A $(2+\varepsilon)$-Factor Approximation Algorithm for Split Vertex Deletion

Daniel Lokshtanov
University of California, Santa Barbara, CA, USA
daniello@ucsb.edu

Pranabendu Misra
Max Planck Institut für Informatik, Saarland Informatics Campus, Saarbrücken, Germany
pmisra@mpi-inf.mpg.de

Fahad Panolan
IIT Hyderabad, India
fahad@iith.ac.in

Geevarghese Philip
Chennai Mathematical Institute, UMI ReLaX, Chennai, India
gphilip@cmi.ac.in

Saket Saurabh
Institute of Mathematical Sciences, Chennai, India
University of Bergen, Norway
saket@imsc.res.in

Abstract

In the Split Vertex Deletion (SVD) problem, the input is an n-vertex undirected graph G and a weight function $w\colon V(G) \to \mathbb{N}$, and the objective is to find a minimum weight subset S of vertices such that $G - S$ is a split graph (i.e., there is bipartition of $V(G-S) = C \uplus I$ such that C is a clique and I is an independent set in $G - S$). This problem is a special case of 5-Hitting Set and consequently, there is a simple factor 5-approximation algorithm for this. On the negative side, it is easy to show that the problem does not admit a polynomial time $(2-\delta)$-approximation algorithm, for any fixed $\delta > 0$, unless the Unique Games Conjecture fails.

We start by giving a simple quasipolynomial time ($n^{\mathcal{O}(\log n)}$) factor 2-approximation algorithm for SVD using the notion of *clique-independent set separating collection*. Thus, on the one hand SVD admits a factor 2-approximation in quasipolynomial time, and on the other hand this approximation factor cannot be improved assuming UGC. It naturally leads to the following question: Can SVD be 2-approximated in polynomial time? In this work we almost close this gap and prove that for any $\varepsilon > 0$, there is a $n^{\mathcal{O}(\log \frac{1}{\varepsilon})}$-time $2(1+\varepsilon)$-approximation algorithm.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation → Approximation algorithms analysis

Keywords and phrases Approximation Algorithms, Graph Algorithms, Split Vertex Deletion

Digital Object Identifier 10.4230/LIPIcs.ICALP.2020.80

Category Track A: Algorithms, Complexity and Games

Funding This work has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (via grant no. 819416 and no. 715744), the Norwegian Research Council(via grants MULTIVAL and CLASSIS), and Swarnajayanti Fellowship grant DST/SJF/MSA-01/2017-18.

1 Introduction

The Hitting Set problem encompasses a large number of well studied problems in Computer Science. Here, the input is a family $\mathcal{F}$ of sets over an n-element universe U and a weight function $w \colon U \to \mathbb{N}$, and the objective is to compute a hitting set of minimum weight. A *hitting set* is a subset $S \subseteq U$ such that for any $F \in \mathcal{F}$, $F \cap S \neq \emptyset$ and the weight of S is $w(S) = \sum_{u \in S} w(u)$. This problem generalizes a number of other well studied problems in computer science, and consequently it is very hard to approximate: it can not be approximated within a factor $2^{\log^{1-\delta_c(n)} n}$ in polynomial time, for any constant $c < 1/2$, unless SAT can be decided in slightly subexponential time, where $\delta_c(n) = 1/(\log\log n)^c$ [11]. A restricted version of this problem, is the d-Hitting Set problem, where $d \in \mathbb{N}$ and the cardinality of every set in $\mathcal{F}$ is at most d. This problem also generalizes a number of well studied problems, and it admits a simple factor d-approximation algorithm: Solve the natural LP relaxation and select all elements whose corresponding variable in the LP is set to at least $1/d$. Unfortunately, this simple algorithm is likely to be the best possible. That is, assuming Unique Game Conjecture (UGC), there is no c-factor approximation algorithm for d-Hitting Set, for any $c < d$ in the general case [7].

A number of vertex deletion problems on graphs can be considered as special cases of d-Hitting Set, and it is of great interest to devise factor-α approximation algorithm for them where $\alpha < d$, or rule out any such algorithm. For example, in the Vertex Cover problem, the input is a graph G and a weight function $w \colon V(G) \to \mathbb{N}$, and the objective is to find a subset of vertices of minimum weight that hits all edges in G. This is same as 2-Hitting Set, and assuming the Unique Games Conjecture we cannot do better than a factor-2 approximation in polynomial time. However, there are other examples of vertex deletion problems on graphs, that are special cases of d-Hitting Set, for which we can indeed do better than a factor-d approximation. Consider the Cluster Vertex Deletion problem, where the input is a graph G and a weight function $w \colon V(G) \to \mathbb{N}$, and the objective is to find a minimum weight subset S of vertices such that S is a cluster graph. Equivalently, S hits all induced paths of length 3 in G. Hence, it is a special case of 3-Hitting Set and admits a simple 3-approximation algorithm. You et al. [13] showed that the unweighted version of Cluster Vertex Deletion admits a 5/2 approximation algorithm. Recently, this was improved to factor 9/4 by Fiorini et al. [5]. The problem also admits an approximation-preserving reduction from Vertex Cover and hence there is a lower bound of 2 on the approximation-factor assuming UGC [5]. Fiorini et al. [5] have conjectured that Cluster Vertex Deletion admits a 2-approximation algorithm. Another example is the Tournament Feedback Vertex Set (TFVS) problem, which is equivalent to hitting all directed triangles in a digraph. It is very well studied in the realm of approximation algorithms [3, 1, 10, 9], and very recently a 2-approximation algorithm was designed by Lokshtanov et al. [9], matching the lower-bound under UGC [12]. Similarly, a number of such "implicit" d-Hitting Set problems are studied in Computer Science, and it is of great interest to settle their approximation complexity.

In this work we study another implicit d-Hitting Set problem called Split Vertex Deletion(SVD) (defined below). A subset S of vertices in a graph G is a split vertex deletion set if $G - S$ is a split graph (i.e., there is bipartition of $V(G - S) = C \uplus I$ such that C is a clique and I is an independent set in $G - S$).

Split Vertex Deletion (SVD)
Input: An undirected graph G and a weight function $w : V(G) \to \mathbb{N}$.
Output: A split vertex deletion set $S \subseteq V(G)$ of G of the smallest weight (an *optimum* split vertex deletion set of G).

A graph G is a split graph if and only if it does not contain C_4, C_5 and $2K_2$ as induced subgraphs in G [6]. This implies that SVD is special case of 5-Hitting Set and hence it admits a simple 5-approximation algorithm. Furthermore, it is interesting to note that we can obtain a 2-approximation algorithm for SVD in time $n^{O(\log n)}$ using the notion of *clique-independent set separating collection* [4]. For a graph G, a clique-independent set separating collection is a family $\mathcal{C}$ of vertex subsets of $V(G)$ such that for a clique C and an independent set I in G such that $C \cap I = \emptyset$, there is subset X in the collection $\mathcal{C}$ such that $C \subseteq X$ and $I \subseteq V(G) \setminus X$. Thus, if there is a "small" clique-independent set separating collection, then we can enumerate such a collection $\mathcal{C}$ and solve Vertex Cover of $\overline{G}[X]$ and $G - X$ for each $X \in \mathcal{C}$. Notice that for any $X \in \mathcal{C}$, the union of the two solutions of the two Vertex Cover instances on $\overline{G}[X]$ and $G - X$, respectively, is a solution to SVD. Moreover, the best c-approximation solutions over all choices of X, is a c-approximate solution of SVD. It is known that for any n-vertex graph, there is clique-independent set separating collection of size $n^{\mathcal{O}(\log n)}$ and this can be enumerated in time linear in the size of the collection [4]. This along with a 2-approximation algorithm of Vertex Cover leads to an $n^{\mathcal{O}(\log n)}$-time 2-approximation algorithm for SVD. There is also a simple approximation preserving reduction from Vertex Cover to SVD, which shows that we cannot improve upon factor 2-approximation algorithm, unless UGC fails. The reduction is as follows: Given an instance (G, w) of Vertex Cover, we add a large complete graph H of size $2|V(G)|$ into G with weight of each vertex in H to be $\max\{w(u) : u \in V(G)\}$. One can easily verify that this is an approximation preserving reduction.

Thus, on the one hand SVD admits a 2-approximation in quasipolynomial ($n^{\mathcal{O}(\log n)}$) time, and on the other hand this approximation factor cannot be improved assuming UGC. It naturally leads to the following question: Can SVD be 2-approximated in polynomial time? This is precisely the question we address in this paper, and obtain the following result.

▶ **Theorem 1.** *Let G be a graph on n vertices, w a weight function on $V(G)$ and let $\varepsilon > 0$ be a constant. Then there exists a randomized algorithm that runs in time $\mathcal{O}(n^{g(\varepsilon)})$ and outputs $S \subseteq V(G)$ such that $G - S$ is a split graph and $w(S) \leq 2(1+\varepsilon)w(OPT)$ with probability at least $1/2$. Here OPT is a minimum weight split vertex deletion set of G, and $g(\varepsilon) = 6 + 8\log(80(1 + \frac{12}{\varepsilon})) \cdot \log(\frac{30}{\varepsilon}) / \log(4/3)$.*

Overview of Theorem 1. At a very high level the algorithm described in Theorem 1 is inspired from the algorithm developed for factor 2-approximation algorithm for TFVS [9]. In TFVS knowing just one vertex is sufficient to completely split the instance into two independent sub-instances and thus leading to a natural divide and conquer scheme. However, in our case (SVD) the instances don't become truly independent before every vertex is classified as either *potential clique* or *potential independent set vertex.* Classifying all the vertices requires several new ideas and insights in the problem. This classification could be vaguely viewed as a polynomial time algorithm that quickly navigates through sets in clique-independent set separating collection $\mathcal{C}$, and almost reaches a correct partition.

