Query Complexity of Stochastic Minimum Vertex Cover

Mahsa Derakhshan
Northeastern University, Boston, MA, USA

Mohammad Saneian
Northeastern University, Boston, MA, USA

Zhiyang Xun
University of Texas at Austin, TX, USA

Abstract

We study the stochastic minimum vertex cover problem for general graphs. In this problem, we are given a graph $G = (V, E)$ and an existence probability p_e for each edge $e \in E$. Edges of G are realized (or exist) independently with these probabilities, forming the realized subgraph $\mathcal{G}$. The existence of an edge in $\mathcal{G}$ can only be verified using edge queries. The goal of this problem is to find a near-optimal vertex cover of $\mathcal{G}$ using a small number of queries.

Previous work by Derakhshan, Durvasula, and Haghtalab [STOC 2023] established the existence of $1.5 + \varepsilon$ approximation algorithms for this problem with $O(n/\varepsilon)$ queries. They also show that, under mild correlation among edge realizations, beating this approximation ratio requires querying a subgraph of size $\Omega(n \cdot \text{RS}(n))$. Here, $\text{RS}(n)$ refers to Ruzsa-Szemerédi Graphs and represents the largest number of induced edge-disjoint matchings of size $\Theta(n)$ in an n-vertex graph.

In this work, we design a simple algorithm for finding a $(1 + \varepsilon)$ approximate vertex cover by querying a subgraph of size $O(n \cdot \text{RS}(n))$ for any absolute constant $\varepsilon > 0$. Our algorithm can tolerate up to $O(n \cdot \text{RS}(n))$ correlated edges, hence effectively completing our understanding of the problem under mild correlation.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms

Keywords and phrases Sublinear algorithms, Vertex cover, Query complexity

Digital Object Identifier 10.4230/LIPIcs.ITCS.2025.41

Funding *Zhiyang Xun*: Supported by NSF Grant CCF-2312573 and a Simons Investigator Award (#409864, David Zuckerman).

Acknowledgements The authors thank Sanjeev Khanna for countless insightful discussions and help.

1 Introduction

We study the *stochastic minimum vertex cover* problem. In this problem, we are given an arbitrary n-vertex graph $G(V, E)$ and an existence probability p_e for each edge $e \in E$. Edges of G are realized or exist independently, with their corresponding probability forming the realized subgraph $\mathcal{G}$. While $\mathcal{G}$ is unknown, one can query an edge $e \in E$ to see if this edge exists in $\mathcal{G}$. The goal is to find a near-optimal vertex cover of $\mathcal{G}$ by querying a *small* subset of E.

There is extensive literature studying various graph problems in this stochastic setting, such as minimum spanning tree [21, 22], all-pairs shortest paths [24], and maximum matching [13, 5, 12, 25, 3, 6, 9, 10, 8, 16, 15]. The stochastic vertex cover for general graphs was first studied by Blum, Behnezhad, and Derakhshan [7]. They provided a 2-approximation algorithm with $O(n/p)$ queries where $p = \min_{e \in E} p_e$. Later, this approximation ratio was improved to $1.5 + \varepsilon$ by the work of Derakhshan, Durvasula, and Haghtalab [14] with $O(n/\varepsilon p)$

queries. Even though these stochastic settings are primarily concerned with information-theoretical questions, this was the first work to provide positive results for a computationally hard problem. However, the key unresolved question remains:

▶ **Question 1.** *How many non-adaptive edge queries are needed to find a* $(1+\varepsilon)$*-approximation to the minimum vertex cover problem in the stochastic setting?*

In this paper, we take a step toward answering this question. We design a simple algorithm and establish a relation between the number of queries it makes and the density of a well-known class of graphs called Ruzsa-Szemerédi Graphs (Definition 2). Our number of queries matches the lower bound of Derakhshan, Haghtalab, and Durvasula [14] for the number of queries needed to surpass a 1.5-approximation in the presence of $O(n)$ correlated edges.

▶ **Definition 2** (Ruzsa-Szemerédi Graphs [23]). *An* n*-vertex graph* $G(V, E)$ *is an* $RS_n(r, t)$ *graph if its edge set* E *can be decomposed into* t *edge-disjoint induced matchings of size* r*. We use* $RS_n(r)$ *to denote the maximum* t *for which* $RS_n(r, t)$ *exists.*

Our main result is formalized in the theorem below.

▶ **Result 3.** *Given any* $\varepsilon \in (0, 1)$ *let* t *be* $RS_n(cn)$ *for* $c = \frac{\varepsilon^2}{128}$*. There exists an algorithm that queries a subgraph of size* $O(tn/p)$ *and finds a vertex cover with the expected approximation ratio of* $(1 + \varepsilon)$*.*

Proving lower and upper bounds parameterized by the density of RS-graphs has been of interest in various communities, such as dynamic algorithms [11, 4], streaming algorithms [20, 1, 2], property testing [17], and additive combinatorics [19]. Despite this, our understanding of their density remains incomplete. What we currently know is:

$$n^{\Omega_c(1/\log\log(n))} \overset{[17]}{\leq} RS_n(cn) \overset{[18]}{\leq} n/\log^{(O(\log(1/c)))}$$

where $\log^{(x)}$ is the x-iterated logarithmic function. Our result is most interesting if the lower-bound turns out to be tight. In such a case the number of edges queried by our algorithms is

$$|Q| = O(n^{1+\Omega_c(1/\log\log(n))}/p) = O(n^{1+o(1)}/p).$$

On the opposite end, if the actual density of RSgraphs is closer to the known upper-bound, our result implies $|Q| = O(\frac{n^2}{p\log^{O(\log(1/c))}})$.

