[Von07] Jan Vondrák. Shortest-path metric approximation for random subgraphs. *Random Struct. Algorithms*, 30(1-2):95–104, 2007.

[YM18] Yutaro Yamaguchi and Takanori Maehara. Stochastic Packing Integer Programs with Few Queries. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 293–310, 2018.