Our algorithm in fact finds a $(2+\varepsilon)$-factor approximate solution for a more general *annotated* variant of the problem, where the solution must obey certain additional constraints.

Annotated Split Vertex Deletion (A-SVD)
Input: An undirected graph G, a weight function $w : V(G) \to \mathbb{N}$, and a partition of $V(G)$ into three parts $V(G) = C \uplus I \uplus U$, where at most two of these parts may be empty.
Output: A set $S^\star \subseteq V(G)$ of G of the smallest weight such that $G - S^\star$ is a split graph with a split partition $(C^\star, I^\star)$ where $C^\star \subseteq (C \cup U)$ and $I^\star \subseteq (I \cup U)$ hold.

A *feasible solution* to an instance $(G, w, (C, I, U))$ of Annotated Split Vertex Deletion is a split vertex deletion set S of G such that the split graph $G - S$ has a split partition (C', I') where no vertex in the specified set I goes to the split part C' and no vertex in the specified set C goes to the independent part I'. Thus, each vertex in the set I is either deleted as part of S or ends up in the independent set I' in graph $G - S$, and each vertex in C is either deleted or ends up in the clique C' in $G - S$. There are no restrictions on where the vertices in the "unconstrained" set U may go. We call a feasible solution of A-SVD an *annotated split vertex deletion set* of the instance $(G, w, (C, I, U))$; the A-SVD problem asks for an *optimum* annotated split vertex deletion set of the input instance.

First we show that we can, in polynomial time, find 2-factor approximate solutions to A-SVD instances which are of the form $(G, w, (C, I, U = \emptyset))$ (Lemma 12). Let $(G, w, (C, I, U))$ be an instance of A-SVD, let OPT be an (unknown) optimum solution to $(G, w, (C, I, U))$, let $(C^\star \subseteq (C \cup U), I^\star \subseteq (I \cup U))$ be a split partition of $G - OPT$, and let $C^\star_U = (C^\star \cap U), I^\star_U = (I^\star \cap U)$. We show that if $w(C^\star_U \setminus \{c^\star\}) \leq \frac{\varepsilon \cdot w(OPT)}{2}$ holds for some $c^\star \in C^\star_U$ or $w(I^\star_U \setminus \{i^\star\}) \leq \frac{\varepsilon \cdot w(OPT)}{2}$ holds for some $i^\star \in I^\star_U$ then we can, in polynomial time, find a $(2+\varepsilon)$-factor approximate solution to $(G, w, (C, I, U))$ (Lemma 16, Lemma 18). These constitute the base cases of our algorithm. It is not difficult to see that moving a vertex $x \in C^\star_U$ to the set C and moving a vertex $y \in I^\star_U$ to the set I are approximation-preserving transformations. At a high level, our algorithm starts with an arbitrary instance $(G, w, (C, I, U))$ of A-SVD, correctly identifies – with a constant probability of success – a good fraction of vertices which belong to the sets $C^\star_U$ or $I^\star_U$, and moves these vertices to the sets C or I, respectively. It then recurses on the resulting instance, till it reaches one of the base cases described above.

We now briefly and informally outline how our algorithm identifies vertices as belonging to $C^\star_U$ or $I^\star_U$. Consider the bipartite subgraph H of G induced by the pair $(C^\star_U, I^\star_U)$. Define the weight of an edge to be the product of the weights of its two end-points, and suppose the total weight of edges in H is at least half the maximum possible weight. Then each of a constant fraction (by weight) of the vertices in $I^\star_U$ has a constant fraction (by weight) of $C^\star_U$ in its neighborhood (Lemma 4). If we can identify one of these special vertices of $I^\star_U$ then we can safely move all its neighbors in U to the set C while reducing the weight of $C^\star_U$ by a constant fraction. The catch, of course, is that we have no idea what the set $I^\star_U$ is.

To get around this, we find an approximate solution X of the Split Vertex Deletion instance defined by the induced subgraph $G[U]$. Let (C_X, I_X) be a split partition of $G - X$. We show that we can, in polynomial time and with constant probability, sample a vertex from the set $X \cup (I_X \setminus C^\star_U)$ (Lemma 26). We further show that the weight of $X \cup (I_X \setminus C^\star_U)$ is at most a *constant multiple* of the weight of $I^\star_U$ (Lemma 22). So if $I^\star_U \subseteq (X \cup (I_X \setminus C^\star_U))$ holds then we can, with good probability, sample a vertex from the set $I^\star_U$. The hard part is when this condition does not hold. We show using a series of lemmas (summarized in Lemma 25) that we can, even in this case, sample a vertex from one of the two sets $C^\star_U, I^\star_U$ with constant probability. A symmetric analysis applies when the total weight of *non-edges* across $(C^\star_U, I^\star_U)$ is at least half the maximum possible weight.

2 Preliminaries

We use $\uplus$ to denote the disjoint union of sets. Moreover, when we write $X \uplus Y$ we implicitly assert that the sets X and Y are disjoint. We use $V(G)$ (respectively, $E(G)$) to denote the vertex set (respectively, the edge set) of graph G. For a subset $S \subseteq V(G)$ of vertices of G we use $G[S]$ to denote the subgraph of G *induced* by S and $G - S$ to denote the subgraph of G obtained by deleting all vertices in S (and their incident edges) from G. A *non-edge* in a graph G is any 2-subset $\{x, y\} \subseteq V(G)$ of vertices such that xy is not an edge in G. For the sake of brevity we use the notation xy to denote a non-edge $\{x, y\}$. For a finite set U, weight function $w : U \to \mathbb{N}$, and subset $X \subseteq U$ we use w_X to denote the weight function w *restricted to the subset* X, and $w(X)$ to denote the sum $\sum_{x \in X} w(x)$ of weights of all the elements in X. For the sake of brevity we drop the subscript X from the expression w_X when there is no risk of ambiguity.

The operation of *sampling (or picking) proportionately at random* from U according to the weight function w chooses one element from U, where each element $x \in U$ is chosen with probability $w(x)/w(U)$. We use $\overline{G}$ to denote the *complement* of a graph G, defined as follows: The vertex set of $\overline{G}$ is $V(G)$. For every two vertices $\{u, v\} \subseteq V(G)$ there is an edge uv in $\overline{G}$ if and only if uv is *not* an edge in graph G. A *vertex cover* of graph G is any subset $S \subseteq V(G)$ of its vertex set such that the graph $G - S$ has no edges. A *clique* in graph G is any non-empty subset $S \subseteq V(G)$ of its vertex set such that (i) $|S| = 1$, or (ii) if $|S| \geq 2$ then for every two vertices u, v in S, the edge uv is present in graph G.

▶ **Observation 2.** *For an undirected graph G and any $S \subseteq V(G)$, the vertex set $V(G) \setminus S$ is a clique in G if and only if S is a vertex cover of the complement graph $\overline{G}$.*

For a graph G and two disjoint vertex subsets $X, Y \subseteq V(G)$; $X \cap Y = \emptyset$ *the bipartite subgraph of G induced by the pair* (X, Y) has vertex set $X \cup Y$ and edge set $\{xy \mid x \in X, y \in Y, xy \in E(G)\}$. Note that the bipartite subgraph of G induced by the pair (X, Y) is not necessarily identical to the subgraph $G[X \cup Y]$ induced by the subset $X \cup Y$, and is defined even if the induced subgraph $G[X \cup Y]$ is not bipartite. For a bipartite graph H with vertex bipartition $V(H) = V_1 \uplus V_2$ we define $\widehat{E(H)} = \{v_1v_2 \mid v_1 \in V_1, v_2 \in V_2, v_1v_2 \notin E\}$ to be the set of all **non-edges** of H *with one end in* V_1 *and the other end in* V_2. Further, for a weight function $w : V(H) \to \mathbb{N}$ defined on the vertex set of a bipartite graph H we define the weight of its edge set to be $w(E(H)) = \sum_{v_1v_2 \in E(H)}(w(v_1) \cdot w(v_2))$ and the weight of its set of non-edges to be $w(\widehat{E(H)}) = \sum_{v_1v_2 \in \widehat{E(H)}}(w(v_1) \cdot w(v_2))$.

▶ **Definition 3.** *Let G be an undirected graph and $w : V(G) \to \mathbb{N}$ a weight function. Let X, Y be two disjoint vertex subsets of G and let H be the bipartite subgraph of G induced by the pair (X, Y). Let $w(E(H))$ and $w(\widehat{E(H)})$ be defined as above. We say that (X, Y) is a* heavy pair *if $w(E(H)) \geq \frac{w(X) \cdot w(Y)}{2}$ holds, and is a* light pair *if $w(\widehat{E(H)}) \geq \frac{w(X) \cdot w(Y)}{2}$ holds.*

▶ **Lemma 4 (♣).** [1] *Let $H = (V, E)$ be a bipartite graph, let $V = V_1 \uplus V_2$ be a bipartition of H, and let $w : V(H) \to \mathbb{N}$ be a weight function. Then (V_1, V_2) is either a heavy pair or a light pair. Moreover,*

1. *Suppose (V_1, V_2) is a heavy pair, and let $X = \{x \in V_1 \mid w(N(x)) \geq \frac{w(V_2)}{4}\}$ be the set of all vertices x in the set V_1 such that the total weight of the neighborhood of x in the set V_2 is at least one-fourth the total weight of the set V_2. Then $w(X) > \frac{w(V_1)}{4}$.*
2. *Suppose (V_1, V_2) is a light pair, and let $Y = \{y \in V_1 \mid w(V_2 \setminus N(y)) \geq \frac{w(V_2)}{4}\}$ be the set of all vertices y in the set V_1 such that the total weight of the* non-neighbors *of y in the set V_2 is at least one-fourth the total weight of the set V_2. Then $w(Y) > \frac{w(V_1)}{4}$.*

For a graph G given together with a weight function $w : V(G) \to \mathbb{N}$, an *optimum vertex cover* of G is any vertex cover of G with the least total weight.