1.1 Technical Overview

We first design an algorithm with an additive error of at most εn in Section 2 and later show how it can be modified to obtain the $(1 + \varepsilon)$ approximation ratio. To choose which edges to query we rely on a parameter c_e defined for any edge $e \in E$ as follows:

$$c_e = \Pr[e \text{ has an end-point in OPT}],$$

where $\text{OPT} = \text{MVC}(\mathcal{G})$. In this definition, the randomization comes from the realization of $\mathcal{G}$ and MVC(.) is a fixed algorithm which returns the minimum vertex cover of a given graph. Basically, c_e is the probability with which an edge $e \in E$ is covered by $\text{MVC}(\mathcal{G})$.[1]

[1] Since we are not concerned with computational efficiency, we assume c_es are known. Regardless, similar to [14], our results can also be obtained with the estimated value of c_es calculated using polynomially many calls to the MVC oracle.

Given a parameter $\tau \in (0,1)$ whose value we fix later, we set Q, the subgraph that we query, to be the subgraph of edges with $c_e \leq 1-\tau$. I.e.,

$$Q = \{e \in E : c_e \leq 1-\tau\}.$$

Since we need the output to be a valid vertex cover, our final solution also needs to cover all the edges that we do not query. Let us denote this subgraph by $H = G - Q$. Therefore, after we query Q and see its realization $\mathcal{Q}$, the most straightforward way of finding a small vertex cover is to output $\mathrm{MVC}(\mathcal{Q} \cup H)$. However, for the sake of analysis, we need to design an alternative algorithm for finding a vertex cover of $\mathcal{Q} \cup H$. We formally explain this in Algorithm 1 and briefly explain the overall ideas below.

After querying Q and observing its realization $\mathcal{Q}$, we also hallucinate the edges in H to get $\mathcal{H}$. The hallucinated subgraph $\mathcal{H}$ is a random realization of H containing each edge independently w.p. p_e. We then let $P_1 = \mathrm{MVC}(\mathcal{Q} \cup \mathcal{H})$ and commit all its vertices to the final solution. Next, we pick another set of vertices, P_2, to cover all the edges in H that were not covered by P_1. That is, $P_2 = \mathrm{MVC}(H[V - P_1])$. We finally output $P_1 \cup P_2$. This clearly covers both $\mathcal{Q}$ and H and is a valid solution. The extra step of hallucinating H is to ensure that P_1 has the same distribution as the optimal solution (see Claim 8). This gives us two properties:

1. The expected size of P_1 is the same as OPT. To analyze our approximation ratio, it suffices to upper-bound $\mathbf{E}[|P_2|]$.
2. For any edge $e \in H$, we have $\Pr[e \text{ not covered by } P_1] = 1 - c_e$. Since by our choice of Q, the edges in $H = G - Q$ all have $c_e \geq 1-\tau$. This implies that any edge in H remains in $H[V - P_1]$ w.p. at most τ.

Having this, we argue that if the expected size of the solution $P_1 \cup P_2$ is more than $\mathbf{E}[\mathrm{OPT}] + \varepsilon n$, then we have $|P_2| \geq \varepsilon n$. This means that the subgraph $R = H[V - P_1]$ has a large minimum vertex cover and, consequently, a large maximum matching. Now, suppose we draw T independent subgraphs of H from the same distribution as R and call them $R_1, \ldots, R_T$. (We can do so by running our algorithm on T different hallucinations of G (see Definition 4).)

From each R_i, we take the maximum matching of it to be M_i, and from M_i, we delete all the edges present in at least one other R_j for $j \neq i$. We show that this gives us a set of induced matchings. Notice that since we have that any edge is present in R_i with probability at most τ, we can argue that deleting the edges present in other R_j's will not reduce the size of M_i significantly for a choice of $T = \frac{1}{2\tau}$. Therefore, we conclude that if $\mathbf{E}[|P_2|]$ is greater than εn, we can find a dense RS-graph in H. Finally, by carefully choosing $\tau = 1/3t$, where t is $\mathrm{RS}_n(\varepsilon n/8)$, we argue that Algorithm 1 has an additive approximation of at most εn.

In Section 3, we explain how we can modify this algorithm and design our $(1+\varepsilon)$ algorithm Algorithm 3. The main difference is that we pre-commit a subset of vertices to the final solution at the beginning of the algorithm. These vertices are the ones that have a high chance of being in OPT; therefore, including them does not significantly increase the size of the output. This allows us to argue that the optimal solution on H is large enough to allow us to turn the additive error into a multiplicative error.

1.2 Preliminaries

Throughout the paper, we denote the input graph as $G = (V, E)$, where $|V| = n$. From this input graph, we have each edge e present in $\mathcal{G}$ independently with probability p_e. We denote the minimum value among all p_es to be p. Given a small constant $\varepsilon > 0$, our goal is to query a sparse subgraph of G represented by Q non-adaptively and find a $(1+\varepsilon)$ approximate minimum vertex cover of $\mathcal{G}$.

