

Growing a Random Maximal Independent Set Produces a 2-approximate Vertex Cover

Nate Veldt

Department of Computer Science and Engineering
Texas A&M University
nveldt@tamu.edu

Abstract

This paper presents a fast and simple new 2-approximation algorithm for minimum weighted vertex cover. The unweighted version of this algorithm is equivalent to a well-known greedy maximal independent set algorithm. We prove that this independent set algorithm produces a 2-approximate vertex cover, and we provide a principled new way to generalize it to node-weighted graphs. Our analysis is inspired by connections to a clustering objective called correlation clustering. To demonstrate the relationship between these problems, we show how a simple *pivot* algorithm for correlation clustering implicitly approximates a special type of hypergraph vertex cover problem. Finally, we use implicit implementations of this maximal independent set algorithm to develop fast and simple 2-approximation algorithms for certain edge-deletion problems that can be reduced to vertex cover in an approximation preserving way.

1 Introduction

A set of nodes in a graph is a vertex cover if every edge in the graph is adjacent to at least one node in the cover. The VERTEX COVER problem is the task of finding a minimum cardinality vertex cover in a graph, or a minimum weight cover in the case of node-weighted graphs. This is one of the most well-known NP-hard optimization problems, and the decision version of the problem is one of Karp’s 21 NP-complete problems [34]. There are many 2-approximation algorithms for both weighted and unweighted VERTEX COVER that date back to the 1970s and 1980s [7, 8, 24, 47, 48]. More sophisticated algorithms also exist with approximation factors that are slightly (though not a constant amount) better than 2 [8, 32, 33], while for every constant $\varepsilon > 0$ the problem is UGC-hard to approximate below a factor of $2 - \varepsilon$ [37]. Independent of the unique games conjecture, the problem is NP-hard to approximate below a factor of 1.3606 [20]. VERTEX COVER has also been studied extensively from the perspective of fixed-parameter tractability and kernelization [1, 13, 14, 15, 26, 40] and parallel approximation algorithms [27, 28, 39]. Finding a vertex cover is a key substep for many other combinatorial problems and applications [1, 29, 46, 49, 50], and many other problems are known to be reducible to or reducible from VERTEX COVER in an approximation preserving way [25, 36, 38, 49, 50]. Thus, new algorithmic techniques and hardness results for VERTEX COVER can have far reaching implications for many other problems.

This paper presents a fast and simple 2-approximation algorithm for the minimum weighted VERTEX COVER problem based on growing a maximal independent set. At each iteration, the algorithm samples a node proportional to its weight, adds it to an independent set, and then places all neighboring nodes in the vertex cover. This approach highlights several new connections between algorithmic techniques for different problems related to VERTEX COVER. The unweighted

version of our algorithm is equivalent to a well-known greedy random method for finding a maximal independent set (MIS), which selects a uniform random ordering of nodes and greedily adds nodes to an independent set [10, 11, 18, 23]. Although finding maximum independent sets and minimum vertex covers are complementary problems, they are vastly different from the perspective of approximations [22, 53]. Furthermore, *maximal* independent sets can be very different from *maximum* independent sets. Despite these differences, our work provides a proof that the greedy random MIS algorithm produces a 2-approximation for VERTEX COVER, and also provides a principled approach for generalizing this MIS algorithm to node-weighted graphs. The analysis of our algorithm also reveals a connection between approximating VERTEX COVER and approximating a problem called CORRELATION CLUSTERING [6]. In particular, the proof of our approximation guarantee is inspired by the analysis of a simple 3-approximation algorithm called PIVOT [2], which we show implicitly approximates a VERTEX COVER problem on a special type of 3-uniform hypergraph. Our results also imply that an existing $O(\log n)$ -round parallel algorithm for finding a maximal independent set simultaneously serves as an approximation algorithm for VERTEX COVER, CORRELATION CLUSTERING, and an edge-labeling problem related to the principle of strong triadic closure [49].

Finally, we show how to use implicit implementations of our maximal independent set approach to obtain fast and simple approximation algorithms for certain edge-deletion problems that can be reduced to vertex cover in an approximation preserving way. By implicit implementation, we mean that the mechanics of our algorithm are applied without forming the reduced instance of VERTEX COVER. For the problems we consider, an implicit implementation of our method can be made asymptotically faster than naively forming the reduced graph or implicitly iterating through all of the edges in the reduced VERTEX COVER instance. We specifically use our algorithm to develop a simple new combinatorial 2-approximation algorithm for a recent edge-colored hypergraph clustering objective [4], and a faster 2-approximation algorithm for a special case of the DAG EDGE DELETION problem [36].

2 Background and Related Work

Let $G = (V, E)$ denote an undirected graph with $n = |V|$ nodes and $m = |E|$ edges, where each node $v \in V$ is associated with a nonnegative weight $w_v \geq 0$. We use $N(v)$ to denote the set of neighbors of a node $v \in V$. When convenient, we will also denote the node set by $V = \{v_1, v_2, \dots, v_n\}$ and let w_i denote the weight for the i th node v_i . The goal of the minimum vertex cover problem is to find a set of nodes $S \subseteq V$ that covers all edges and has minimum weight $w(S) = \sum_{v \in S} w_v$. This can be encoded by the following binary linear program:

$$\begin{aligned} \min \quad & \sum_{v \in V} w_v x_v \\ \text{s.t.} \quad & x_u + x_v \geq 1 \text{ for } (u, v) \in E \\ & x_v \in \{0, 1\} \text{ for } v \in V. \end{aligned} \tag{1}$$

Before presenting our new algorithm, we survey existing approximation algorithms, previous research on maximal independent sets, and other related work.

2.1 Approximation Algorithms for Vertex Cover

The most widely-known 2-approximation algorithm for unweighted VERTEX COVER works by greedily building a maximal matching in G and adding all nodes adjacent to an edge in the matching to a cover. This can be implemented by iterating through edges in an arbitrary order and adding

Figure 1: Algorithms 1, 2, and 3 are well-known 2-approximation algorithms for VERTEX COVER, and Algorithm 4 is a greedy algorithm for finding a maximal independent set. All run in $O(|E|)$ time. Algorithms 2 and 3 apply to node-weighted graphs, whereas Algorithms 1 and 4 assume $w_v = 1$ for each $v \in V$. We will prove that GREEDYMIS is also a 2-approximation algorithm for unweighted VERTEX COVER, and provide a generalization for node-weighted graphs.

Algorithm 1 MATCHINGVC(G)	Algorithm 2 PITTVc(G)
$\mathcal{C} \leftarrow \emptyset$ // Initialize empty cover	$\mathcal{C} \leftarrow \emptyset$ // Initialize empty cover
for $(u, v) \in E$ do	for $(u, v) \in E$ do
if $u \notin \mathcal{C}$ and $v \notin \mathcal{C}$ then	if $u \notin \mathcal{C}$ and $v \notin \mathcal{C}$ then
// Add both nodes to cover	With prob. $\frac{w_v}{w_u + w_v}$: $\mathcal{C} \leftarrow \mathcal{C} \cup \{u\}$
$\mathcal{C} \leftarrow \mathcal{C} \cup \{u, v\}$	Otherwise: $\mathcal{C} \leftarrow \mathcal{C} \cup \{v\}$
Return \mathcal{C}	Return \mathcal{C}
Algorithm 3 LOCALRATIOVC(G)	Algorithm 4 GREEDYMIS(G)
$\mathcal{C} \leftarrow \emptyset$ // Initialize empty cover	$\mathcal{I} \leftarrow \emptyset; \mathcal{U} \leftarrow V$
for each $v \in V$ set $r(v) = w_v$	Generate random uniform node permutation σ
for $(u, v) \in E$ do	for $v = v_{\sigma(1)}, v_{\sigma(2)}, \dots, v_{\sigma(n)}$ do
$M = \min\{r(v), r(u)\}$	if $v \in \mathcal{U}$ then
$r(v) \leftarrow r(v) - M$	$\mathcal{I} \leftarrow \mathcal{I} \cup \{v\}$ and $\mathcal{U} \leftarrow \mathcal{U} \setminus \{v\}$
$r(u) \leftarrow r(u) - M$	for $u \in N(v) \cap \mathcal{U}$ do
$\mathcal{C} = \{v \in V : r(v) = 0\}$	$\mathcal{U} \leftarrow \mathcal{U} \setminus \{u\}$
Return \mathcal{C}	Return \mathcal{I}

both endpoints of an edge to the cover if the edge can be added to the matching (Algorithm 1). This algorithm is attributed both to Gavril and Yannakakis (see [24] and [45]). The local-ratio algorithm of Bar-Yehuda and Even [7, 8] (Algorithm 3) can be viewed as a generalization of this algorithm that also works on edge-weighted graphs. Pitt’s randomized algorithm [47] (Algorithm 2) also iterates through edges, but whenever it encounters an uncovered edge, it samples one of the two endpoints to add to the vertex cover. This strategy is a randomized 2-approximation for weighted VERTEX COVER. Algorithms 1, 2, and 3 can all be implemented in $O(|E|)$ time. One other way to obtain a 2-approximate VERTEX COVER in the unweighted case is to return the non-leaf nodes of any depth-first tree [48]. This can also be implemented in $O(|E|)$ time, though this method applies only to the unweighted case.