[1] Proofs of statements labeled with a ♣ will appear in the full version of the paper.

Weighted Vertex Cover (wVC)
Input: An undirected graph G and a weight function $w : V(G) \to \mathbb{N}$.
Output: An optimum vertex cover $S \subseteq V(G)$ of G

▶ **Theorem 5** ([2]). *There is an algorithm which, given an instance (G, w) of Weighted Vertex Cover as input, runs in $\mathcal{O}(|E(G)|)$ time and outputs a vertex cover S of G whose weight is at most twice the weight of an optimum vertex cover of G.*

3 The Algorithm

Let (G, w) be an instance of Split Vertex Deletion. Since deleting vertices conserves the property of being a split graph one can safely add zero-weight vertices to any split vertex deletion set. So we assume without loss of generality that $w(v) \geq 1$ holds for every $v \in V(G)$. Split Vertex Deletion is NP-complete by the meta-result of Lewis and Yannakakis [8], and has a simple 5-factor approximation algorithm based on the Local Ratio Technique.

▶ **Theorem 6** (♣). *There is a deterministic algorithm which, given an instance (G, w) of SVD, runs in $\mathcal{O}(|V(G)|^6)$ time and outputs a split vertex deletion set $S \subseteq V(G)$ of G such that $w(S) \leq 5 \cdot w(OPT)$ where OPT is an optimum split vertex deletion set of G.*

We describe a randomized polynomial-time algorithm which outputs a $(2+\varepsilon)$-factor approximate solution for this problem for any fixed $\varepsilon > 0$.

Note that in an instance $(G, w, (C, I, U))$ of Annotated Split Vertex Deletion the set C is not necessarily a clique, nor is I necessarily an independent set in G. But we have the following.

▶ **Observation 7.** *Let S be a feasible solution of an A-SVD instance $(G, w, (C, I, U))$ and let (C', I') be a split partition of $G - S$ where $C' \subseteq (C \cup U)$ and $I' \subseteq (I \cup U)$ hold. Then $C \setminus S \subseteq C'$ and $I \setminus S \subseteq I'$ hold. Hence $C \setminus S$ is a clique in G and $I \setminus S$ is an independent set in G.*

From Observations 2 and 7 we get

▶ **Corollary 8.** *Let S be a feasible solution of an A-SVD instance $(G, w, (C, I, U))$. Let VC_C be an optimum solution of the wVC instance $(\overline{G[C]}, w)$ and let VC_I be an optimum solution of the wVC instance $(G[I], w)$. Then $w(VC_C) \leq w(S \cap C)$ and $w(VC_I) \leq w(S \cap I)$ hold.*

A-SVD is clearly a generalization of SVD: Given an instance (G, w) of SVD, construct the instance $(G, w, (C = \emptyset, I = \emptyset, U = V(G)))$ of A-SVD. Every split vertex deletion set of graph G is a feasible solution of the A-SVD instance, and every annotated split vertex deletion set of $(G, w, (\emptyset, \emptyset, V(G)))$ is a split vertex deletion set of graph G. It follows that for any constant c, a c-factor approximate solution to the A-SVD instance is a c-factor approximate solution to the SVD instance as well.

We can find feasible solutions to an A-SVD instance $(G, w, (C, I, U))$ by computing vertex covers for certain pairs of subgraphs derived from G.

▶ **Observation 9** (♣). *Let $(G, w, (C, I, U))$ be an instance of A-SVD.*

1. *Let V_1 be a vertex cover of the graph $G[I \uplus U]$ and let V_2 be a vertex cover of the graph $\overline{G[C]}$. Then $V_1 \uplus V_2$ is a feasible solution to $(G, w, (C, I, U))$.*
2. *Let V_3 be a vertex cover of the graph $G[I]$ and let V_4 be a vertex cover of the graph $\overline{G[C \uplus U]}$. Then $V_3 \uplus V_4$ is a feasible solution to $(G, w, (C, I, U))$.*

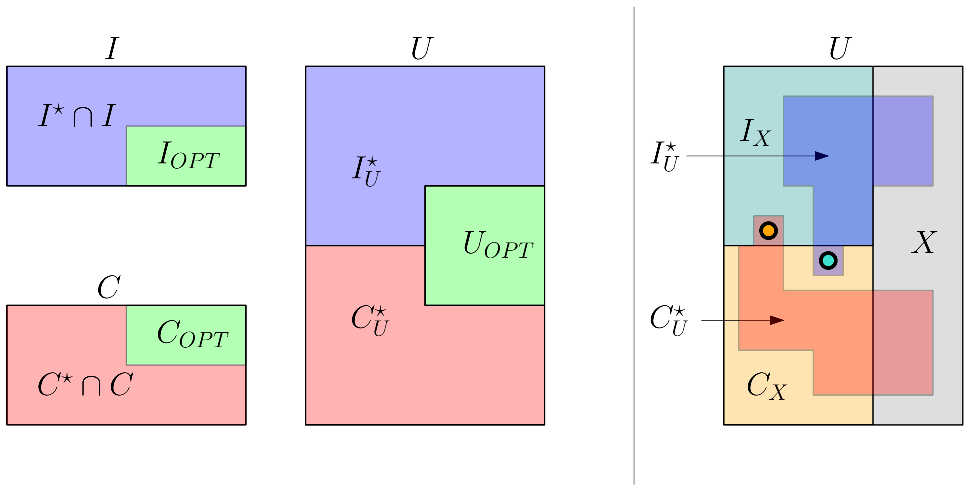

Figure 1 Illustration of Definition 13.

▶ **Observation 10 (♣).** *Let $(G, w, (C, I, U))$ be an instance of A-SVD and let $u \in U$.*

1. *Let V_1 be a vertex cover of the graph $G[I \uplus (U \setminus \{u\})]$ and let V_2 be a vertex cover of the graph $\overline{G[C \cup \{u\}]}$. Then $V_1 \uplus V_2$ is a feasible solution to $(G, w, (C, I, U))$.*
2. *Let V_3 be a vertex cover of the graph $G[I \cup \{u\}]$ and let V_4 be a vertex cover of the graph $\overline{G[C \uplus (U \setminus \{u\})]}$. Then $V_3 \uplus V_4$ is a feasible solution to $(G, w, (C, I, U))$.*

Observation 9 has some interesting consequences. For instance, it implies that when the "unconstrained" set U in an A-SVD instance is *empty*, an optimum solution to the A-SVD instance corresponds to optimum solutions of two Weighted Vertex Cover instances derived from the A-SVD instance in a natural fashion.

▶ **Lemma 11 (♣).** *Let $S^\star$ be an optimum solution to an A-SVD instance $(G, w, (C, I, U = \emptyset))$. Then the set $(S^\star \cap I)$ is an optimum solution to the* wVC *instance $(G[I], w)$, and the set $(S^\star \cap C)$ is an optimum solution to the* wVC *instance $(\overline{G[C]}, w)$.*

This in turn implies that given an A-SVD instance in which the unconstrained set U is empty, we can find a 2-factor approximate solution to the instance in polynomial time.

▶ **Lemma 12 (♣).** *There is a deterministic algorithm that finds a 2-factor approximate solution to an A-SVD instance that is of the form $(G, w, (C, I, U = \emptyset))$, in $\mathcal{O}(|E(G)|)$ time.*

This idea generalizes as follows. Let OPT be an optimum solution to an A-SVD instance $(G, w, (C, I, U))$. Suppose the split graph $G - OPT$ has a split partition $(C^\star, I^\star)$ such that vertices from the unconstrained set U *contribute a small weight* to either the clique $C^\star$ or the independent set $I^\star$. Then a variant of the algorithm in the proof of Lemma 12 yields a small-factor approximate solution to the instance, in polynomial time. We state this formally in Lemma 16 below, for which we need some notation (see Figure 1).

▶ **Definition 13.** *Let $(G, w, (C, I, U))$ be an instance of A-SVD, and let $\varepsilon \geq 0$ be a constant. Let $OPT \subseteq V(G)$ be an optimum solution of $(G, w, (C, I, U))$ and let $(C^\star, I^\star)$ be a split partition of the split graph $G^\star = (G - OPT)$ such that $C^\star \subseteq (C \cup U)$ and $I^\star \subseteq (I \cup U)$ hold. Let $C^\star_U = (C^\star \cap U)$ be the set of vertices from the unconstrained set U which become part of the clique $C^\star$ and let $I^\star_U = (I^\star \cap U)$ be the set of vertices from U which become part of the independent set $I^\star$ in $G^\star$. Let $U_{OPT} = (U \cap OPT)$, $C_{OPT} = (C \cap OPT)$ and $I_{OPT} = (I \cap OPT)$.*

Further, let X be a 5-factor approximate solution of the Split Vertex Deletion *instance* $(G[U], w_U)$ defined by the induced subgraph $G[U]$*, and let* (C_X, I_X) *be a split partition of the split graph* $G[U] - X$*.*

▶ Remark 14. Given an instance $(G, w, (C, I, U))$ of A-SVD we can, using Theorem 6, compute such a set X and partition (C_X, I_X) in polynomial time.