We let MVC() be a fixed algorithm which given any graph outputs its minimum vertex cover. We may also refer to this as the minimum vertex cover oracle. We define $\mathrm{OPT} = \mathrm{MVC}(\mathcal{G})$ to be the optimal solution to our problem. Note that OPT is a random variable since $\mathcal{G}$ is a random realization of G. We also let $\textsc{opt} = \mathbf{E}[|\mathrm{OPT}|]$ be the expected size of this optimal solution. We let $\mu(G)$ be the size of the maximum matching of the graph G.

For any vertex $v \in V$, we define

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

which is the probability that v joins the optimal solution. This means $\sum_{v\in V} c_v = \textsc{opt}$. Similarly for any edge $e = (u, v)$ in the graph G we define

$$c_e = \Pr[u \in \mathrm{OPT} \text{ or } v \in \mathrm{OPT}].$$

▶ **Definition 4** (Graph Hallucination). *Given a any subgraph L of G we define a hallucination of L to be a subgraph $\mathcal{L}$ which contains each edge $e \in L$ independently with probability p_e.*

We define $\mathrm{RS}(n)$ to be the largest number of induced edge-disjoint matchings of size $\Theta(n)$ in an n-vertex graph.

2 The algorithm

In this section, we prove the following theorem.

▶ **Theorem 5.** *Let t be $RS_n(\varepsilon n/8)$. Running Algorithm 1 with $\tau = \frac{1}{3t}$ finds a vertex cover that has an additive approximation of at most εn to the optimal minimum vertex cover and queries a subgraph of size $O(tn/p)$.*

To prove this theorem, we design Algorithm 1, which takes a parameter τ, and queries the subgraph Q with $c_e \le 1 - \tau$. Let us call the edges realized in Q to be $\mathcal{Q}$. We then hallucinate the edges not in Q which we call the subgraph $H = G - Q$ to get $\mathcal{H}$. We then let P_1 be $\mathrm{MVC}(\mathcal{Q} \cup \mathcal{H})$.

This makes P_1 have the same distribution of OPT (see Claim 8 for the proof). Now we commit the vertices in P_1 to the final solution and all that remains to be covered are edges in H that have not been covered so far. Therefore we add $P_2 = \mathrm{MVC}(H[V - P_1])$ to the solution. All we need to do is to argue $\mathbf{E}[|P_2|]$ is small since we have $\mathbf{E}[|P_1|] = \textsc{opt}$.

We show that if $\mathbf{E}[|P_2|]$ is large we can find a large RS-graph in H (see Algorithm 2) so by carefully choosing τ which we relate to the density of RS-graphs we prove $\mathbf{E}[|P_2|]$ is small and hence we prove Algorithm 1 has at most additive error of εn. In the next section we modify Algorithm 1 slightly to get Algorithm 3 and show that this algorithm finds a $(1 + \varepsilon)$ approximation to the minimum vertex cover.

2.1 A Two-Step Algorithm

Here we describe the algorithm for finding the minimum vertex cover of G. We call it the two step algorithm because the output consists of union of two sets P_1 and P_2.

First, let us prove that the output of Algorithm 1 is a vertex cover.

▷ Claim 6. The output of Algorithm 1 is a vertex cover for $\mathcal{G}$.

Proof. Since P_2 is a vertex cover on the graph that P_1 does not cover we can argue that $P_1 \cup P_2$ is a vertex cover for $\mathcal{G}$. ◁

Algorithm 1 An algorithm to find MVC for $\mathcal{G}$.

```
Parameter: $\tau$
1. Let $Q$ be the subgraph of edges with $c_e \leq 1-\tau$.
2. Let $H$ be $G - Q$.
3. Query edges in $Q$ to get its realization $\mathcal{Q}$.
4. Hallucinate edges in $H$ to form $\mathcal{H}$.
5. Let $P_1 = \text{MVC}(\mathcal{Q} \cup \mathcal{H})$.
6. Let $R = H[V \setminus P_1]$.
7. Let $P_2 = \text{MVC}(R)$.
8. Return $O = P_1 \cup P_2$.
```

Now let us bound the number of queries that Algorithm 1 makes.

▷ Claim 7. The size of subgraph Q that we query is at most $O(\frac{n}{p\tau})$ for $p = \min(p_e)$ for $e \in E$.

Proof. Due to [14] we know that if we take a random minimum vertex cover P for $\mathcal{G}$ at most $O(n/p)$ edges will not be covered by it with high probability. Here we repeat how this argument works for convenience. Let us call this number of edges X and call this subgraph F. For any subgraph with more than n/p edges, the probability of none of its edges being in $\mathcal{G}$ is at most $(1-p)^{n/p}$. Moreover, since F is an induced subgraph of G, there are at most 2^n possibilities for it. By a union bound, we get that with high probability, F has at most n/p edges. That is:

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

Due to linearity of expectation we have

$$E[X] = \sum_e 1 - c_e.$$

For the edges in Q we have $1 - c_e \geq \tau$ so we can get $|E(Q)| \leq \frac{X}{\tau}$ and since we have $X = O(n/p)$ we get that $|E(Q)| \leq O(\frac{n}{p\tau})$. ◁

▷ Claim 8. The expected size of P_1 from Algorithm 1 is equal to OPT and the distribution of P_1 is the same as OPT