Several other algorithms achieve a 2-approximation or better for VERTEX COVER, but take longer than $O(|E|)$ time. One approach relies on solving the linear programming (LP) relaxation obtained by replacing the constraint $x_v \in \{0, 1\}$ in (1) with linear constraints $0 \leq x_v \leq 1$. If $\{x_v^*\}$ denotes an optimal set of dual variables, the set $S = \{v \in V : x_v^* \geq 1/2\}$ is a 2-approximate solution for weighted VERTEX COVER. Other more sophisticated algorithms have also been developed, including a $2 - \Theta(1/\sqrt{\log n})$ approximation algorithm for a graph with n nodes [33], and a $2 - (1 - o(1))\frac{2 \ln \ln \Delta}{\ln \Delta}$ approximation algorithm where Δ is the maximum degree of the graph [32]. The latter two algorithms rely on semidefinite programming relaxations. In the opposite direction, *list heuristic* algorithms for VERTEX COVER [5, 19] run in $O(|E|)$ time but have approximation factors worse than 2. A list heuristic is an algorithm that iterates through nodes in a fixed order and at each step makes a decision whether to add the current node to the vertex cover or not. These algorithms

are designed specifically for unweighted VERTEX COVER. The best known approximation for a list heuristic is $\frac{\sqrt{\Delta}}{2} + \frac{3}{2}$ where Δ is the maximum degree [5, 19].

Among all of these algorithms for VERTEX COVER, Algorithms 2 and 3 are unique in that they both achieve a 2-approximation for *weighted* VERTEX COVER in $O(|E|)$ time. Both of these methods rely on iterating through all edges in the graph and deciding whether to add nodes from the edge to the vertex cover. These algorithms could equivalently be described as selecting an arbitrary *uncovered* edge at each iteration, though this requires the algorithm to update the set of covered edges at the end of an iteration. The overall runtime in either case is $O(|E|)$.

2.2 Finding Maximal Independent Sets

An independent set in an undirected graph $G = (V, E)$ is a set of nodes in which no two nodes share an edge. Equivalently, a set of nodes $\mathcal{I} \subseteq V$ is an independent set if and only if its complement set $\mathcal{C} = V - \mathcal{I}$ is a vertex cover. The new approximation algorithm we develop for node-weighted VERTEX COVER can in fact be viewed as a generalization of an existing greedy algorithm for finding a maximal independent set (MIS) in an unweighted graph. This GREEDYMIS algorithm (Algorithm 4) generates a random permutation of nodes and iteratively adds nodes to an independent set. This algorithm and its slight variants date back to roughly the same time period as the earliest VERTEX COVER approximation algorithms [10, 18, 23, 27, 35]. Given the complementary relationship between independent sets and vertex covers, it may at first seem very intuitive to try to approximate VERTEX COVER using a maximal independent set algorithm. However, this simple reasoning overlooks key differences between algorithmic techniques and theoretical guarantees for finding small vertex covers and finding large independent sets. First of all, although finding a *maximum* independent set is equivalent at optimality to finding a minimum vertex cover, these problems are vastly different from the perspective of approximation algorithms, with the former problem being much harder to approximate [22, 53]. Furthermore, there can be a significant difference between a *maximum* and *maximal* independent set in a graph. As a simple example, consider a star graph on n nodes: the singleton set consisting of the center node in the star is a maximal independent set of size 1, but the the maximum independent set has size $n - 1$.

As a result of these differences, approximating VERTEX COVER and finding a maximal independent set are typically treated as different tasks. Many research papers on finding maximal independent sets do not even mention VERTEX COVER [3, 11, 18, 23, 35, 42], while other papers that address both apply different techniques for each problem [27, 28]. One indirect relationship between these two problems is that any maximal independent set algorithm can be used as a subroutine for approximating VERTEX COVER. If the goal is to approximate VERTEX COVER on a graph $G = (V, E)$, one can first run a MIS algorithm on the *line graph* of G . This produces a maximal matching in G , which can be combined with Algorithm 1 to obtain a 2-approximate vertex cover. However, an arbitrary maximal independent set in G provides no guarantees for the VERTEX COVER objective in G . This can be seen by again considering the maximal independent set consisting of the center node in a star graph.

2.3 Correlation Clustering and Edge-Deletion Objectives

Our work builds on connections between VERTEX COVER and CORRELATION CLUSTERING [6], which is the problem of partitioning an unweighted and undirected graph $G = (V, E)$ into an arbitrary number of clusters in a way that minimizes the number of *mistakes*. There are two types of mistakes: a *positive mistake* is when a pair of adjacent nodes is separated into different clusters, and a *negative mistake* is when two non-adjacent nodes are placed in the same cluster. The problem

Algorithm 5 NEIGHBORCOVER(G)

```
 $\mathcal{C} \leftarrow \emptyset, \mathcal{I} \leftarrow \emptyset, \mathcal{U} \leftarrow V.$ 
while  $\mathcal{U} \neq \emptyset$  do
    Randomly select  $u \in \mathcal{U}$  proportional to  $w_u$ 
     $\mathcal{I} \leftarrow \mathcal{I} \cup \{u\}$ 
5:   for  $v \in N(u) \cap \mathcal{U}$  do
        $\mathcal{C} \leftarrow \mathcal{C} \cup \{v\}$ 
        $\mathcal{U} \leftarrow \mathcal{U} \setminus \{v\}$ 
Return  $\mathcal{C}$ 
```

is NP-hard but many approximation algorithms have been developed [2, 6, 9, 16, 17, 12, 50]. One of the simplest and fastest algorithms is a randomized 3-approximation commonly known as PIVOT, which iteratively selects an unclustered node uniformly at random (the *pivot*) and clusters it with all its unclustered neighbors [2]. This is closely related to Algorithm 4 in that the pivot nodes form a random greedy maximal independent set. This relationship has also been noted in previous work [9, 23].

We also draw on connections between CORRELATION CLUSTERING and an NP-hard edge-labeling problem called *minimum strong triadic closure labeling with edge insertions* (MINSTC+), which is known to be reducible to a special type of hypergraph VERTEX COVER problem [30, 31, 43, 49]. Recent work showed how to use VERTEX COVER algorithms as subroutines for CORRELATION CLUSTERING approximation algorithms [50], though this did not involve new algorithms for the general VERTEX COVER problem. Section 4 expands on these connections between clustering, edge-labeling, and MIS algorithms, and how they relate to our new approximation algorithm for VERTEX COVER. Finally, our algorithmic techniques lead to new approximation algorithms for multiple edge-deletion problems in graphs and hypergraphs, including a recent objective for clustering edge-colored hypergraphs [4, 50] and a path-deletion problem in directed acyclic graphs [36]. We cover formal definitions and additional background as needed for these objectives in Section 5.

3 The Maximal Independent Set Algorithm for Vertex Cover

Our main result is a simple algorithm that simultaneously grows a maximal independent set and builds a 2-approximate vertex cover. This algorithm can be seen as a special type of weighted generalization of GREEDYMIS (Algorithm 4). Our proof that this is a 2-approximation for VERTEX COVER is closely related to the proof that PIVOT is a 3-approximation algorithm for CORRELATION CLUSTERING [2]. We discuss the relationship between these algorithms in more depth in Section 4.