▶ **Observation 15 (♣).** *Let* $(G, w, (C, I, U)), X, I_X, C_X, I^\star_U, C^\star_U$ *be as in Definition 13. Then both* $|I^\star_U \setminus (X \cup (I_X \setminus C^\star_U))| \leq 1$ *and* $|C^\star_U \setminus (X \cup (C_X \setminus I^\star_U))| \leq 1$ *hold.*

▶ **Lemma 16 (♣).** *Let* $(G, w, (C, I, U)), \varepsilon, OPT, C^\star_U, I^\star_U$ *be as in Definition 13. Let* S_1 *be a 2-factor approximate solution for the* WVC *instance* $(G[I \cup U], w)$ *and* S_2 *a 2-factor approximate solution for the* WVC *instance* $(\overline{G[C]}, w)$*. Let* $S_{12} = (S_1 \cup S_2)$*. Let* S_3 *be a 2-factor approximate solution for the* WVC *instance* $(\overline{G[C \cup U]}, w)$ *and* S_4 *a 2-factor approximate solution for the* WVC *instance* $(G[I], w)$*. Let* $S_{34} = (S_3 \cup S_4)$*. Then the sets* S_{12} *and* S_{34} *can be computed in* $\mathcal{O}(|E(G)|)$ *time. Further,*

1. *If* $w(C^\star_U) \leq \frac{\varepsilon \cdot w(OPT)}{2}$ *holds then the set* S_{12} *is a* $(2+\varepsilon)$*-factor approximate solution for the* Annotated Split Vertex Deletion *instance* $(G, w, (C, I, U))$*.*
2. *If* $w(I^\star_U) \leq \frac{\varepsilon \cdot w(OPT)}{2}$ *holds then the set* S_{34} *is a* $(2+\varepsilon)$*-factor approximate solution for the* Annotated Split Vertex Deletion *instance* $(G, w, (C, I, U))$*.*

▶ Remark 17. Note that these two cases are neither exclusive nor exhaustive.

By repeatedly applying the procedure in the proof of Lemma 16 and taking the minimum, we can find a $(2+\varepsilon)$-factor approximate solution in polynomial time even in the more general case where there is at most one "heavy" vertex in $C^\star_U$ or $I^\star_U$ that

▶ **Lemma 18 (♣).** *Let* $(G, w, (C, I, U)), \varepsilon, OPT, C^\star_U, I^\star_U$ *be as in Definition 13. For each vertex* $u \in U$ *let* S^u_1 *be a 2-factor approximate solution for the* WVC *instance* $(G[I \cup (U \setminus \{u\})], w)$*,* S^u_2 *a 2-factor approximate solution for the* WVC *instance* $(\overline{G[C \cup \{u\}]}, w)$*, and let* $S^u_{12} = S^u_1 \cup S^u_2$*. Let* S^u_3 *be a 2-factor approximate solution for the* WVC *instance* $(\overline{G[C \cup (U \setminus \{u\})]}, w)$*,* S^u_4 *a 2-factor approximate solution for the* WVC *instance* $(G[I \cup \{u\}], w)$*, and let* $S^u_{34} = S^u_3 \cup S^u_4$*. Finally, let* $S^\dagger$ *be a set of the form* S^u_{12} *of the minimum weight and let* $S^\ddagger$ *be a set of the form* S^u_{34} *of the minimum weight, both minima taken over all vertices* $u \in U$*.*

The sets $S^\dagger$ *and* $S^\ddagger$ *can be computed in* $\mathcal{O}(|V(G)| \cdot |E(G)|)$ *time. Further,*

1. *If* $w(C^\star_U \setminus \{c^\star\}) \leq \frac{\varepsilon \cdot w(OPT)}{2}$ *holds for some vertex* $c^\star \in C^\star_U$ *then the set* $S^\dagger$ *is a* $(2+\varepsilon)$*-factor approximate solution for the A-SVD instance* $(G, w, (C, I, U))$*.*
2. *If* $w(I^\star_U \setminus \{i^\star\}) \leq \frac{\varepsilon \cdot w(OPT)}{2}$ *holds for some vertex* $i^\star \in I^\star_U$ *then the set* $S^\ddagger$ *is a* $(2+\varepsilon)$*-factor approximate solution for the A-SVD instance* $(G, w, (C, I, U))$*.*

▶ Remark 19. Note that these two cases are neither exclusive nor exhaustive.

▶ **Definition 20.** *Let* $(G, w, (C, I, U)), \varepsilon, OPT, C^\star, I^\star, C^\star_U, I^\star_U$ *be as in Definition 13. We say that* $(G, w, (C, I, U))$ *is an* easy *instance if* $U = \emptyset$ *holds, or if* at least one *of the following holds: (i)* $w(C^\star_U) \leq \frac{\varepsilon \cdot w(OPT)}{2}$*, (ii)* $w(I^\star_U) \leq \frac{\varepsilon \cdot w(OPT)}{2}$*, (iii)* $w(C^\star_U \setminus \{c^\star\}) \leq \frac{\varepsilon \cdot w(OPT)}{2}$ *holds for some vertex* $c^\star \in C^\star_U$*, (iv)* $w(I^\star_U \setminus \{i^\star\}) \leq \frac{\varepsilon \cdot w(OPT)}{2}$ *holds for some vertex* $i^\star \in I^\star_U$*. We say that* $(G, w, (C, I, U))$ *is a* hard *instance otherwise.*

From Lemma 12, Lemma 16 and Lemma 18 we get

▶ **Corollary 21.** *There is an algorithm which, given an* easy *instance* $(G, w, (C, I, U))$ *of A-SVD and a constant* $\varepsilon > 0$ *as input, computes a* $(2+\varepsilon)$*-factor approximate solution for* $(G, w, (C, I, U))$ *in deterministic polynomial time.*

▶ **Lemma 22** (♣). *Let $(G, w, (C, I, U))$ be a* hard *instance of A-SVD and let $\varepsilon, C^\star_U, I^\star_U, X, I_X, C_X$ be as in Definition 13. Then the following hold:*

1. $w(X \cup (I_X \setminus C^\star_U)) < (1 + \frac{12}{\varepsilon}) \cdot w(I^\star_U)$
2. $w(X \cup (C_X \setminus I^\star_U)) < (1 + \frac{12}{\varepsilon}) \cdot w(C^\star_U)$

Recall the notion of heavy and light pairs from Definition 3.

▶ **Lemma 23** (♣). *Let $(G, w, (C, I, U))$ be a* hard *instance of A-SVD and let $\varepsilon, OPT, C^\star, I^\star, C^\star_U, I^\star_U$ be as in Definition 13. Suppose $(I^\star_U, C^\star_U)$ is a* heavy pair*. Let $I^{\bigcirc} = \{v \in I^\star_U \; ; \; w(N(v) \cap C^\star_U) \geq \frac{w(C^\star_U)}{4}\}$ be the set of vertices in $I^\star_U$ which have a "heavy" neighborhood in $C^\star_U$, and let $i^{\bigcirc}$ be a heaviest vertex in $I^{\bigcirc}$, i.e. a vertex of maximum weight. Let $C^{\bigcirc} = \{v \in C^\star_U \; ; \; w((I^\star_U \setminus \{i^{\bigcirc}\}) \setminus (N(v) \cap I^\star_U)) \geq \frac{w(I^\star_U \setminus \{i^{\bigcirc}\})}{4}\}$ be the set of vertices in $C^\star_U$ which have a "heavy"* non-neighborhood *in the subset $I^\star_U \setminus \{i^{\bigcirc}\}$, and let $c^{\bigcirc}$ be a heaviest vertex in $C^{\bigcirc}$. Let $I^{\square} = \{v \in (I^\star_U \setminus \{i^{\bigcirc}\}) \; ; \; w(N(v) \cap (C^\star_U \setminus \{c^{\bigcirc}\})) \geq \frac{w(C^\star_U \setminus \{c^{\bigcirc}\})}{4}\}$ be the set of vertices in $I^\star_U \setminus \{i^{\bigcirc}\}$ which have a "heavy" neighborhood in $C^\star_U \setminus \{c^{\bigcirc}\}$, and let $C^{\square} = \{v \in (C^\star_U \setminus \{c^{\bigcirc}\}) \; ; \; w((I^\star_U \setminus \{i^{\bigcirc}\}) \setminus (N(v) \cap I^\star_U)) \geq \frac{w(I^\star_U \setminus \{i^{\bigcirc}\})}{4}\}$ be the set of vertices in $(C^\star_U \setminus \{c^{\bigcirc}\})$ which have a "heavy"* non-neighborhood *in $I^\star_U \setminus \{i^{\bigcirc}\}$.*

Then at least one of the following statements holds:

(1a) *Picking a vertex proportionately at random from the set $X \cup (I_X \setminus C^\star_U)$ yields a vertex $v \in I^{\bigcirc}$ with probability at least $1/(20(1 + \frac{12}{\varepsilon}))$.*

(1b) *Picking a vertex proportionately at random from the set $X \cup (I_X \setminus C^\star_U)$ yields a vertex $v \in I^{\square}$ with probability at least $1/(4(1 + \frac{12}{\varepsilon}))$.*

(2a) *Picking a vertex proportionately at random from the set $X \cup (C_X \setminus I^\star_U)$ yields a vertex $v \in C^{\bigcirc}$ with probability at least $1/(20(1 + \frac{12}{\varepsilon}))$.*

(2b) *Picking a vertex proportionately at random from the set $X \cup (C_X \setminus I^\star_U)$ yields a vertex $v \in C^{\square}$ with probability at least $1/(4(1 + \frac{12}{\varepsilon}))$.*