Proof. Look at the graph that P_1 is a minimum vertex cover for. It is the graph $\mathcal{Q} \cup \mathcal{H}$. For any edge e, if it is in H then it is present in $\mathcal{H}$ with probability p_e and if it is in Q it is present in $\mathcal{Q}$ with probability p_e. Since $Q \cup H = G$ we can see that the distribution of $\mathcal{Q} \cup \mathcal{H}$ is the same as the distribution for $\mathcal{G}$ and hence the expected size of the minimum vertex cover for $\mathcal{Q} \cup \mathcal{H}$ is the same as $\mathcal{G}$ which is OPT. Also since the distribution of $\mathcal{Q} \cup \mathcal{H}$ is the same as $\mathcal{G}$ we get

$$P_1 = \text{MVC}(\mathcal{Q} \cup \mathcal{H}) = \text{MVC}(\mathcal{G}) = \text{OPT}$$

Hence P_1 comes from the same distribution as OPT. ◁

▶ **Lemma 9.** *If Algorithm 1 does not have additive approximation of at most k in expectation, then* $\mathbf{E}[\mu(R)] \geq \frac{k}{2}$

Proof. Due to Claim 8, the expected size of P_1 is OPT. Therefore, if Algorithm 1 has worse than k additive approximation then we can conclude $\mathbf{E}[|P_2|] \geq k$. In any graph the maximum matching size it at least half of the size of minimum vertex cover. Applying this statement on the graph R give us that

$$\mathbf{E}[\mu(R)] \geq \frac{E[|P_2|]}{2} \geq \frac{k}{2}.$$ ◀

2.2 Finding a Dense RS-graph in H

Suppose in Algorithm 1 instead of querying Q and hallucinating H we hallucinate all the edges. We repeat this process T times, calling the resulting graph R in Algorithm 1 R_i for each i from 1 to T. For each R_i, we find the maximum matching M_i. Then, from each M_i, we delete the edges that are present in at least one other R_j for $j \neq i$, and we define S_i to be the remaining edges of M_i after these deletions. In Algorithm 2, we formalize this process. Our goal in this section is to argue that if Algorithm 1 does not have an additive error of at most εn, then the matchings S_i will be large, and we can find a large RS-graph as a subgraph of H.

Algorithm 2 Finding a large RS-graph in H in Algorithm 1.

```
Parameter: $\tau$
1. For $i$ from 1 to $T$:
   a. Let $Q$ and $H$ be graphs defined in Algorithm 1
   b. Hallucinate graph $G$ to get $\mathcal{G}_i$
   c. Let $P_i$ be MVC($\mathcal{G}_i$)
   d. Let $R_i = H[V \setminus P_i]$
2. Let $M_i$ be the maximum matching of $R_i$
3. Delete the edges in $M_i$ present in at least one other $R_j$ with $j \neq i$ to get $S_i$
4. Return $S_1 \cup S_2 \ldots \cup S_T$
```

The next claim will bound the probability of edges being in R_i.

▷ Claim 10. For each edge e and each graph R_i, we have that $\Pr[e \in R_i] \leq \tau$.

Proof. For the edge e we have two cases, $e \in Q$ or $e \in H$. If we have $e \in Q$ and $e \notin \mathcal{Q}$, then this edge will not be present in R, otherwise if $e \in \mathcal{Q}$ since P_1 covers all the edges of $\mathcal{Q}$ this edge will not be present in R. So the only case that $e \in R$ happens is when we have $e \in H$. In this case since we have $c_e \geq 1 - \tau$ and P_1 is a random minimum vertex cover of G, then the edge e will be covered with probability c_e in P_1 (this is because P_1 comes from the same distribution as OPT according to Claim 8). So this edge is present in P_2 with a probability of at most $1 - c_e \leq \tau$. ◁

The following claim will prove that the matching S_i are induced matchings.

▷ Claim 11. The matchings S_i formed in the random process are induced matchings.

Proof. The graphs R_i are induced subgraphs on $V - P_1$. Suppose that S_i's are not induced matchings. Since we delete duplicate edges from R_i to get S_i there is no edge e that is present in S_i and S_j for $i \neq j$. Hence, there must exist two edges $e_1 = (u_1, v_1)$ and $e_2 = (u_2, v_2)$ present in S_i and $e_3 = (u_1, u_2)$ present in S_j for $i \neq j$ to break the property of induced matchings. We call edges like e_3 a *cross edge*. In this case, we can argue that the edge e_3 is also present in R_i since both of its endpoints are in the vertices of R_i. Since e_3 is present in R_i and R_j this edge gets deleted and can not be part of S_j. This contradicts with what we assumed. Therefore we can conclude that S_i's are induced matchings. ◁

▷ Claim 12. Suppose we set $T = \frac{1}{2\tau}$, then we have $\mathbf{E}[|S_i|] \geq \frac{k}{4}$.