3.1 Overview and Approximation Guarantee

Our algorithm for VERTEX COVER (Algorithm 5) iteratively grows a cover set \mathcal{C} and an independent set \mathcal{I} . During the course of the algorithm, every node that has not yet been added to \mathcal{I} or \mathcal{C} is in an *undecided* node set \mathcal{U} . At each iteration, the algorithm randomly chooses a node $v \in \mathcal{U}$ proportional to its node weight w_v . That node v is added to set \mathcal{I} , and all of its undecided neighbors are added to the vertex cover. The algorithm terminates when all nodes are either in \mathcal{C} or \mathcal{I} . By design, \mathcal{I} is guaranteed to be a maximal independent set and \mathcal{C} is a vertex cover. We refer to this algorithm as NEIGHBORCOVER.

Theorem 3.1. NEIGHBORCOVER is a randomized 2-approximation algorithm for the minimum weighted VERTEX-COVER problem.

Proof. The linear programming relaxation for VERTEX COVER is given by

$$\begin{aligned} \min \quad & \sum_{v \in V} w_v x_v \\ \text{s.t.} \quad & x_u + x_v \geq 1 \text{ for } (u, v) \in E \\ & x_v \geq 0 \text{ for } v \in V \end{aligned} \tag{2}$$

where there is variable x_v for each node $v \in V$ and a constraint for each edge. The dual of this relaxation the following linear program:

$$\begin{aligned} \max \quad & \sum_{e \in E} y_e \\ \text{s.t. for each } u \in V: \quad & \sum_{e: u \in e} y_e \leq w_u \\ & y_e \geq 0 \text{ for } e \in E. \end{aligned} \tag{3}$$

When $w_v = 1$ for every $v \in E$, the solution to the dual linear program is the largest fractional edge matching. By LP duality theory, every feasible solution to the dual LP is a lower bound for the VERTEX COVER instance. We will show how to construct a feasible solution whose value is half the expected cost of NEIGHBORCOVER, proving the 2-approximation.

Expected cost of the algorithm. If $v \in V$ is added to \mathcal{I} by NEIGHBORCOVER, we will refer to it as a *MIS-node*. If a node $u \in V$ is never chosen as a MIS-node, this means that the algorithm eventually places u in the cover \mathcal{C} , incurring a cost of w_u . This means that some node v adjacent to u was chosen as a MIS-node in some iteration, so we will charge the cost w_u to the edge $(u, v) \in E$. For an edge $e \in E$, let A_e denote the event that one of the two nodes in e is chosen as a MIS-node in an iteration where both are still undecided, and let $p_e = \mathbb{P}[A_e]$. An edge $e = (u, v)$ receives a charge if and only if A_e occurs, and it can only receive a charge once. Conditioned on A_e being true, the charge assigned to e depends on whether u or v is chosen as a MIS-node. With probability $w_u/(w_u + w_v)$, node u is chosen as a MIS-node, meaning that node v is placed in the vertex cover and e is charged cost w_v . With probability $w_v/(w_u + w_v)$, u is placed in the vertex cover and the charge is w_u . If we let X_e be a random variable denoting the charge to edge e , then $C = \sum_{e \in E} X_e$ is the total cost incurred by NEIGHBORCOVER and has the following expected value:

$$\begin{aligned} \mathbb{E}[C] &= \sum_{e \in E} \mathbb{E}[X_e] = \sum_{e \in E} \mathbb{E}[X_e \mid A_e] \mathbb{P}[A_e] \\ &= \sum_{e=(u,v) \in E} \left(w_v \cdot \frac{w_u}{w_u + w_v} + w_u \cdot \frac{w_v}{w_u + w_v} \right) p_e = \sum_{e=(u,v) \in E} \frac{2w_u w_v}{w_u + w_v} p_e. \end{aligned}$$

Lower bound. For a node $u \in V$, let B_u be the event that node u enters the vertex cover \mathcal{C} at some point during the algorithm. For every edge $e = (u, v)$, we have

$$\mathbb{P}[B_u \wedge A_e] = \mathbb{P}[B_u \mid A_e] \cdot \mathbb{P}[A_e] = \frac{w_v}{w_u + w_v} \cdot p_e. \tag{4}$$

Observe now that the node cost w_u can be charged to only one edge (v, u) incident to u . This means that for two different edges e and f that share node u , the events $B_u \wedge A_e$ and $B_u \wedge A_f$ are disjoint, and more generally we know that for an arbitrary node $u \in V$,

$$\sum_{v: e=(u,v) \in E} \frac{w_v}{w_u + w_v} p_e = \sum_{e: u \in e} \mathbb{P}[B_u \wedge A_e] \leq 1. \tag{5}$$

For each $e = (u, v) \in E$, define a variable $\hat{y}_e = \frac{w_v w_u}{w_u + w_v} p_e$. By (5), for every $u \in V$ we have

$$\sum_{e: u \in e} \hat{y}_e = \sum_{v: e=(u,v) \in E} \frac{w_v w_u}{w_u + w_v} p_e \leq w_u, \quad (6)$$

so the variables $\{\hat{y}_e\}_{e \in E}$ satisfy the constraints of the dual LP (3) and we can see that half the expected cost of the algorithm is a lower bound on the VERTEX-COVER instance:

$$\sum_{e \in E} \hat{y}_e = \sum_{(u,v) \in E} \frac{w_v w_u}{w_u + w_v} p_e = \frac{1}{2} \mathbb{E}[C]. \quad (7)$$

□

This result immediately implies that GREEDYMIS is a 2-approximation for unweighted VERTEX COVER. Theorem 3.1 can also be viewed as an improved theoretical result for list heuristic algorithms for VERTEX COVER [5, 19]. In particular, the LISTRIGHT algorithm [19] is nearly identical to GREEDYMIS, and only differs in that the node ordering is given rather than chosen uniformly at random. The best previous approximation factor for this method is $\frac{\sqrt{\Delta}}{2} + \frac{3}{2}$, where Δ is the maximum degree, which is obtained by ordering vertices by degree. Our result shows that a random ordering provides an expected 2-approximation. We summarize these observations as a corollary.

Corollary 3.2. *GREEDYMIS (Algorithm 4) is equivalent to applying LISTRIGHT [19] with a uniform random node ordering, and is a randomized 2-approximation for unweighted VERTEX COVER.*

3.2 Runtime Guarantees and Implementation

When the graph $G = (V, E)$ is unweighted, NEIGHBORCOVER can be implemented by first generating a uniform random permutation to determine the order in which to visit nodes. If the i th node that is visited is undecided, it is added to the independent set, otherwise it is a vertex cover node and the algorithm continues to the next step. The random permutation can be generated in $O(|V|)$ time (e.g., using the Fisher-Yates shuffle), so the runtime for the unweighted version is $O(|E|)$.

If we assume the nodes have arbitrary nonnegative weights, the implementation and runtime analysis is made more challenging by the node sampling procedure. In particular, sampling a node based on its weight from among all undecided nodes is more involved than the random sampling procedure in Pitt's algorithm (Algorithm 2), which only requires sampling one of two nodes in an edge. A naive sampling procedure would take $O(|V|)$ time each round, which would lead to an overall runtime of $O(|V|^2 + |E|)$, since we must sample a node in each of $O(|V|)$ iterations. With a more careful implementation we can achieve a runtime of $O(|V| \log |V| + |E|)$. One simple way to achieve this is to use the implementation in Algorithm 7, which decouples the random sampling strategy from the procedure of growing a maximal independent set. Algorithm 6 is used to generate a permutation of *all* nodes based on their weights, and can be implemented in $O(|V| \log |V|)$ time [52]. In iteration i , Algorithm 7 may visit a node that is decided already, but in this case the node will simply be ignored, so that the selection of the next independent set node follows the same sampling distribution.