▶ **Lemma 24** (♣). *Let $(G, w, (C, I, U))$ be a* hard *instance of A-SVD and let $\varepsilon, OPT, C^\star, I^\star, C^\star_U, I^\star_U$ be as in Definition 13. Suppose $(I^\star_U, C^\star_U)$ is a* light pair*. Let $C^{\parallel} = \{v \in C^\star_U \; ; \; w(I^\star_U \setminus (N(v) \cap I^\star_U)) \geq \frac{w(I^\star_U)}{4}\}$ be the set of vertices in $C^\star_U$ which have a "heavy"* non-neighborhood *in $I^\star_U$, and let $c^{\parallel}$ be a heaviest vertex in $C^{\parallel}$. Let $I^{\parallel} = \{v \in I^\star_U \; ; \; w(N(v) \cap (C^\star_U \setminus \{c^{\parallel}\})) \geq \frac{w(C^\star_U \setminus \{c^{\parallel}\})}{4}\}$ be the set of vertices in $I^\star_U$ which have a "heavy" neighborhood in the subset $C^\star_U \setminus \{c^{\parallel}\}$, and let $i^{\parallel}$ be a heaviest vertex in $I^{\parallel}$. Let $C^{\ddagger} = \{v \in (C^\star_U \setminus \{c^{\parallel}\}) \; ; \; w((I^\star_U \setminus \{i^{\parallel}\}) \setminus (N(v) \cap I^\star_U)) \geq \frac{w(I^\star_U \setminus \{i^{\parallel}\})}{4}\}$ be the set of vertices in $C^\star_U \setminus \{c^{\parallel}\}$ which have a "heavy"* non-neighborhood *in $I^\star_U \setminus \{i^{\parallel}\}$, and let $I^{\ddagger} = \{v \in (I^\star_U \setminus \{i^{\parallel}\}) \; ; \; w(N(v) \cap (C^\star_U \setminus \{c^{\parallel}\})) \geq \frac{w(C^\star_U \setminus \{c^{\parallel}\})}{4}\}$ be the set of vertices in $(I^\star_U \setminus \{i^{\parallel}\})$ which have a "heavy" neighborhood in $C^\star_U \setminus \{c^{\parallel}\}$.*

Then at least one of the following statements is true.

(1a) *Picking a vertex proportionately at random from the set $X \cup (C_X \setminus I^\star_U)$ yields a vertex $v \in C^{\parallel}$ with probability at least $1/(20(1 + \frac{12}{\varepsilon}))$, or*

(1b) *Picking a vertex proportionately at random from the set $X \cup (C_X \setminus I^\star_U)$ yields a vertex $v \in C^{\ddagger}$ with probability at least $1/(4(1 + \frac{12}{\varepsilon}))$.*

(2a) *Picking a vertex proportionately at random from the set $X \cup (I_X \setminus C^\star_U)$ yields a vertex $v \in I^{\parallel}$ with probability at least $1/(20(1 + \frac{12}{\varepsilon}))$, or*

(2b) *Picking a vertex proportionately at random from the set $X \cup (I_X \setminus C^\star_U)$ yields a vertex $v \in I^{\ddagger}$ with probability at least $1/(4(1 + \frac{12}{\varepsilon}))$.*

From Lemma 23 and Lemma 24 we get

▶ **Lemma 25.** *Let $(G, w, (C, I, U))$ be a* hard *instance of A-SVD and let $\varepsilon, OPT, C^\star, I^\star, C^\star_U$, $I^\star_U$ be as in Definition 13. Then one of the following statements is true.*

(1a) *Picking a vertex proportionately at random from $X \cup (I_X \setminus C^\star_U)$ yields a vertex from $\{v \in I^\star_U \mid w(N(v) \cap C^\star_U) \geq \frac{w(C^\star_U)}{4}\}$ with probability at least $1/20(1+\frac{12}{\varepsilon})$.*

(1b) *Picking a vertex proportionately at random from $X \cup (I_X \setminus C^\star_U)$ yields a vertex from $\{v \in I^\star_U \mid w(N(v) \cap (C^\star_U \setminus \{c^\star\})) \geq \frac{w(C^\star_U \setminus \{c^\star\})}{4}\}$ with probability at least $1/20(1+\frac{12}{\varepsilon})$, for some vertex $c^\star \in C^\star_U$.*

(2a) *Picking a vertex proportionately at random from $X \cup (C_X \setminus I^\star_U)$ yields a vertex from $\{v \in C^\star_U \mid w(I^\star_U \setminus N(v)) \geq \frac{w(I^\star_U)}{4}\}$ with probability at least $1/20(1+\frac{12}{\varepsilon})$.*

(2b) *Picking a vertex proportionately at random from $X \cup (C_X \setminus I^\star_U)$ yields a vertex from $\{v \in C^\star_U \mid w((I^\star_U \setminus \{i^\star\}) \setminus N(v)) \geq \frac{w(I^\star_U \setminus \{i^\star\})}{4}\}$ with probability at least $1/20(1+\frac{12}{\varepsilon})$, for some vertex $i^\star \in I^\star_U$.*

Proof. From Lemma 4 we get that $(I^\star_U, C^\star_U)$ is either a heavy pair or a light pair. If $(I^\star_U, C^\star_U)$ is a heavy pair then Lemma 23 applies, and at least one of the four options of that lemma holds. Option **(1a)** of Lemma 23 implies option **(1a)** of the current lemma. Option **(1b)** of Lemma 23 implies option **(1b)** of the current lemma. Options **(2a)** and **(2b)** of Lemma 23 both imply option **(2b)** of the current lemma.

If $(I^\star_U, C^\star_U)$ is a light pair then Lemma 24 applies, and at least one of the four options of that lemma holds. Option **(1a)** of Lemma 24 implies option **(2a)** of the current lemma. Option **(1b)** of Lemma 24 implies option **(2b)** of the current lemma. Options **(2a)** and **(2b)** of Lemma 24 both imply option **(1b)** of the current lemma.

Thus in every case, one of the four options of the current lemma holds. ◀

▶ **Lemma 26 (♣).** *Let $(G, w, (C, I, U))$ be a* hard *instance of A-SVD and let $\varepsilon, OPT, C^\star, I^\star$, $C^\star_U, I^\star_U$ be as in Definition 13.*

1. *There is a randomized polynomial-time algorithm which, given $(G, w, (C, I, U))$ as input, picks a vertex proportionately at random from the set $X \cup (I_X \setminus C^\star_U)$ with probability at least $\frac{1}{2}$. That is, the algorithm runs in polynomial time and outputs a vertex v, and the following hold with probability at least $\frac{1}{2}$: (i) $v \in X \cup (I_X \setminus C^\star_U)$, and (ii) for any $x \in (X \cup (I_X \setminus C^\star_U))$, $Pr[v = x] = w(x)/w(X \cup (I_X \setminus C^\star_U))$.*
2. *There is a randomized polynomial-time algorithm which, given $(G, w, (C, I, U))$ as input, picks a vertex proportionately at random from the set $X \cup (C_X \setminus I^\star_U)$ with probability at least $\frac{1}{2}$. That is, the algorithm runs in polynomial time and outputs a vertex v, and the following hold with probability at least $\frac{1}{2}$: (i) $v \in X \cup (C_X \setminus I^\star_U)$, and (ii) for any $x \in (X \cup (C_X \setminus I^\star_U))$, $Pr[v = x] = w(x)/w(X \cup (C_X \setminus I^\star_U))$.*

3.1 Polynomially Bounded Weights

Let us first consider instances (G, w) of SVD which have polynomially bounded weights. Let $n = |V(G)|$. Recall that $w(v) \geq 1$ holds for each vertex v of G. We say that the weight function w is *polynomially bounded* if, in addition, $\sum_{v \in V(G)} w(v) \leq c_1 n^{c_0}$ holds for every $v \in V(G)$ and some constants c_0, c_1. For such instances we have the following theorem.

▶ **Theorem 27.** *There exists a randomized algorithm that given a graph G, a polynomially bounded weight function w on $V(G)$ and $\varepsilon > 0$, runs in time $\mathcal{O}(n^{f(\varepsilon)})$ and outputs $S \subseteq V(G)$ such that $G - S$ is a split graph and $w(S) \leq (2+\varepsilon)w(OPT)$ with probability at least $1/2$, where OPT is a minimum weight split vertex deletion set of G. Here, $f(\varepsilon) = 6 + \log(80(1 + \frac{12}{\varepsilon})) \cdot 4c_0 \log(c_1)/\log(4/3)$, where c_0, c_1 are constants such that $w(V(G)) \leq c_1 \cdot n^{c_0}$.*