Proof. Due to Lemma 9, we have that $\mathbf{E}[|M_i|] \geq \frac{k}{2}$. Now we should argue that with the edge deletions, each M_i will lose at most $\frac{k}{4}$ edges in expectation. Take an edge e in M_i. For each $j \neq i$ we have $\Pr[e \in M_i] \leq \tau$ due to Claim 10 since for an edge to be in M_i it should also be in R_i. By an application of union bound the edge e is present in at least one other M_j with probability at most $T\tau$. Hence we get,

$$\Pr[\text{e not in any } M_j] \geq 1 - T\tau = \frac{1}{2}$$

And for the size of S_i we have:

$$\mathbf{E}[|S_i|] = \sum_{e \in M_i} \mathbb{1}_{e \notin M_j \text{ for } j \neq i} \geq \sum_{e \in M_i} \frac{1}{2} \geq \frac{k}{4} \qquad ◁$$

▶ **Lemma 13.** *By running the process explained above in Algorithm 2 we can find an $RS_n(r,t)$ graph (see Definition 2) with $t = \frac{1}{2\tau}$ and $r = \frac{k}{8}$ on H.*

Proof. Due to Claim 12 we can have $T = \frac{1}{2\tau}$ matchings each with expected size $\frac{k}{4}$. Putting these matchings together will give us expected $\frac{Tk}{4}$ edges in total. Suppose we break each matching S_i to smaller matchings of size $\frac{k}{8}$ size. If the number of edges in S_i is not divisible by $\frac{k}{8}$ we will lose at most the reminder of $|S_i|$ to $\frac{k}{8}$ which is at most $\frac{k}{8}$. The new matchings each have size exactly $\frac{k}{8}$ and are induced matchings. This is because the original S_i's did not have a cross edge (see Claim 11 for what a cross edge is) and hence the new matchings also do not have a cross edge since we are just partitioning the edges of S_i to smaller matchings. The total size of the new set of matchings is at least

$$\frac{Tk}{4} - \frac{Tk}{8} = \frac{Tk}{8}$$

This is because the total number of removed edges is bounded by the number of edges removed from each S_i which is $\frac{\varepsilon n}{8}$ times the number of S_i's which is T. Since we are running the random process which in expectation gives us induced matchings of size $\frac{k}{8}$ with total size of $\frac{Tk}{8}$ there is an instance which we have at least $\frac{Tk}{8}$ edges in total. From this we can construct an $RS_n(r,t)$ graph with $r = \frac{k}{8}$ and $t = \frac{1}{2\tau}$. ◀

Now we are ready to prove Theorem 5. We restate the theorem below.

▶ **Theorem 5.** *Let t be $RS_n(\varepsilon n/8)$. Running Algorithm 1 with $\tau = \frac{1}{3t}$ finds a vertex cover that has an additive approximation of at most εn to the optimal minimum vertex cover and queries a subgraph of size $O(tn/p)$.*

Proof. Suppose that Algorithm 1 does not find a vertex cover with additive approximation of εn. Then by setting $k = \varepsilon n$ in Lemma 13 we get

$$t' = \frac{1}{2\tau} = \frac{1}{2\frac{1}{3t}} = 1.5t$$

matchings each of size $\frac{\varepsilon n}{8}$ that are induced matchings. This is in contradiction with what we assumed since t is the maximum number of induced matchings of size $\frac{\varepsilon n}{8}$ for an RS-graph and now we have found $1.5t$ matchings of this size. This means that indeed Algorithm 1 has an additive approximation of at most εn for $\tau = \frac{1}{3t}$. Due to Claim 7, we have $|E(Q)| \leq O(\frac{n}{p\tau})$ which for $\tau = 1/3t$ we get $|E(Q)| = O(tn/p)$. ◀

3 Getting a Multiplicative Approximation

In this section, we modify Algorithm 1 to give us a multiplicative approximation of $1+\varepsilon$. Our modification is simple yet effective. At the beginning of the algorithm, we set $P_0 = \{v : c_v \geq 1-\alpha\varepsilon\}$ and commit P_0 to the final solution. The intuition behind this is that since these vertices have high c_v, they have a high probability of being in the final solution so it does not hurt us to have them in the final solution.

We do this specifically to be able to prove Lemma 16 which states c_vs for a subgraph of vertices is lower bounded by $\frac{\alpha\varepsilon}{2}$. This means that if we find an algorithm that is additive in terms of n' the size of this subgraph, then it is also a multiplicative approximation because of the lower bound for c_vs of this subgraph. Here is the modified algorithm.

Algorithm 3 Multiplicative Approximation Algorithm.

```
1. Let $P_0 = \{v : c_v \geq 1-\alpha\varepsilon\}$.
2. Let $G' = G[V \setminus P_0]$.
3. Let $Q$ be the subgraph of edges with $c_e \leq 1-\tau$ in $G'$.
4. Let $H = G' - Q$.
5. Query edges in $Q$ to get its realization $\mathcal{Q}$.
6. Hallucinate edges in $G \setminus Q$ to form subgraph $\mathcal{F}$.
7. Let $P_1 = \text{MVC}(\mathcal{Q} \cup \mathcal{F})$.
8. Let $R = H[V \setminus P_1]$.
9. Let $P_2 = \text{MVC}(R)$.
10. Return $O = P_0 \cup P_1 \cup P_2$.
```

The first lemma we prove is for bounding the expected size of $P_0 \cup P_1$.