The $O(|V| \log |V|)$ term in the runtime is one disadvantage of NEIGHBORCOVER relative to edge-visiting algorithms that run in linear time even in the weighted case (e.g., Algorithms 2 and 3). Nevertheless, the runtime is still $O(|E|)$ when the graph is unweighted and whenever $|E| = \Omega(|V| \log |V|)$. As we shall see later, one advantage of NEIGHBORCOVER is that it leads to several particularly simple approximation algorithms for certain edge-deletion problems that can be

Algorithm 6 WEIGHTEDSHUFFLE($\{w_1, w_2, \dots, w_n\}$)

```
U = {1, 2, ..., n}
for i = 1 to n do
    Sample t ∈ U proportional to w_t
    σ(i) = v; U ← U \ {t}
Return σ
```

Algorithm 7 NEIGHBORCOVER(G)

```
C ← ∅, I ← ∅
σ = WEIGHTEDSHUFFLE({w_1, w_2, ..., w_n})
for v = v_{σ(1)}, v_{σ(2)}, ..., v_{σ(n)} do
    // Check if v has to be covered
    for u ∈ N(v) do
        if u ∈ I then
            C ← C ∪ {v}
            break
    // If not, add v to MIS
    if v ∉ C then
        I ← I ∪ {v}
Return C
```

reduced to VERTEX COVER in an approximation preserving way. These implicit implementations can easily be made more efficient than applying a naive approach that relies on explicitly forming the reduced instance of VERTEX COVER.

4 Algorithm Equivalence Results

Theorem 3.1 shows for the first time that GREEDYMIS (Algorithm 4) is an expected 2-approximation for unweighted VERTEX COVER, and provides a principled new way to generalize this method to node-weighted graphs. Our method is also related to CORRELATION CLUSTERING [6] and strong triadic closure edge-labeling problems [49]. In particular, the proof of Theorem 3.1 is inspired by the analysis of the 3-approximate PIVOT algorithm for CORRELATION CLUSTERING [2]. In this section we highlight two separate ways in which NEIGHBORCOVER and PIVOT are related. We also discuss a simple existing parallelization scheme for GREEDYMIS which, based on our equivalence results, can be viewed as an approximation algorithm for several different problems at once.

Several of the connections and equivalences highlighted in this section are already present in some form in previous literature, though not all in one place. We bring these connections together to highlight how our new approximation algorithm for VERTEX COVER relates to algorithmic techniques for other problems. These connections also lay the groundwork for several open directions for future research that we discuss at the end of the paper.

4.1 Correlation Clustering and 3-uniform Hypergraph Vertex Cover

Many approximation algorithms of CORRELATION CLUSTERING are based on counting open wedges (also called *bad triangles* or *bad triplets*) [2, 6, 9, 50]. An open wedge in G is a set of three nodes whose induced subgraph contains only two edges. Every way of clustering these nodes leads to at

least one disagreement: either all nodes will be placed in the same cluster (producing a negative mistake) or two adjacent nodes will be separated (producing a positive mistake). Letting \mathcal{W} denote the set of open wedges in G , the following binary linear program provides a lower bound for the optimal CORRELATION CLUSTERING objective:

$$\begin{aligned} \min \quad & \sum_{\{i,j\} \in \binom{V}{2}} x_{ij} \\ \text{s.t.} \quad & x_{ij} + x_{ik} + x_{jk} \geq 1 \text{ for } \{i,j,k\} \in \mathcal{W} \\ & x_{ij} \in \{0,1\} \text{ for } \{i,j\} \in \binom{V}{2}. \end{aligned} \tag{8}$$

The constraint $x_{ij} + x_{ik} + x_{jk} \geq 1$ reflects that fact that there will be at least one mistake among the node pairs in the open wedge $\{i,j,k\}$. There is a close relationship between this binary program and the binary program for VERTEX COVER in (1). Instead of variables for nodes, there is a variable for each node pair, and instead of the constraint $x_u + x_v \geq 1$ we have $x_{ij} + x_{ik} + x_{jk} \geq 1$. Problem (8) in fact encodes a VERTEX COVER problem in a 3-uniform *open wedge hypergraph* $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ constructed from the original graph $G = (V, E)$ as follows:

- For each node pair $\{i,j\} \in \binom{V}{2}$, define an edge $v_{ij} \in \mathcal{V}$.
- For each open wedge $\{i,j,k\} \subseteq \mathcal{W}$, define a hyperedge $\{v_{ij}, v_{ik}, v_{jk}\} \in \mathcal{E}$.

Figure 2 provides an illustration of this reduction. Every clustering of nodes in $G = (V, E)$ can be mapped to a vertex cover in \mathcal{H} : if the clustering makes a mistake at node pair $\{i,j\}$, this means node v_{ij} is covered in the hypergraph. However, the reverse is not necessarily true, and it is not hard to come up with simple examples where a vertex cover in \mathcal{H} does not translate to a node clustering in G (see Figure 2).

This special 3-uniform VERTEX COVER problem is equivalent to an NP-hard edge *labeling* problem that is based on the principle of *strong triadic closure* [49, 50]. Strong triadic closure (STC) posits that two individuals in a social network will share at least a weak connection to one another if they both share strong ties to a mutual friend (see chapter 3 in [21]). This principle can be related back to open wedges in the graph $G = (V, E)$. If $\{u, v, w\} \in \mathcal{W}$ and $(v, w) \notin E$, this means that v and w have a mutual “friend” (node u) though they do not share an edge. Strong triadic closure suggests that one of the following must be true: (1) (u, v) is a weak tie, (2) (u, w) is a weak tie, or (3) nodes v and w actually do share at least a weak tie but the graph G simply has a “missing” edge. An STC+ labeling¹ for graph $G = (V, E)$ is defined to be a set of edges $E_W \subseteq E$ to label as *weak* along with a set of node pairs $E_N \subseteq \binom{V}{2} - E$ to turn into new *weak* edges, in order to ensure that strong triadic closure holds. In other words, for an open wedge $\{u, v, w\}$ centered at u , either (u, v) or (u, w) is labeled as weak, or the non-adjacent pair $\{v, w\}$ is added to E_N . The MINSTC+ problem is the task of finding an STC+ labeling that minimizes $|E_N| + |E_W|$. The equivalence between MINSTC+ and a special type of 3-uniform VERTEX COVER problem was noted when this edge-labeling problem was first introduced [49]. Further connections between MINSTC+ and CORRELATION CLUSTERING were explored in subsequent work [30, 31, 43, 50]. These connections provide the foundation for understanding the relationship between NEIGHBORCOVER and the PIVOT approximation algorithm for CORRELATION CLUSTERING.

4.2 Pivot as a Hypergraph Vertex Cover Algorithm

The PIVOT algorithm for CORRELATION CLUSTERING selects an unclustered node uniformly at random (the *pivot* node) in each iteration, and clusters it with all its unclustered neighbors. This is

¹The ‘+’ in STC+ indicates that edge additions are allowed. Sintos and Tsasparas [49] also considered a version that only involved labeling existing edges as weak or strong.

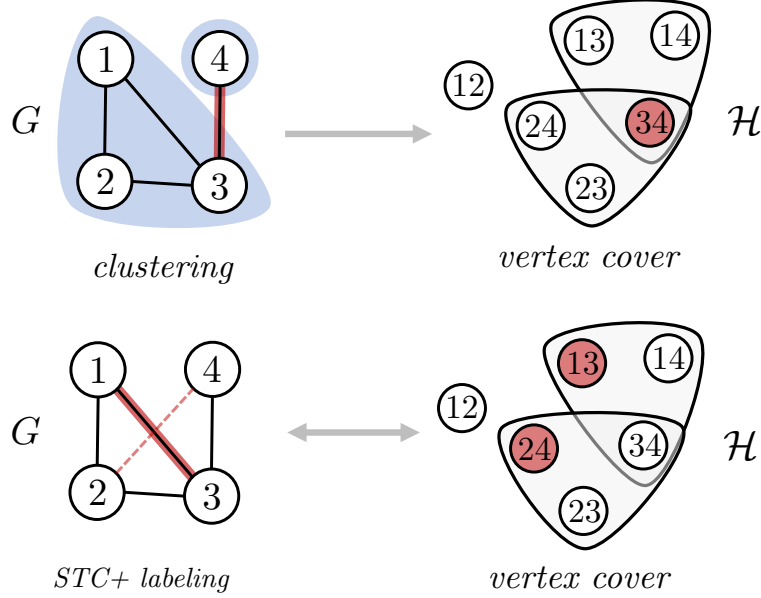


Figure 2: The open wedge hypergraph \mathcal{H} of a graph $G = (V, E)$ is obtained by introducing a node $\langle ij \rangle$ for each node pair $\{i, j\}$ in G . (Top row) A clustering of the nodes in G always maps to a vertex cover in \mathcal{H} . In the example, the only “mistake” is to separate nodes 3 and 4, hence node $\langle 34 \rangle$ is a vertex cover in \mathcal{H} . (Bottom row) Vertex covers in \mathcal{H} do not always correspond to clusterings, but are in one-to-one correspondence with STC+ labelings. Covering nodes $\langle 13 \rangle$ and $\langle 24 \rangle$ means labeling $(1, 3)$ as *weak* and introducing a new (*weak*) edge $(2, 4)$ in G . All open wedges in G now satisfy strong triadic closure (at least one edge in each open wedge is *weak*).