Algorithm 1 Approximation algorithm for the case of polynomially bounded weights.

```
    Input: An instance $(G, w, (C, I, U))$ of A-SVD, a tuples $(\beta_1^C, \beta_2^C, \beta_1^I, \beta_2^I)$ and $\varepsilon > 0$.
    Output: A $(2 + \varepsilon)$-factor approximate solution to $(G, w, (C, I, U))$.
 1: procedure ASVD-Approx($(G, w, (C, I, U)), \varepsilon, \beta_1^C, \beta_2^C, \beta_1^I, \beta_2^I$)
 2:     if $U = \emptyset$ then
 3:         Compute a 2-approximation $S$ using Lemma 12
 4:         return $S$
 5:     end if
 6:     $X \leftarrow$ 5-approximate solution to $(G[U], w)$ from Theorem 6
 7:     $I_X, C_X \leftarrow$ the independent set and the clique in the split partition of $G[U] - X$.
 8:     Compute the sets $S_{12}$ and $S_{34}$ as described in Lemma 16.
 9:     Compute the sets $S^\dagger$ and $S^\ddagger$ as described in Lemma 18.
10:     if $\beta_1^C \geq 0$ and $\beta_2^C \geq 0$ and $\beta_1^I \geq 0$ and $\beta_2^I \geq 0$ then
11:         for all $j \in \{1, 2, \ldots, b(\varepsilon)\}$ do          ▷ $b(\varepsilon) = \lceil 80(1 + \frac{12}{\varepsilon}) \rceil$.
12:             Sample a vertex $v_I$ proportionally at random from the set $X \cup (I_X \setminus C_U^\star)$
    using Lemma 26.
13:             Set $Z_C \leftarrow N(v_I) \cap U$.
14:             Set $C' \leftarrow C \cup Z_C$
15:             Set $U' \leftarrow U \setminus Z_C$
16:             Set $S_{j,1}^C \leftarrow$ ASVD-Approx($(G, w, (C', I, U')), \varepsilon, \beta_1^C - 1, \beta_2^C, \beta_1^I, \beta_2^I$)
17:             Set $S_{j,2}^C \leftarrow$ ASVD-Approx($(G, w, (C', I, U')), \varepsilon, \beta_1^C, \beta_2^C - 1, \beta_1^I, \beta_2^I$)
18:             Sample a vertex $v_C$ proportionally at random from the set $X \cup (C_X \setminus I_U^\star)$
    using Lemma 26.
19:             Set $Z_I \leftarrow U \setminus N(v_C)$.
20:             Set $I' \leftarrow I \cup Z_I$
21:             Set $U' \leftarrow U \setminus Z_I$
22:             Set $S_{j,1}^I \leftarrow$ ASVD-Approx($(G, w, (C, I', U')), \varepsilon, \beta_1^C, \beta_2^C, \beta_1^I - 1, \beta_2^I$)
23:             Set $S_{j,1}^I \leftarrow$ ASVD-Approx($(G, w, (C, I', U')), \varepsilon, \beta_1^C, \beta_2^C, \beta_1^I, \beta_2^I - 1$)
24:         end for
25:     else
26:         for all $j \in \{1, 2, \ldots, b(\varepsilon)\}$ do
27:             $S_{j,1}^C, S_{j,2}^C, S_{j,1}^I, S_{j,2}^I \leftarrow V(G), V(G), V(G), V(G)$
28:         end for
29:     end if
30:     $S \leftarrow$ a min weight set in $\bigcup_{j=1,2,\ldots b(\varepsilon)} \{S_{j,1}^C, S_{j,2}^C, S_{j,1}^I, S_{j,2}^I\} \bigcup \{S_{12}, S_{34}, S^\dagger, S^\ddagger\}$.
31:     return $S$
32: end procedure
```

Proof. Let us fix an optimum solution OPT to (G, w). We treat the instance (G, w) of SVD as an instance $(G, w, (C = \emptyset, I = \emptyset, U = V(G)))$ of A-SVD, and apply Algorithm 1 to it, along with the given value of ε and four integers $\beta_1^C, \beta_2^C, \beta_1^I, \beta_2^I$ each set to $\lceil \log_{4/3}(w(V(G))) \rceil$. Note that, as w is polynomially bounded, we have $w(V(G)) \leq c_1 n^{c_0}$ for some constants c_0, c_1, and hence $\beta' \leq c_2 \log(n)$ for every $\beta' \in \{\beta_1^C, \beta_2^C, \beta_1^I, \beta_2^I\}$ where c_2 is a constant. We will show that the value $\beta = 1 + \beta_1^C + \beta_2^C + \beta_1^I + \beta_2^I \leq 1 + 4c_2 \log(n)$ is an upper-bound on the depth of the recursion tree of Algorithm 1, and that in each recursive call this value drops by 1. Hence the depth of recursion is bounded by β. Each recursive call is made on more constrained sub-instances of A-SVD where the underlying graph G, weight function w, and the value of ε remain fixed. When one of $\{\beta_1^C, \beta_2^C, \beta_1^I, \beta_2^I\}$ falls to -1, we argue that the current instance must be an easy instance (see Definition 20), assuming all the recursive calls leading the current call were "good" (as defined below). During its run the algorithm also computes a 5-approximate solution X to $(G[U], w)$ using Theorem 6; let (I_X, C_X) be a fixed split partition of $G[U] - X$. We have a split partition $(C^\star, I^\star)$ of $G - OPT$ and we define $I_U^\star = I^\star \cap U, C_U^\star = C^\star \cap U$. These sets, introduced in Definition 13, play an important role in Algorithm 1 and its analysis.

To argue the correctness of Algorithm 1, we require the following definition. An invocation ASVD-APPROX$(G, w, (C, I, U), \varepsilon, \beta_1^C, \beta_2^C, \beta_1^I, \beta_2^I)$ is *good* if the following conditions are true:

- $\beta_1^C \geq \log_{4/3}(w(C_U^\star))$,
- $\beta_2^C \geq \log_{4/3}(w(C_U^\star \setminus \{c\}))$ for some $c \in C_U^\star$,
- $\beta_1^I \geq \log_{4/3}(w(I_U^\star))$, and
- $\beta_2^I \geq \log_{4/3}(w(I_U^\star \setminus \{i\}))$ for some $i \in I_U^\star$.

Note that the definitions of $C_U^\star$ and $I_U^\star$ depend only on $(G, w, (C, I, U))$ and on the optimum solution OPT that was fixed at the beginning. These sets are hypothetical and unknown, and we can't directly test if an invocation of Algorithm 1 is a good invocation. However, observe that in the initial call, $U = V(G)$ and we set each of $\beta_1^C, \beta_2^C, \beta_1^I, \beta_2^I$ to $\lceil \log_{4/3}(w(V(G))) \rceil$, and hence the initial invocation is good. We will argue that if the current invocation is good and the instance of A-SVD is a hard instance (see Definition 20), then each recursive call made by the algorithm is good with a constant probability (which depends on ε). Then (via an induction) we argue that a good recursive call will return a $(2+\varepsilon)$-approximate solution with probability at least $\frac{1}{2}$, and hence with constant probability we obtain a $(2+\varepsilon)$-approximate solution from a recursive call. To boost the probability of success to $\frac{1}{2}$, we need to repeat this process constantly many times, so we make constantly many recursive calls. Finally, to bound the running time, we argue that the depth of the recursion tree is bounded by $\beta = \mathcal{O}(\log n)$, and we make constantly many recursive calls in each invocation of the algorithm. So the total number of calls made to this algorithm, which is upper-bounded by the size of the recursion tree, is $n^{\mathcal{O}(1)}$. This means that in polynomial time, with probability at least 1/2, we obtain a $(2+\varepsilon)$-approximate solution to (G, w). Let us now present these arguments formally.

Let us recall the optimum solution OPT to (G, w) that was fixed at the beginning. We say that an instance $(G, w, (C, I, U))$ is a *nice instance* if the solution OPT is also an optimum solution to this A-SVD instance. This means that a split partition $(C^\star, I^\star)$ of $G - OPT$ satisfies, $C^\star \cap I = \emptyset$ and $I^\star \cap C = \emptyset$. Note that this condition is trivially satisfied at the beginning for the starting instance $(G, w, (C = \emptyset, I = \emptyset, U = V(G))$. Let us consider an invocation of Algorithm 1 on a nice instance of $(G, w, (C, I, U))$ with polynomially bounded weight function w and $\beta_1^C, \beta_2^C, \beta_1^I, \beta_2^I$ such that it is a good invocation. Let S denote the solution returned by it. We will show that S is a $(2+\varepsilon)$-approximate solution with probability at least $\frac{1}{2}$, by an induction on $|U|$. Suppose that $|U| = 0$, i.e. $U = \emptyset$. Then Lemma 12 ensures that S is a 2-approximate solution. This forms the base case of our induction on $|U|$.

Now suppose that $|U| > 0$, and we have two cases depending on whether $(G, w, (C, I, U))$ is an easy instance or not. If it is an easy instance, then either the premise of Lemma 16 or the premise of Lemma 18 holds. Hence, one of $S_{12}, S_{34}, S^\dagger, S^\ddagger$ is a $(2+\varepsilon)$-approximation to $(G, w, (C, I, U))$. Moreover, we claim that if any one of $\beta_1^C, \beta_2^C, \beta_1^I, \beta_2^I$ drops to -1, then the instance $(G, w, (C, I, U))$ is an easy instance. Consider the case when $\beta_2^C = -1$. Then $\log_{4/3}(w(C_U^\star \setminus \{c\})) = -1$ for some $c \in C_U^\star$. This means $w(C_U^\star \setminus \{c\}) < 3/4$, and since $w(v) \geq 1$ for every $v \in V(G)$, it must be the case that $C_U^\star = \{c\}$. Hence, the premise of Lemma 18 holds and we obtain a $(2+\varepsilon)$-approximate solution for $(G, w, (C, I, U))$. Similar arguments apply to the other cases, i.e. when $\beta_1^C = -1$, or $\beta_1^I = -1$ or $\beta_2^I = -1$, and we can obtain a $(2+\varepsilon)$-approximation in all these cases. Therefore, in all these cases S is a $(2+\varepsilon)$-approximation to $(G, w, (C, I, U))$.

Now, consider the case when the given instance is a hard instance, i.e. $U \neq \emptyset$ and the premises of Lemma 16 and Lemma 18 don't hold. In this case $\beta_1^C, \beta_2^C, \beta_1^I, \beta_2^I \geq 0$. Recall that X is a 5-approximate solution to SVD in the subgraph $G[U]$, and hence $w(X) \leq 5 \cdot w(OPT)$. We will make recursive calls on instances of A-SVD of the form $(G, w, (C', I', U'))$ such that $C \subseteq C', I \subseteq I'$ and $U' \subsetneq U$. Suppose that $(G, w, (C', I', U'))$ is a nice instance. Then by the induction hypothesis, as $|U'| < |U|$, we can assume that Algorithm 1 returns a $(2+\varepsilon)$-approximate solution $\widehat{S}$ to this instance with probability at least 1/2. This is an approximate solution to the current instance as well:

▷ Claim 28. $\widehat{S}$ is a $(2+\varepsilon)$-approximate solution to $(G, w, (C, I, U))$

Proof. Observe that, since $\widehat{S}$ is feasible solution to the nice instance $(G, w, (C', I', U'))$, there is a split partition $(C_{\widehat{S}}, I_{\widehat{S}})$ of $G - \widehat{S}$ such that $C' \cap I_{\widehat{S}} = \emptyset$ and $I' \cap C_{\widehat{S}} = \emptyset$. Therefore, we have $C \cap I_{\widehat{S}} = \emptyset$ and $I \cap C_{\widehat{S}} = \emptyset$, i.e. $\widehat{S}$ is a feasible solution to $(G, w, (C, I, U))$. Since $w(\widehat{S}) \leq (2+\varepsilon)w(OPT)$, the claim is true. ◁

Let us now consider the recursive calls made by the algorithm for each $j \in \{1, 2, \ldots, b(\varepsilon) = \lceil 80(1 + \frac{12}{\varepsilon})\rceil\}$, and argue that with a constant probability (depending on ε) we can obtain a $(2+\varepsilon)$-approximation to the given instance. In each recursive call, one of $\beta_1^C, \beta_2^C, \beta_1^I, \beta_2^I$ drops by exactly 1. Let us fix $j \in \{1, 2, \ldots, b(\varepsilon)\}$ and consider the two vertices v_I, v_C sampled using Lemma 26. Since $(G, w, (C, I, U))$ is a hard instance, the following hold.