▶ **Lemma 14.** $\mathbf{E}[|P_0 \cup P_1|] \leq (1+\varepsilon/2)\textsc{opt}$ *for* $\alpha \leq \frac{1}{4}$

Proof. We can write $E[|P_0 \cup P_1|]$ as follows:

$$E[|P_0 \cup P_1|] = |P_0| + \mathbf{E}[|P_1|] - \mathbf{E}[|P_0 \cap P_1|].$$

Due to Claim 8, we know that $\mathbf{E}[|P_1|] = \textsc{opt}$. Take an arbitrary vertex $v \in P_0$. Since P_1 is a random minimum vertex cover, any vertex v in P_0 has a probability of c_v of also being in P_1. And since $c_v \geq 1-\alpha\varepsilon$ for $v \in P_0$ we have $\Pr[v \in P_1] \geq 1-\alpha\varepsilon$. Hence we have

$$|P_0| - \mathbf{E}[|P_0 \cap P_1|] = \sum_{v \in P_0} 1 - \mathbb{1}_{v \in P_1} \leq |P_0| \cdot (\alpha\varepsilon) \tag{1}$$

We know that $|P_0| \leq \frac{\textsc{opt}}{1-\alpha\varepsilon}$ this is because $\textsc{opt} = \sum_{v \in V} c_v$ and we have $\sum_{v \in P_0} c_v \geq (1-\alpha\varepsilon)|P_0|$. Form Equation (1), we can get

$$|P_0| - \mathbf{E}[|P_0 \cap P_1|] \leq \frac{\textsc{opt}}{1-\alpha\varepsilon} \cdot (\alpha\varepsilon) \tag{2}$$

Since we have $\alpha \leq \frac{1}{4}$ and $\varepsilon \leq 1$ from Equation (2) we have

$$\begin{aligned} |P_0| - \mathbf{E}[|P_0 \cap P_1|] &\leq \frac{\textsc{opt}}{1-\alpha\varepsilon} \cdot (\alpha\varepsilon) \\ &\leq \frac{\textsc{opt}}{1-\alpha\varepsilon} \cdot (\frac{\varepsilon}{4}) \\ &\leq \frac{\textsc{opt}}{0.5} \cdot \frac{\varepsilon}{4} \\ &= \textsc{opt} \cdot \frac{\varepsilon}{2} \end{aligned}$$

Putting this together with $\mathbf{E}[|P_1|] = \textsc{opt}$ due to Claim 8 we get,

$$\mathbf{E}[|P_0 \cup P_1|] \leq (1+\varepsilon/2)\textsc{opt}$$ ◀

From Lemma 14 we can argue that if Algorithm 3 has worse than εOPT additive approximation in expectation, then $|P_2|$ is greater than $\frac{\varepsilon\text{OPT}}{2}$. We use this fact in the following lemma.

▶ **Lemma 15.** *If Algorithm 3 does not have a approximation of at most $(1+\varepsilon)$ in expectation, then $\mathbf{E}[\mu(R)] \geq \frac{\varepsilon OPT}{4}$*

Proof. Due to Lemma 14 we have $\mathbf{E}[|P_0 \cup P_1|] \leq (1+\varepsilon/2)\text{OPT}$ so if Algorithm 3 does not have a multiplicative approximation of at most $(1+\varepsilon)$ then we can conclude $\mathbf{E}[|P_2|] \geq \frac{\varepsilon\text{OPT}}{2}$. In any graph, the maximum matching size is at least half the minimum vertex cover size. Applying this statement on the graph R gives us,

$$\mathbf{E}[\mu(R)] \geq \frac{E[|P_2|]}{2} \geq \frac{\varepsilon\text{OPT}}{4}$$ ◀

The next lemma proves a lower bound for c_v of vertices in H having at least one edge. We call this graph H for future reference, meaning we remove the vertices in H having no edge.

▶ **Lemma 16.** *For any vertex $v \in H$, we have $c_v \geq \frac{\alpha\varepsilon}{2}$ for $\tau \leq \frac{\alpha\varepsilon}{2}$.*

Proof. Take an edge $e = (u, v) \in H$ since it has a degree of at least one. For this edge we know that $c_e \geq 1 - \tau$ since we put all the edges with $c_e \leq 1 - \tau$ in Q. Also we have $c_u < 1 - \alpha\varepsilon$ since $u \notin P_0$. We also know that $c_e \leq c_v + c_u$. Putting this all together we get that $c_v \geq \alpha\varepsilon - \tau$ which for a $\tau \leq \frac{\alpha\varepsilon}{2}$ we get $c_v \geq \frac{\alpha\varepsilon}{2}$. ◀

▷ Claim 17. We have $\text{OPT} \geq \frac{n'\alpha\varepsilon}{2}$ for $n' = |V(H)|$

Proof. This is because we have,

$$\text{OPT} = \sum_{v \in V} c_v \geq \sum_{v \in H'} c_v$$

and we know that $\sum_{v \in H'} c_v \geq n' \cdot \frac{\alpha\varepsilon}{2}$ due to Lemma 16. ◁

Putting Claim 17 and Lemma 15 together we get that if Algorithm 3 does not give a $(1+\varepsilon)$ approximation, then $\mathbf{E}[\mu(R)] \geq \frac{n'\alpha\varepsilon^2}{8}$. The next lemma asserts that if Algorithm 3 does not find a $(1+\varepsilon)$ approximation then we can find a large RS-graph in H.