repeated until all nodes are clustered. Ailon, Charikar, and Newman [2] proved that this algorithm provides a 3-approximation for CORRELATION CLUSTERING by considering the linear programming relaxation of objective (8). These authors showed that the expected cost of PIVOT can be bounded below by constructing an implicit feasible solution for the dual linear program, which encodes the notion of a fractional open wedge packing. An open wedge packing is a node-pair-disjoint set of open wedges in G , which provides a lower bound for CORRELATION CLUSTERING since at least one mistake must be made at each disjoint open wedge. The dual LP encodes *fractional* packings in the sense that each node pair is allowed to partially contribute to multiple open wedges as long as the sum of contributions is at most 1.

The connection between CORRELATION CLUSTERING and 3-uniform VERTEX COVER was not explicitly noted in the work of Ailon, Charikar, and Newman [2], but this relationship sheds light on why the analysis for PIVOT can be adapted to prove NEIGHBORCOVER is a 2-approximation for VERTEX COVER. In particular, the fractional open wedge packing that PIVOT relies on corresponds to a fractional matching in the open wedge hypergraph, just as NEIGHBORCOVER relies on a fractional matching lower bound in a graph. We formalize the relationship with a simple lemma that follows quickly from previous observations, but has not been explicitly noted elsewhere in the literature.

Lemma 4.1. *PIVOT is a 3-approximation algorithm for MINSTC+. Equivalently, PIVOT is a 3-approximation algorithm for the problem of finding a minimum vertex cover in the open wedge hypergraph of a graph.*

Proof. The original analysis of PIVOT [2] shows that the expected cost of this algorithm is at most 3

times the optimal solution value of the linear programming relaxation of objective (8). This linear program lower bounds MINSTC+ in addition to lower bounding CORRELATION CLUSTERING. Because every clustering of G also maps to a vertex cover in its open wedge hypergraph (i.e., a valid STC+ labeling), PIVOT returns an edge-labeling that is a 3-approximation for MINSTC+. \square

The fact that PIVOT is a 3-approximation for both CORRELATION CLUSTERING and MINSTC+ is somewhat surprising given the difference between these problems. As mentioned previously, every clustering of G can be mapped to a vertex cover in the open wedge hypergraph \mathcal{H} , but the reverse statement is not true. It was recently shown that any α -approximation for MINSTC+ can be used to design a (2α) -approximation for CORRELATION CLUSTERING [50]. This procedure starts with an α -approximate vertex cover in \mathcal{H} and then applies a rounding step that distorts the approximation by a factor of 2 in order to convert the vertex cover in \mathcal{H} into a clustering in G . With this result in hand, one can also prove that any α -approximation for CORRELATION CLUSTERING can provide a (2α) -approximation for MINSTC+. However, Lemma 4.1 indicates that PIVOT is able to overcome this factor 2 difference.

While Lemma 4.1 provides insight into one relationship between PIVOT and NEIGHBORCOVER, there are still a few key differences between how these algorithms apply to VERTEX COVER problems. First of all, PIVOT applies to a very specific type of 3-uniform hypergraph, and even then only implicitly. Its analysis provides no guarantees for the general 3-uniform hypergraph VERTEX COVER problem, while NEIGHBORCOVER applies to all graph VERTEX COVER problems. Secondly, NEIGHBORCOVER applies to node-weighted VERTEX COVER, whereas PIVOT does not apply to weighted CORRELATION CLUSTERING. Using a weighted shuffling procedure such as Algorithm 6 to choose pivot nodes does not make sense in the context of CORRELATION CLUSTERING, since weighted versions of CORRELATION CLUSTERING involve *edge* weights and not node weights. Finally, perhaps the most interesting difference is that a single iteration of PIVOT (implicitly) adds multiple nodes in \mathcal{H} to an independent set. This is because clustering a pivot node $v \in V$ with its neighbors in G means *not* making a mistake at all node pairs involving v . In other words, the nodes in \mathcal{H} corresponding to multiple node pairs in G will not be added to the implicit vertex cover. As an example, the clustering of graph G in Figure 2 can be obtained by selecting nodes 2 and 4 as pivots, in that order. When 2 is selected as a pivot, all nodes in \mathcal{H} other than the node corresponding to edge $(3, 4) \in E$ are added to an independent set in \mathcal{H} . In contrast, NEIGHBORCOVER adds a single node to an independent set in each iteration.

4.3 Equivalence among Pivot, GreedyMIS, and NeighborCover

Although NEIGHBORCOVER grows an independent set in G in a different way than PIVOT grows an independent set in the open wedge hypergraph of G , this is essentially because the algorithms are actually very similar in a different regard. Namely, they both operate *on the graph G* by selecting a random undecided node in each iteration and making a decision about how to deal with that node's undecided neighbors. Here, *undecided* either means *unclustered* (in the case of PIVOT) or *not assigned to a cover or independent set* (in the case of NEIGHBORCOVER). Pseudocode for PIVOT is given in Algorithm 8, written in a way that best highlights its close relationship to the unweighted version of NEIGHBORCOVER, i.e., GREEDYMIS. In particular, the set of pivot nodes defining its clusters exactly corresponds to a maximal independent set grown from a uniform random ordering of nodes. This relationship between PIVOT and GREEDYMIS has already been noted in previous work [9, 23]. Combining this observation with Theorem 3.1 and Lemma 4.1 leads to the following simple corollary.

Algorithm 8 PIVOT(G)

```
 $\mathcal{U} = V$  // unclustered node set
For  $v \in V$ ,  $c[v] = 0$  // initialize cluster indicator vector
 $clus = 1$  // current cluster index
Generate random uniform node permutation  $\sigma$ 
5: for  $v = v_{\sigma(1)}, v_{\sigma(2)}, \dots, v_{\sigma(n)}$  do
    if  $v \in \mathcal{U}$  then
         $c[v] = clus$ 
         $\mathcal{U} \leftarrow \mathcal{U} \setminus \{v\}$ 
        for  $u \in N(v) \cap \mathcal{U}$  do
10:          $c[u] = clus$ 
         $\mathcal{U} \leftarrow \mathcal{U} \setminus \{u\}$ 
         $clus \leftarrow clus + 1$ 
Return  $c$ 
```

Corollary 4.2. *Running GREEDYMIS on a graph G simultaneously produces a maximal independent set in G , an expected 2-approximate VERTEX COVER for G , and pivot nodes for an expected 3-approximation for CORRELATION CLUSTERING and 3-approximation for MINSTC+ on G .*

One interesting consequence of this corollary is that a parallel variant of GREEDYMIS also directly provides a simple parallel approximation algorithm for VERTEX COVER. This parallel variant generates a uniform random ordering of nodes, and in each round, all nodes that come before their neighbors in the ordering are added to the independent set. These nodes and their neighbors are removed from the graph, and the algorithm repeats this procedure in rounds until no nodes are left [11, 23]. For a fixed ordering of nodes, this returns the same output as the sequential GREEDYMIS algorithm, and with high probability the algorithm requires only $O(\log n)$ rounds [23] before termination. Our equivalence result implies this is a parallel $O(\log n)$ -round 2-approximation for VERTEX COVER as well.

5 Fast and Simple Algorithms for Edge-Deletion Problems

NEIGHBORCOVER can be used to design fast and simple approximation algorithms for combinatorial problems that can be reduced to VERTEX COVER. In particular, we consider certain edge-deletion problems whose reduction to VERTEX COVER leads to a graph with a very special edge structure. By implicitly implementing NEIGHBORCOVER and taking advantage of this special edge structure, we can design methods that are significantly faster than forming the VERTEX COVER instance explicitly and applying a linear-time VERTEX COVER algorithm as a black-box.