- With probability at least 1/2, $v_I \in X \cup (I_X \setminus C_U^\star)$, and for any $x \in (X \cup (I_X \setminus C_U^\star))$, $Pr[v_I = x] = w(x)/w(X \cup (I_X \setminus C_U^\star))$.
- With probability at least 1/2, $v_C \in X \cup (C_X \setminus I_U^\star)$, and for any $x \in (X \cup (C_X \setminus I_U^\star))$, $Pr[v_C = x] = w(x)/w(X \cup (C_X \setminus I_U^\star))$.

By the induction hypothesis, any good invocation ASVD-Approx$(G, w, (C', I', U'), \varepsilon, \widehat{\beta_1^C}, \widehat{\beta_2^C}, \widehat{\beta_1^I}, \widehat{\beta_2^I})$ where $(G, w, (C', I', U'))$ is a nice instance and $|U'| < |U|$ holds, returns a $(2+\varepsilon)$-approximate solution to $(G, w, (C', I', U'))$ with probability at least $\frac{1}{2}$. We now have four cases, depending on which of the four statements in Lemma 25 is true for $(G, w, (C, I, U))$. In each case we will argue that with constant probability, we make a good recursive call on a nice instance and obtain a $(2+\varepsilon)$-approximate solution from it.

(i) Suppose that statement (1a) of Lemma 25 is true. That is, picking a vertex proportionally at random from $X \cup (I_X \setminus C_U^\star)$ yields a vertex from $\{v \in I_U^\star \mid w(N(v) \cap C_U^\star) \geq \frac{w(C_U^\star)}{4}\}$ with probability at least $1/20(1+\frac{12}{\varepsilon})$. Then $v_I \in \{v \in I_U^\star \mid w(N(v) \cap C_U^\star) \geq \frac{w(C_U^\star)}{4}\}$ with probability at least $1/40(1+\frac{12}{\varepsilon})$. As $v_I \in I_U^\star$, every vertex in $Z_C = N(v_I) \cap U$ must either be in OPT_U or in $C_U^\star$. Furthermore, $w(Z_C \cap C_U^\star) \geq \frac{w(C_U^\star)}{4}$. Let $U' = U \setminus Z_C$, $C' = C \cup Z_C$ and consider the invocation ASVD-Approx$(G, w, (C', I, U'), \varepsilon, \beta_1^C - 1, \beta_2^C, \beta_1^I, \beta_2^I)$. Let us argue that it is a good invocation. By definition $C_{U'}^\star = C^\star \cap U'$ satisfies $w(C_{U'}^\star) \leq \frac{3}{4} w(C_U^\star)$. Therefore, as $\beta_1^C \geq \log_{4/3}(w(C_U^\star))$, we have $\beta_1^C - 1 \geq \log_{4/3}(w(C_{U'}^\star))$. Furthermore, observe that $\beta_C^2 \geq \log_{4/3}(w(C_{U'}^\star \setminus \{c^\star\}))$, and I, β_1^I, β_2^I remain unchanged. Hence, assuming that the current invocation is good, this invocation is also good. Let us argue that $(G, w, (C', I, U'))$ is a nice instance, i.e. OPT is an optimum solution to it. Towards this, recall that $C' = C \cup Z_C$ where $Z_C = N(v_I) \cap U$ and $v_I \in I_U^\star \subseteq I^\star$. Hence, every vertex in Z_C is either in OPT or in $C^\star$, i.e. $Z_C \cap I^\star = \emptyset$. Since OPT is feasible for $(G, w, (C, I, U))$ we have that $C \cap I^\star = \emptyset$. Therefore, $C' \cap I^\star = (C \cup Z_C) \cap I^\star = \emptyset$, and hence OPT is a feasible solution for $(G, w, (C', I, U'))$. Finally, as any feasible solution for $(G, w, (C', I, U'))$ is also feasible for (G, w), OPT is an optimum solution for $(G, w, (C', I, U'))$. Now $|U'| < |U|$, and by the induction hypothesis, this invocation returns a solution $S^C_{j,1}$ to $(G, w, (C', I, U'))$ with probability at least 1/2. By Claim 28, $S^C_{1,j}$ is a $(2+\varepsilon)$-approximate solution to $(G, w, (C, I, U))$. Hence, we obtain a solution $S^C_{1,j}$ that is a $(2+\varepsilon)$-approximation to $(G, w, (C, I, U))$, and this event happens with probability at least $1/80(1+\frac{12}{\varepsilon})$. Note that β^C_1 drops by 1 in the recursive call .

(ii) Suppose that statement (1b) of Lemma 25 is true. That is, picking a vertex proportionately at random from $X \cup (I_X \setminus C^\star_U)$ yields a vertex from $\{v \in I^\star_U \mid w(N(v) \cap (C^\star_U \setminus \{c^\star\})) \geq \frac{w(C^\star_U)\setminus\{c^\star\}}{4}\}$ with probability at least $1/20(1+\frac{12}{\varepsilon})$, for some vertex $c^\star \in C^\star_U$ (as determined by Lemma 25). Then, with probability at least $1/40(1+\frac{12}{\varepsilon})$, $v_I \in \{v \in I^\star_U \mid w(N(v) \cap (C^\star_U \setminus \{c^\star\})) \geq \frac{w(C^\star_U)\setminus\{c^\star\}}{4}\}$. As $v_I \in I^\star_U$, every vertex in $Z_C = N(v_I) \cap U$ must either be in OPT or in $C^\star_U$. Let $C' = C \cup Z_C, U' = U \setminus Z_C$ and consider the invocation ASVD-Approx$(G, w, (C', I, U'), \varepsilon, \beta^C_1, \beta^C_2 - 1, \beta^I_1, \beta^I_2)$. Let us argue that it is a good invocation. Let $\widehat{C} = (C^\star_U \setminus \{c^\star\}) \setminus N(v_I)$ and $C^\star_{U'} = C^\star \cap U'$, and note that either $C^\star_{U'} = \widehat{C}$ or $C^\star_{U'} = \widehat{C} \cup \{c^\star\}$. Since $w(\widehat{C}) \leq \frac{3}{4} w(C^\star_U \setminus \{c^\star\})$ by the choice of v_I, we have $\log_{4/3}(w(\widehat{C})) \leq \log_{4/3}(w(C^\star_U \setminus \{c^\star\}) - 1 \leq \beta^C_2 - 1$. Therefore, if $C^\star_{U'} = \widehat{C}$, then for any arbitrary $c' \in C^\star_{U'}$ we have $\beta^C_2 - 1 \geq \log_{4/3}(w(C^\star_{U'} \setminus \{c'\}))$; otherwise $C^\star_{U'} = \widehat{C} \cup \{c^\star\}$, and $\beta^C_2 - 1 \geq \log_{4/3}(w(C^\star_{U'} \setminus \{c^\star\}))$. Furthermore, observe that β^C_1 is unchanged and $C^\star_{U'} \subseteq C^\star_U$, we have $\log_{4/3}(w(C^\star_{U'})) \leq \beta^C_1$. Similarly, I, β^I_1, β^I_2 are also unchanged. Hence, this invocation is good. Next, as in the previous case, we can argue that $(G, w, (C', I, U')$ is a nice instance. Then, as $|U'| < |U|$, by the induction hypothesis the invocation returns a $(2+\varepsilon)$-approximate solution $S^C_{j,2}$ to $(G, w, (C', I, U'))$ with probability at least 1/2. By Claim 28, $S^C_{j,2}$ is a $(2+\varepsilon)$-approximate solution to $(G, w, (C, I, U))$. Hence, we obtain a solution $S^C_{j,2}$ that is a $(2+\varepsilon)$-approximation to $(G, w, (C, I, U))$, and this event happens with probability at least $1/80(1+\frac{12}{\varepsilon})$. Note that β^C_2 drops by 1 in recursive call made here.

(iii) Suppose that statement (2a) of Lemma 25 is true. This case is symmetric to Case-(i), above, where the arguments are made with respect to $v_C \in X \cup (C_X \setminus I^\star_U)$. Here $v_C \in \{v \in C^\star_U \mid w(I^\star_U \setminus N(v)) \geq \frac{w(I^\star_U)}{4}\}$ with probability at least $1/40(1+\frac{12}{\varepsilon})$. We consider the instance $(G, w, (C, I', U'))$ where $Z_I = U \setminus N(v_C)$, $I' = I \cup Z_I$ and $U' = U \setminus Z_I$. We can argue that this invocation is good and the instance $(G, w, (C, I', U'))$ is nice. Then, as $|U'| \leq |U|$, by the induction hypothesis, this invocation returns a $(2+\varepsilon)$-approximate solution to $(G, w, (C, I', U'))$ with probability at least 1/2. Let $S^I_{j,1}$ denote this solution, and we argue that it is also a $(2+\varepsilon)$-approximate solution to $(G, w, (C, I, U))$. In conclusion, we obtain a solution $S^I_{j,1}$ that is a $(2+\varepsilon)$-approximation to $(G, w, (C, I, U))$, and this event happens with probability at least $1/80(1+\frac{12}{\varepsilon})$. Note that β^I_1 drops by 1 in recursive call made here.