▶ **Lemma 18.** *If Algorithm 3 is not a $(1+\varepsilon)$ approximation then, we can find an $RS_{n'}(r, t)$ graph (see Definition 2) with $t = 1/2\tau$ and $r = \frac{\alpha\varepsilon^2 n'}{32}$ on H*

Proof. Suppose we set $k = \frac{n'\alpha\varepsilon^2}{4}$. Then we can run the algorithm explained in Algorithm 2 and put together matchings S_i formed from different realizations of H to find an RS-graph where each matching size is $k/8$. Due to having $\mathbf{E}[\mu(R)] \geq \frac{n'\alpha\varepsilon^2}{8}$ we can prove Claim 10. This is because for edges in R_i we can argue that they exist in R_i with a probability of at most τ similar to how we proved it for R_i's in Algorithm 1 in Claim 10. Moreover, like how we proved for Algorithm 2 that the matchings S_i are induced matchings we form S_i's like in Algorithm 2 and we can argue that they are induced matchings. Then we can also prove Claim 12 and Lemma 13 for the new value of k. Therefore, similar to how we proved Lemma 13 we can find an $\text{RS}_{n'}(r, t)$ graph with $t = \frac{1}{2\tau}$ and $r = \frac{k}{8} = \frac{\alpha\varepsilon^2 n'}{32}$ on H. ◀

Now we are ready to prove the main result of this work.

▶ **Result 3.** *Given any $\varepsilon \in (0,1)$ let t be $RS_n(cn)$ for $c = \frac{\varepsilon^2}{128}$. There exists an algorithm that queries a subgraph of size $O(tn/p)$ and finds a vertex cover with the expected approximation ratio of $(1+\varepsilon)$.*

Proof. Let us run Algorithm 3 with $\alpha = 1/4$ and $\tau = 1/3t$. Meaning we have $c = \frac{\alpha\varepsilon^2}{32}$. We proceed similar to how we proved Theorem 5. Suppose that Algorithm 3 does not find a vertex cover with $(1+\varepsilon)$ approximation. Then we can find an $RS_{n'}(r,t)$ graph in H for $t = 1/2\tau$ and $r = \frac{\alpha\varepsilon^2 n'}{32}$ since $\mathrm{RS}_n(cn) \geq \mathrm{RS}_{n'}(cn')$ for $n \geq n'$. By setting $k = \frac{n'\alpha\varepsilon^2}{4}$ in Lemma 13 we get

$$t' = \frac{1}{2\tau} = \frac{1}{2\frac{1}{3t}} = 1.5t$$

matchings each of size $\frac{k}{8}$ that are induced matchings. This is in contradiction with what we assumed since t is the maximum number of induced matchings of size $\frac{k}{8}$ for an RS-graph and now we have found $1.5t$ matchings of this size. This means that indeed Algorithm 3 has an approximation of at most $(1+\varepsilon)$ for $\tau = \frac{1}{3t}$. Due to Claim 7, we have $|E(Q)| \leq O(\frac{n}{p\tau})$ which for $\tau = 1/3t$ we get $|E(Q)| = O(tn/p)$. ◀

Correlated Edges

Suppose that G has some edges that are realized in $\mathcal{G}$ independently called G_1 and some edges that are correlated with each other and the edges in G_1 called G_2. This means that edges in G_2 might exist based on edges in G_1 or G_2 that are realized. We need to adjust the definition of hallucination and realize the correlated edges by drawing them from the given distribution rather than realizing them independently. Moreover, if a set of them are queried and we want to hallucinate the rest, it should be done conditionally.

Notice that the only place we use independent realization of edges is in the proof of Claim 7. So we can argue that among the edges in G_1 at most $O(tn/p)$ edges will be in Q. Furthermore we know that $|Q| \leq |Q_1| + |Q_2|$ where Q_i is the subgraph of G_i that is in Q. Since the number of correlated edges is at most $\mathrm{RS}_n(cn)$ we have $|Q_2| \leq n \cdot \mathrm{RS}_n(cn)$ so we get:

$$|Q| \leq |Q_1| + |Q_2| \leq O(tn/p) + n \cdot \mathrm{RS}_n(cn) = O(tn/p) + n \cdot t = O(tn/p)$$

Based on the explanation above we drive the following corollary.

▶ **Corollary 19.** *Our Algorithm 3 can handle up to $t = RS_n(cn)$ correlated edges for $c = \frac{\varepsilon^2}{128}$ and still provide a $(1+\varepsilon)$ approximation to the optimal solution with $O(tn/p)$ queries.*

References

1 Sepehr Assadi. A two-pass (conditional) lower bound for semi-streaming maximum matching. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 708–742. SIAM, 2022. doi:10.1137/1.9781611977073.32.

2 Sepehr Assadi, Soheil Behnezhad, Sanjeev Khanna, and Huan Li. On regularity lemma and barriers in streaming and dynamic matching. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 131–144. ACM, 2023. doi:10.1145/3564246.3585110.

3 Sepehr Assadi and Aaron Bernstein. Towards a unified theory of sparsification for matching problems. In Jeremy T. Fineman and Michael Mitzenmacher, editors, *2nd Symposium on Simplicity in Algorithms, SOSA 2019, January 8-9, 2019, San Diego, CA, USA*, volume 69 of *OASIcs*, pages 11:1–11:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/OASICS.SOSA.2019.11.

4 Sepehr Assadi and Sanjeev Khanna. Improved bounds for fully dynamic matching via ordered ruzsa-szemerédi graphs. *CoRR*, abs/2406.13573, 2024. doi:10.48550/arXiv.2406.13573.