For the problems we consider, a careful implicit implementation of edge-visiting algorithms for VERTEX COVER (e.g., Algorithms 2 and 3) can also lead to improvements over forming the reduced graph explicitly. However, one advantage of NEIGHBORCOVER is that it is often simpler to implicitly iterate through the nodes of the reduced graph in an efficient way than to implicitly iterate through the edges. If *all* edges in the reduced graph are visited implicitly, this leads to the same runtime issues as explicitly forming the VERTEX COVER instance, so one must carefully reason about edges that can be safely skipped. We avoid this issue altogether when implicitly implementing NEIGHBORCOVER, which iterates over nodes in the reduced graph.

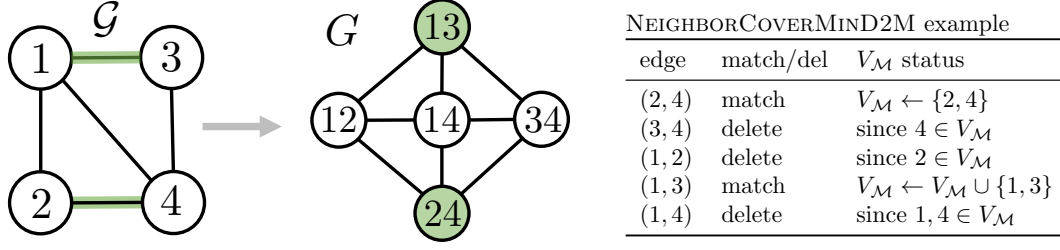


Figure 3: A small example of converting an (unweighted) MIND2M problem on a graph \mathcal{G} into a VERTEX COVER problem on graph G and implicitly applying NEIGHBORCOVER for the given ordering of edges in \mathcal{G} . Maximal independent set nodes are in green, and correspond to a matching in \mathcal{G} . An implicit implementation of NEIGHBORCOVER does not need to check all edges in G , but can tell when an edge (i, j) in \mathcal{G} (i.e., node in G) is ineligible to be added to a matching in \mathcal{G} (i.e., independent set in G) based on whether nodes i and j are already in the matching.

Algorithm 9 NEIGHBORCOVERMIND2M(G)

Input: edge-weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
Output: 2-approximate solution to MIND2M
 $\mathcal{D} \leftarrow \emptyset$ // edges to delete
 $\mathcal{M} \leftarrow \emptyset$ // edges in matching
 $\mathcal{V}_M = \emptyset$ // nodes in the matched edges
 $\sigma = \text{WEIGHTEDSHUFFLE}(\{\omega_1, \omega_2, \dots, \omega_{|\mathcal{E}|}\})$
for $e = e_{\sigma(1)}, e_{\sigma(2)}, \dots, e_{\sigma(|\mathcal{E}|)}$ **do**
 $e = (i, j)$ // identify nodes in e
 if $i \in \mathcal{V}_M$ or $j \in \mathcal{V}_M$ **then**
 $\mathcal{D} \leftarrow \mathcal{D} \cup \{e\}$
 else
 $\mathcal{M} \leftarrow \mathcal{M} \cup \{e\}$
 $\mathcal{V}_M \leftarrow \mathcal{V}_M \cup \{i, j\}$
Return \mathcal{D}

5.1 Minimum Delete-to-Matching

As an illustrative warm-up, we consider a simple edge-deletion problem where the goal is to find a minimum weight set of edges in an edge-weighted undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to delete in order to convert \mathcal{G} into a matching. Let $e_i \in \mathcal{E}$ denote the i th edge (for an arbitrary ordering of edges), and let $\omega_i \geq 0$ be its weight. We refer to this as *minimum delete-to-matching*, or simply MIND2M. This problem can be optimally solved in polynomial time by computing a maximum matching and deleting all edges not in the matching. We will illustrate how to obtain a much faster 2-approximation algorithm using an implicit implementation of NEIGHBORCOVER, which amounts to finding a maximal matching in the edge-weighted graph \mathcal{G} using a specific edge-sampling strategy.

The MIND2M objective on \mathcal{G} is equivalent to solving VERTEX COVER on the line graph $G = (V, E)$ of \mathcal{G} : a node in \mathcal{G} is an edge in G , and a maximal independent set in \mathcal{G} is a maximal matching in G . Explicitly forming the line graph of \mathcal{G} and applying any linear-time VERTEX COVER algorithm as a black-box yields a 2-approximation algorithm for MIND2M.² This basic

²One must be careful to distinguish this from a reduction that has often been applied other direction, namely, approximating *unweighted* VERTEX COVER by first obtaining a maximal matching, which itself can be found by

approach takes $O(|\mathcal{E}|^2)$ time, as this is a bound on the number of edges in G . Even if we avoid forming G explicitly, VERTEX COVER algorithms that iterate through all of the edges in G will take $O(|\mathcal{E}|^2)$ time, even if they only implicitly visit edges in G by iterating through pairs of adjacent edges in \mathcal{G} . In contrast, Algorithm 9 is an implicit implementation of NEIGHBORCOVER applied to MIND2M that takes $O(|\mathcal{E}| \log |\mathcal{E}|)$ time when \mathcal{G} has arbitrary edge weights, and $O(|\mathcal{E}|)$ time in the unweighted case. At each iteration, the MIS algorithm must check whether a node in G (i.e., an edge in \mathcal{G}) can be added to an independent node set in G (i.e., a matching \mathcal{M} in \mathcal{G}). The key to a fast implementation is realizing that we can quickly see if an edge can be added to \mathcal{M} by checking whether either of its nodes already belongs to an edge in \mathcal{M} . Selecting a random permutation of edges takes $O(|\mathcal{E}| \log |\mathcal{E}|)$ in the weighted case or only $O(|\mathcal{E}|)$ in the unweighted case. The rest of the algorithm takes $O(|\mathcal{E}|)$ time, since each iteration just involves visiting an edge $(i, j) \in \mathcal{E}$, checking if i or j already belongs to a matched node set $\mathcal{V}_{\mathcal{M}}$, and then either adding (i, j) to the matching or deleting it. See Figure 3 for an illustration of running Algorithm 9. To summarize, keeping track of one additional fact about each node in \mathcal{G} (specifically, whether or not it is in a set of matched nodes $\mathcal{V}_{\mathcal{M}}$) is sufficient to avoid iterating through all edges in the reduced VERTEX COVER instance.

5.2 DAG Edge Deletion

Let $D = (\mathcal{V}, A)$ be an edge-weighted directed acyclic graph where $e_i \in A$ is the i th directed edge and $\omega_i \geq 0$ is its weight. The DAG EDGE DELETION problem with parameter k , or simply DED- k , seeks a minimum weight set of edges to remove in order to destroy all paths of length k in D . The problem was first considered by Kendre et al. [36] as the minimization version of the MAX- k -ORDERING problem. It is the edge-deletion version of the DAG VERTEX DELETION problem [44].

We focus on DED-2 specifically. This can be reduced in an approximation preserving way to an instance of VERTEX COVER on a graph $G = (V, E)$ by replacing each directed edge e in D with a node v_e in G , and by adding an edge between two nodes v_e and v_f in G when the edges $\{e, f\}$ define a directed path in D . DED-2 is known to be NP-hard, as can be observed from its equivalence (at optimality) with the maximum directed cut problem [38, 41]. Kendre et al. [36] presented two combinatorial 2-approximation algorithms for this problem, one for unweighted graphs and another for weighted graphs. For our purposes it is interesting to note that the unweighted algorithm corresponds to an implicit implementation of the maximal matching method (Algorithm 1)—while there exists a directed 2-path in the graph, find it and delete both edges. The weighted algorithm is similarly an implicit implementation of the local ratio method (Algorithm 3). Kendre et al. [36] confirmed that the algorithms run in polynomial time, but did not provide any strategies for quickly finding paths in the directed graph that need to be covered. The number of length two paths can be significantly larger than $O(|A|)$, so iterating through all of these paths can be much worse than linear-time in terms of the size of D , even in the unweighted case.