(iv) Suppose that statement (2b) of Lemma 25 is true. This case is symmetric to Case-(ii) above. Here we have a vertex $v_C \in \{v \in C^\star_U \mid w(I^\star_U \setminus N(v)) \geq \frac{w(I^\star_U)}{4}\}$ with probability at least $1/40(1+\frac{12}{\varepsilon})$. We make a recursive call ASVD-Approx$(G, w, (C, I', U'), \varepsilon, \beta^C_1, \beta^C_2, \beta^I_1, \beta^I_2 - 1)$, where $I' = I \cup Z_I$, $U' = U \setminus Z_I$ and $Z_I = U \setminus N(v_C)$. Here, we obtain a solution $S^I_{j,2}$ that is a $(2+\varepsilon)$-approximation to $(G, w, (C, I, U))$, and this event happens with probability at least $1/80(1+\frac{12}{\varepsilon})$. Note that β^I_2 drops by 1 in recursive call made here.

Therefore, if $(G, w, (C, I, U))$ is a hard instance, then for each $j \in \{1, 2, \ldots, b(\varepsilon)\}$, one of $S^C_{j,1}, S^C_{j,2}, S^I_{j,1}, S^I_{j,2}$ is a $(2+\varepsilon)$-approximate solution to it with probability at least $1/80(1+\frac{12}{\varepsilon})$. Note that the recursive calls made for any two distinct $j, j' \in \{1, 2, \ldots, b(\varepsilon)\}$ are independent events. Therefore, by setting $b(\varepsilon) = \lceil 80(1+\frac{12}{\varepsilon})\rceil$, we obtain that with probability at least 1/2 there exists $j \in \{1, 2, \ldots, b(\varepsilon)\}$ such that one of $S^C_{j,1}, S^C_{j,2}, S^I_{j,1}, S^I_{j,2}$ is a $(2+\varepsilon)$-approximate solution to $(G, w, (C, I, U))$.

Finally, let us bound the running time of this algorithm. Towards this, we must bound the total number of calls made to Algorithm 1, when run on an instance (G, w) with polynomially bounded weights. Observe that, we start with an instance $(G, w, (C = \emptyset, I = \emptyset, U = V(G)))$ of A-SVD along with $\beta^C_1, \beta^C_2, \beta^I_1, \beta^I_2$ set to $\lceil \log_{4/3}(w(V(G))\rceil = c_2 \log(n)$ for some constant c_2. Then, for each instance $(G, w, (C, I, U))$, we make $b(\varepsilon)$ recursive calls and at least one of $\beta^C_1, \beta^C_2, \beta^I_1, \beta^I_2$ drops by 1 in each of these calls. Additionally U drops to a strict subset in each of these calls. Hence in a finite number of steps, either U becomes empty, or one of $\beta^C_1, \beta^C_2, \beta^I_1, \beta^I_2$ becomes equal to -1, and we reach an easy instance. Observe that this must happen at some point before the depth of recursion exceeds $\beta = 1 + 4c_2 \log(n)$. Hence, the number of recursive calls made for the instance (G, w) is upper bounded by $b(\varepsilon)^\beta = \mathcal{O}(n^{h(\varepsilon)})$ where $h(\varepsilon) = \log(80(1 + \frac{12}{\varepsilon})) \cdot 4c_0 \log(c_1)/\log(4/3)$. Recall that c_0, c_1 are constants such that $w(V(G)) \leq c_1 \cdot n^{c_0}$. Observe that in each recursive call, we spend $O(n^6)$ time (excluding the recursive calls). Hence the total running time is upper-bounded by $n^{f(\varepsilon)}$ where $f(\varepsilon) = 6 + \log(80(1 + \frac{12}{\varepsilon})) \cdot 4c_0 \log(c_1)/\log(4/3)$. Alternatively, this bound on the running time can be obtained from the Master Theorem. ◀

3.2 General Weight Functions

In this section, we extend Theorem 27 to instances of SVD with general weight function. In particular we show that given an instance with general weights, we can construct an instance with polynomially-bounded weights such that an approximate solution to the new instance can be lifted back to the original instance.

▶ **Lemma 29** (♣). *Let (G, w) be an instance of SVD, and $\varepsilon > 0$ be a constant. Then we can construct another instance (G', w') of SVD such that G' is a subgraph of G and given any α-approximate solution to (G', w') where $\alpha \leq 5$, we can obtain an $(\alpha + \varepsilon)$-approximate solution to (G, w). Moreover, the weight function w' is polynomially bounded, and $w'(V(G')) \leq \frac{30n^2}{\varepsilon}$.*

We have the following corollary of Theorem 27 and Lemma 29.

▶ **Theorem 30.** *There exists a randomized algorithm that given a graph G, a weight function w on $V(G)$ and $\varepsilon > 0$, runs in time $\mathcal{O}(n^{g(\varepsilon)})$ and outputs $S \subseteq V(G)$ such that $G - S$ is a split graph and $w(S) \leq 2(1+\varepsilon)w(OPT)$ with probability at least 1/2, where OPT is a minimum weight split vertex deletion set of G. Here, $g(\varepsilon) = 6 + 8\log(80(1 + \frac{12}{\varepsilon})) \cdot \log(\frac{30}{\varepsilon})/\log(4/3)$.*

Proof. Given the instance (G, w) and ε, we apply Lemma 29 and obtain an instance (G', w'), where $w'(V(G')) \leq \frac{30n^2}{\varepsilon}$. We then apply Theorem 27 to (G', w') and ε and obtain a solution S' to it. This algorithm runs in time $|V(G')|^{g(\varepsilon)} \leq n^{g(\varepsilon}$, where $g(\varepsilon) = 6 + 8\log(80(1 + \frac{12}{\varepsilon})) \cdot \log(\frac{30}{\varepsilon})/\log(4/3)$, and with probability at least 1/2 S' is a $(2+\varepsilon)$-approximate solution to (G', w'). Then by Lemma 29, S' can be lifted to a $2(1+\varepsilon)$-approximate solution S to (G, w). ◀

4 Conclusion

One of the natural open question is to obtain a polynomial time 2-approximation algorithm for SVD and match the lower bound obtained under UGC. It will be interesting to find other implicit d-Hitting Set problems and find its correct "approximation complexity". Towards this we restate the conjecture of Fiorini et al. [5] about a concrete implicit 3-Hitting Set problem: there is a 2-approximation algorithm for Cluster Vertex Deletion matching the lower bound under UCG.

References

1 R Bar-Yehuda and S Even. A linear-time approximation algorithm for the weighted vertex cover problem. *Journal of Algorithms*, 2(2):198–203, 1981. doi:10.1016/0196-6774(81)90020-1.

2 Reuven Bar-Yehuda and Shimon Even. A linear-time approximation algorithm for the weighted vertex cover problem. *Journal of Algorithms*, 2(2):198–203, 1981.

3 Mao-cheng Cai, Xiaotie Deng, and Wenan Zang. An approximation algorithm for feedback vertex sets in tournaments. *SIAM J. Comput.*, 30(6):1993–2007, 2000.

4 Marek Cygan and Marcin Pilipczuk. Split vertex deletion meets vertex cover: New fixed-parameter and exact exponential-time algorithms. *Inf. Process. Lett.*, 113(5-6):179–182, 2013.

5 Samuel Fiorini, Gwenaël Joret, and Oliver Schaudt. Improved approximation algorithms for hitting 3-vertex paths. *CoRR*, abs/1808.10370, 2018. arXiv:1808.10370.

6 Stephane Foldes and Peter L. Hammer. Split graphs. *Proceedings of the 8th Southeastern Conference on Combinatorics, Graph Theory, and Computing*, pages 311–315, 1977.

7 Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within $2-\epsilon$. *Journal of Computer and System Sciences*, 74(3):335–349, 2008. Computational Complexity 2003. doi:10.1016/j.jcss.2007.06.019.

8 John M Lewis and Mihalis Yannakakis. The node-deletion problem for hereditary properties is np-complete. *Journal of Computer and System Sciences*, 20(2):219–230, 1980.

9 Daniel Lokshtanov, Pranabendu Misra, Joydeep Mukherjee, Fahad Panolan, Geevarghese Philip, and Saket Saurabh. 2-approximating feedback vertex set in tournaments. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1010–1018. SIAM, 2020. doi:10.1137/1.9781611975994.61.

10 Matthias Mnich, Virginia Vassilevska Williams, and Lászlo A Végh. A 7/3-approximation for feedback vertex sets in tournaments. In *24th Annual European Symposium on Algorithms (ESA 2016)*. Schloss Dagstuhl, 2016.

11 Jelani Nelson. A note on set cover inapproximability independent of universe size. *Electronic Colloquium on Computational Complexity (ECCC)*, 14(105), 2007. URL: http://eccc.hpi-web.de/eccc-reports/2007/TR07-105/index.html.

12 Ewald Speckenmeyer. On feedback problems in diagraphs. In *Graph-Theoretic Concepts in Computer Science, 15th International Workshop, WG '89, Castle Rolduc, The Netherlands, June 14-16, 1989, Proceedings*, volume 411 of *Lecture Notes in Computer Science*, pages 218–231. Springer, 1989. doi:10.1007/3-540-52292-1_16.

13 Jie You, Jianxin Wang, and Yixin Cao. Approximate association via dissociation. *Discrete Applied Mathematics*, 219:202–209, 2017. doi:10.1016/j.dam.2016.11.007.