5 Sepehr Assadi, Sanjeev Khanna, and Yang Li. The stochastic matching problem: Beating half with a non-adaptive algorithm. In Constantinos Daskalakis, Moshe Babaioff, and Hervé Moulin, editors, *Proceedings of the 2017 ACM Conference on Economics and Computation, EC '17, Cambridge, MA, USA, June 26-30, 2017*, pages 99–116. ACM, 2017. doi:10.1145/3033274.3085146.

6 Sepehr Assadi, Sanjeev Khanna, and Yang Li. The stochastic matching problem with (very) few queries. *ACM Trans. Economics and Comput.*, 7(3):16:1–16:19, 2019. doi:10.1145/3355903.

7 Soheil Behnezhad, Avrim Blum, and Mahsa Derakhshan. Stochastic vertex cover with few queries. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 1808–1846. SIAM, 2022. doi:10.1137/1.9781611977073.73.

8 Soheil Behnezhad and Mahsa Derakhshan. Stochastic weighted matching: (stochastic weighted matching: (1-ε) approximation. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 1392–1403. IEEE, 2020. doi:10.1109/FOCS46700.2020.00131.

9 Soheil Behnezhad, Mahsa Derakhshan, Alireza Farhadi, MohammadTaghi Hajiaghayi, and Nima Reyhani. Stochastic matching on uniformly sparse graphs. In Dimitris Fotakis and Evangelos Markakis, editors, *Algorithmic Game Theory - 12th International Symposium, SAGT 2019, Athens, Greece, September 30 - October 3, 2019, Proceedings*, volume 11801 of *Lecture Notes in Computer Science*, pages 357–373. Springer, 2019. doi:10.1007/978-3-030-30473-7_24.

10 Soheil Behnezhad, Mahsa Derakhshan, and MohammadTaghi Hajiaghayi. Stochastic matching with few queries: (1-ε) approximation. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 1111–1124. ACM, 2020. doi:10.1145/3357713.3384340.

11 Soheil Behnezhad and Alma Ghafari. Fully dynamic matching and ordered ruzsa-szemerédi graphs. In *65st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2024*, 2024.

12 Soheil Behnezhad and Nima Reyhani. Almost optimal stochastic weighted matching with few queries. In Éva Tardos, Edith Elkind, and Rakesh Vohra, editors, *Proceedings of the 2018 ACM Conference on Economics and Computation, Ithaca, NY, USA, June 18-22, 2018*, pages 235–249. ACM, 2018. doi:10.1145/3219166.3219226.

13 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 Tim Roughgarden, Michal Feldman, and Michael Schwarz, editors, *Proceedings of the Sixteenth ACM Conference on Economics and Computation, EC '15, Portland, OR, USA, June 15-19, 2015*, pages 325–342. ACM, 2015. doi:10.1145/2764468.2764479.

14 Mahsa Derakhshan, Naveen Durvasula, and Nika Haghtalab. Stochastic minimum vertex cover in general graphs: A 3/2-approximation. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 242–253. ACM, 2023. doi:10.1145/3564246.3585230.

15 Mahsa Derakhshan and Mohammad Saneian. Query efficient weighted stochastic matching. *CoRR*, abs/2311.08513, 2023. doi:10.48550/arXiv.2311.08513.

16 Shaddin Dughmi, Yusuf Hakan Kalayci, and Neel Patel. On sparsification of stochastic packing problems. In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, July 10-14, 2023, Paderborn, Germany*, volume 261 of *LIPIcs*, pages 51:1–51:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.ICALP.2023.51.

17 Eldar Fischer, Eric Lehman, Ilan Newman, Sofya Raskhodnikova, Ronitt Rubinfeld, and Alex Samorodnitsky. Monotonicity testing over general poset domains. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 474–483, 2002. doi:10.1145/509907.509977.

18 Jacob Fox. A new proof of the graph removal lemma. *Annals of Mathematics*, pages 561–579, 2011.

19 Jacob Fox, Hao Huang, and Benny Sudakov. On graphs decomposable into induced matchings of linear sizes. *Bulletin of the London Mathematical Society*, 49(1):45–57, 2017.

20 Ashish Goel, Michael Kapralov, and Sanjeev Khanna. On the communication and streaming complexity of maximum bipartite matching. In Yuval Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 468–485. SIAM, 2012. doi:10.1137/1.9781611973099.41.

21 Michel X. Goemans and Jan Vondrák. Covering minimum spanning trees of random subgraphs. In J. Ian Munro, editor, *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004, New Orleans, Louisiana, USA, January 11-14, 2004*, pages 934–941. SIAM, 2004. URL: http://dl.acm.org/citation.cfm?id=982792.982933.

22 Michel X. Goemans and Jan Vondrák. Covering minimum spanning trees of random subgraphs. *Random Struct. Algorithms*, 29(3):257–276, 2006. doi:10.1002/RSA.20115.

23 Imre Z Ruzsa and Endre Szemerédi. Triple systems with no six points carrying three triangles. *Combinatorics (Keszthely, 1976), Coll. Math. Soc. J. Bolyai*, 18(939-945):2, 1978.

24 Jan Vondrák. Shortest-path metric approximation for random subgraphs. *Random Struct. Algorithms*, 30(1-2):95–104, 2007. doi:10.1002/RSA.20150.

25 Yutaro Yamaguchi and Takanori Maehara. Stochastic packing integer programs with few queries. In Artur Czumaj, editor, *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. SIAM, 2018. doi:10.1137/1.9781611975031.21.