Algorithm 10 is an implicit implementation of NEIGHBORCOVER that gives a 2-approximation for DED-2 in time $O(|A|)$ for unweighted graphs and $O(|A| \log |A|)$ time for the weighted case. This method builds an independent set and a vertex cover in G implicitly by building a set \mathcal{K} of edges to keep and a set \mathcal{D} of edges to delete in the acyclic graph D . The algorithm searches through directed edges in D (i.e., nodes in G), in search of edges that can be added to the set \mathcal{K} without creating 2-paths. Similar to our observations for MIND2M, there is an easy way to check whether an edge is “allowed” to be added to \mathcal{K} . Given an edge (i, j) where i is the tail node and j is the head node, we know we can add (i, j) to \mathcal{K} as long as there is currently no edge in \mathcal{K} where i is the

running a MIS algorithm on the line graph. Here, instead of using an *unweighted* maximal matching algorithm to approximate *unweighted* VERTEX COVER, we are using a *weighted* VERTEX COVER approximation algorithm to approximate (the minimization version of) a *weighted* matching problem.

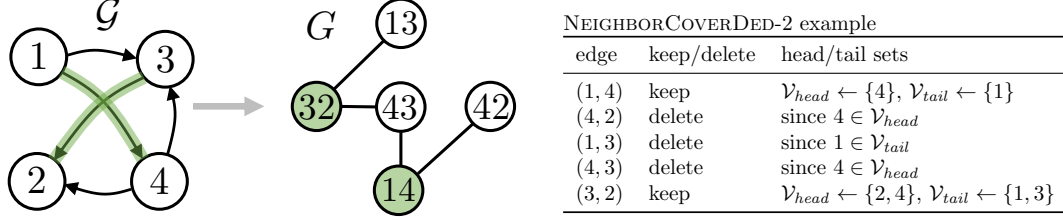


Figure 4: A small example of converting an instance of DED-2 on \mathcal{G} into a VERTEX COVER problem on graph G and implicitly applying NEIGHBORCOVER for the given ordering of edges in \mathcal{G} (nodes in G). The algorithm does not need to check all edges in G . It takes constant time to check whether endpoints of a directed edge (i, j) are in \mathcal{V}_{head} and \mathcal{V}_{tail} , and this is sufficient to know whether node (ij) can be added to an independent set in G .

Algorithm 10 NEIGHBORCOVERDED2(G)

Input: edge-weighted DAG $D = (\mathcal{V}, A)$
Output: 2-approximate solution to DED-2
 $\mathcal{D} \leftarrow \emptyset$ // edges to delete
 $\mathcal{K} \leftarrow \emptyset$ // edges to keep
 $\mathcal{V}_{head} = \emptyset$ // head nodes for edges in \mathcal{K}
 $\mathcal{V}_{tail} = \emptyset$ // tail nodes for edges in \mathcal{K}
 $\sigma = \text{WEIGHTEDSHUFFLE}(\{\omega_1, \omega_2, \dots, \omega_{|A|}\})$
for $e = e_{\sigma(1)}, e_{\sigma(2)}, \dots, e_{\sigma(|A|)}$ **do**
 $e = (i, j)$ // identify nodes in e
 if $i \in \mathcal{V}_{head}$ or $j \in \mathcal{V}_{tail}$ **then**
 $\mathcal{D} \leftarrow \mathcal{D} \cup \{e\}$
 else
 $\mathcal{K} \leftarrow \mathcal{K} \cup \{e\}$
 $\mathcal{V}_{tail} \leftarrow \mathcal{V}_{tail} \cup \{i\}$
 $\mathcal{V}_{head} \leftarrow \mathcal{V}_{head} \cup \{j\}$
Return \mathcal{D}

head or j is the tail. This can be checked in constant time in each iteration. See Figure 4 for an illustration of this process.

5.3 Edge-colored Hypergraph Clustering

We finally present a simple new approximation algorithm for the COLORED EDGE CLUSTERING problem in hypergraphs [4]. The input is a hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E}, \ell, k)$ where $w_e \geq 0$ denotes the weight of a hyperedge $e \in \mathcal{E}$ and $\ell: \mathcal{E} \rightarrow \{1, 2, \dots, k\}$ maps each hyperedge to one of k colors. The goal is to construct a node color label function $Y: \mathcal{V} \rightarrow \{1, 2, \dots, k\}$ that disagrees as little as possible with the hyperedge colors. We say a hyperedge e is *satisfied* if $Y[u] = \ell(e)$ for every $e \in u$. Formally, the goal is to minimize the weight of unsatisfied hyperedges. This is equivalent to deleting a minimum weight set of hyperedges so that remaining hyperedges of different colors never overlap. A node labeling can be viewed as a partitioning of nodes into k clusters where each cluster corresponds to one color. The problem is known to be APX-hard, but various approximation algorithms have been designed [4, 50].

The best approximation factors for COLORED EDGE CLUSTERING are based on linear program-

Algorithm 11 NEIGHBORCOVERCOLOREC(G)

Input: Edge-colored hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E}, \ell, k)$
Output: Node label function $Y: V \rightarrow \{1, 2, \dots, k\}$
 $\sigma = \text{WEIGHTEDSHUFFLE}(\{w_1, w_2, \dots, w_{|\mathcal{E}|}\})$
 For each $v \in V$ set $Y[v] = 0$
for $e = e_{\sigma(1)}, e_{\sigma(2)}, \dots, e_{\sigma(|\mathcal{E}|)}$ **do**
 if $Y[u] \in \{\ell(e), 0\}$ for every $u \in e$ **then**
 for $u \in e$ **do**
 $Y[u] = \ell(e)$
 Return Y

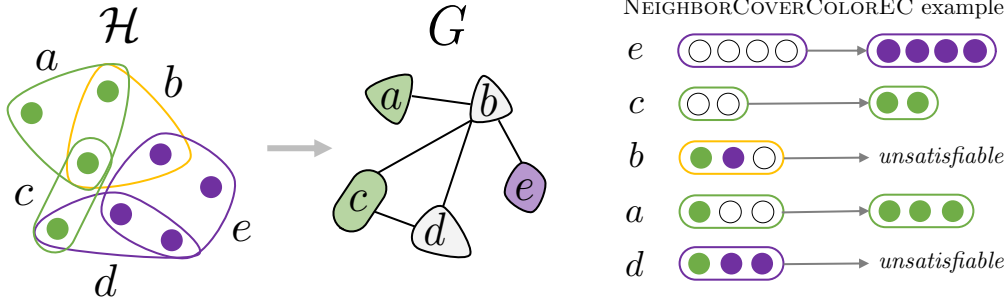


Figure 5: A small example of converting an edge-colored hypergraph \mathcal{H} into VERTEX COVER on a graph G and implicitly applying NEIGHBORCOVER to approximate COLORED EDGE CLUSTERING. Hyperedge colors are given as input; node colors in \mathcal{H} are the result of running the algorithm. Colored nodes in G indicate maximal independent set nodes, and are colored to match hyperedge colors in \mathcal{H} . Gray nodes in G are vertex cover nodes. Growing a maximal independent set in G is equivalent to visiting hyperedges at random and *satisfying* them when possible, meaning that all nodes are given the color of the hyperedge. If an edge is not satisfiable, the algorithm does nothing.

ming [4, 50], but faster 2-approximations are obtained by reducing COLORED EDGE CLUSTERING to VERTEX COVER [51]. For this reduction, each hyperedge $e \in \mathcal{E}$ corresponds to a node v_e with node-weight w_e in a new graph $G = (V, E)$. Two nodes in G share an edge if they correspond to hyperedges in \mathcal{H} that overlap and have different colors. A naive approach that explicitly iterates through all hyperedge pairs to form $G = (V, E)$, and then applies a black-box linear time VERTEX COVER algorithm will take $O(\sum_{v \in V} d_v^2 + |\mathcal{E}|^2)$ -time where d_v is the degree of node $v \in V$. However, an implicit implementation of Pitt’s VERTEX COVER algorithm leads to a 2-approximation with a runtime of $O(\sum_{e \in E} |e|)$ [51], which is linear in terms of the hypergraph size. We complement this result with a an even simpler randomized 2-approximation that corresponds to an implicit implementation of NEIGHBORCOVER (Algorithm 11).

Finding a maximal independent set in G is equivalent to finding a maximal set of satisfied hyperedges. Using the hyperedge-deletion view of the objective, an unsatisfied hyperedge is a hyperedge that must be deleted. Following the basic strategy of NEIGHBORCOVER, Algorithm 11 iterates through the hyperedges in \mathcal{H} (i.e., nodes in G) and greedily adds them to the satisfied set (i.e., an independent set in G). The only reason to not add a hyperedge e to the satisfied set is if an overlapping hyperedge of a different color (i.e., an adjacent node in G) was already satisfied in an earlier iteration. This means that at least one of the nodes $u \in e$ was already assigned a color $Y[u] \neq \ell(e)$. Therefore, in an iteration where we visit a hyperedge e , we simply need to check the

current color assignment for each node in e , and give all these node color $\ell(e)$ if possible. The fact that this algorithm is a 2-approximation is a corollary of Theorem 3.1.

Corollary 5.1. *Algorithm 11 is a randomized 2-approximation for COLORED EDGE CLUSTERING. Its runtime is $O(|\mathcal{E}| \log |\mathcal{E}| + \sum_{e \in \mathcal{E}} |e|)$ for weighted hypergraphs and $O(\sum_{e \in \mathcal{E}} |e|)$ for unweighted hypergraphs.*

For the runtime analysis, note that it takes $O(\sum_{e \in \mathcal{E}} |e|)$ time to iterate through all the edges looking for satisfiable edges. The additional $O(|\mathcal{E}| \log |\mathcal{E}|)$ term for the weighted case comes from applying Algorithm 6 to order edges.

6 Conclusions and Discussion

We have introduced a simple new approximation algorithm for VERTEX COVER and have discussed its connections to related previous algorithms for clustering nodes, labeling edges, and finding maximal independent sets. This method leads to fast and simple approximation algorithms for certain edge-deletion problems that can be reduced to VERTEX COVER in an approximation preserving way. One open direction is to explore other problems that are reducible to VERTEX COVER which might also benefit from implicit implementations. There are in fact examples where applying our method implicitly *does not* lead to runtime improvements over explicitly forming the reduced VERTEX COVER instance. One example is a version of the STC edge-labeling problem that does not allow edge additions [49]. This problem can be reduced to VERTEX COVER and provides a lower bound for a variant of CORRELATION CLUSTERING called CLUSTER DELETION [31, 49, 50]. We were unable to develop faster approximation algorithms for either problem using implicit implementations of NEIGHBORCOVER, as there does not appear to be a way to avoid iterating through all edges in the reduced VERTEX COVER instance.

One disadvantage of our method is that the weighted version involves an $O(|V| \log |V|)$ time node sampling step, whereas several previous algorithms for VERTEX COVER run in linear time even in the node-weighted case. An $O(|E|)$ -time implementation for the weighted version of our algorithm would be a useful improvement, though this seems challenging. Another advantage of some other VERTEX COVER algorithms is that they generalize easily to hypergraph VERTEX COVER. Although PIVOT can be viewed as a 3-approximation algorithm for VERTEX COVER in a very restrictive type of 3-uniform hypergraph, generalizing NEIGHBORCOVER to the general 3-uniform hypergraph VERTEX COVER problem remains open.

The connections highlighted in Section 4 suggest several other compelling directions for future research. Our equivalence results show that a simple parallel version of the greedy MIS algorithm also approximates VERTEX COVER. Although this is not the first parallel 2-approximation for VERTEX COVER, nor the best in terms of the number of rounds, it is particularly simple and has the attractive feature that it simultaneously solves multiple problems. There do exist $O(\log \log n)$ -round algorithms for finding a maximal independent set and for approximating VERTEX COVER [27], but a different approach is used for each problem. Furthermore, the first $O(\log \log n)$ -round $(2 + \varepsilon)$ -approximation for *weighted* VERTEX COVER was developed separately and required yet a different approach [28]. One interesting direction for future research is to explore whether the approximation guarantee for NEIGHBORCOVER, coupled with the fact that this algorithm applies to weighted VERTEX COVER, can be used to further simplify, unify, and improve existing parallel algorithms for MIS, VERTEX COVER, and CORRELATION CLUSTERING. Another open question is to see whether we can leverage weighted VERTEX COVER algorithms to develop faster approximation algorithms for weighted variants of CORRELATION CLUSTERING.

References

- [1] Faisal N Abu-Khzam, Michael A Langston, Pushkar Shanbhag, and Christopher T Symons. Scalable parallel algorithms for fpt problems. *Algorithmica*, 45(3):269–284, 2006. [1](#)
- [2] Nir Ailon, Moses Charikar, and Alanthan Newman. Aggregating inconsistent information: ranking and clustering. *Journal of the ACM (JACM)*, 55(5):1–27, 2008. [2](#), [5](#), [8](#), [10](#)
- [3] Noga Alon, László Babai, and Alon Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of algorithms*, 7(4):567–583, 1986. [4](#)
- [4] Ilya Amburg, Nate Veldt, and Austin Benson. Clustering in graphs and hypergraphs with categorical edge labels. In *Proceedings of The Web Conference 2020*, pages 706–717, 2020. [2](#), [5](#), [15](#), [16](#)
- [5] David Avis and Tomokazu Imamura. A list heuristic for vertex cover. *Operations research letters*, 35(2):201–204, 2007. [3](#), [4](#), [7](#)
- [6] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. *Machine Learning*, 56:89–113, 2004. [2](#), [4](#), [5](#), [8](#)
- [7] 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. [1](#), [3](#)
- [8] Reuven Bar-Yehuda and Shimon Even. A local-ratio theorem for approximating the weighted vertex cover problem. *Annals of Discrete Mathematics*, 25:27–46, 1985. [1](#), [3](#)
- [9] Soheil Behnezhad, Moses Charikar, Weiyun Ma, and Li-Yang Tan. Almost 3-approximate correlation clustering in constant rounds. *arXiv preprint arXiv:2205.03710*, 2022. [5](#), [8](#), [11](#)
- [10] Patrick Bennett and Tom Bohman. A note on the random greedy independent set algorithm. *Random Structures & Algorithms*, 49(3):479–502, 2016. [2](#), [4](#)
- [11] Guy E Blelloch, Jeremy T Fineman, and Julian Shun. Greedy sequential maximal independent set and matching are parallel on average. In *Proceedings of the twenty-fourth annual ACM symposium on Parallelism in algorithms and architectures*, pages 308–317, 2012. [2](#), [4](#), [12](#)
- [12] Shuchi Chawla, Konstantin Makarychev, Tselil Schramm, and Grigory Yaroslavtsev. Near optimal LP rounding algorithm for correlation clustering on complete and complete k-partite graphs. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, pages 219–228. ACM, 2015. [5](#)
- [13] Jianer Chen. *Vertex Cover Kernelization*, pages 1003–1005. Springer US, Boston, MA, 2008. [1](#)
- [14] Jianer Chen, Iyad A Kanj, and Weijia Jia. Vertex cover: further observations and further improvements. *Journal of Algorithms*, 41(2):280–301, 2001. [1](#)
- [15] Jianer Chen, Iyad A Kanj, and Ge Xia. Improved parameterized upper bounds for vertex cover. In *International symposium on mathematical foundations of computer science*, pages 238–249. Springer, 2006. [1](#)

- [16] Vincent Cohen-Addad, Silvio Lattanzi, Slobodan Mitrović, Ashkan Norouzi-Fard, Nikos Parotsidis, and Jakub Tarnawski. Correlation clustering in constant many parallel rounds. In *International Conference on Machine Learning*, pages 2069–2078. PMLR, 2021. [5](#)
- [17] Vincent Cohen-Addad, Euiwoong Lee, and Alantha Newman. Correlation clustering with sherali-adams. *arXiv preprint arXiv:2207.10889*, 2022. [5](#)
- [18] Don Coppersmith, Prabhakar Raghavan, and Martin Tompa. Parallel graph algorithms that are efficient on average. In *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, pages 260–269. IEEE, 1987. [2](#), [4](#)
- [19] François Delbot and Christian Laforest. A better list heuristic for vertex cover. *Information Processing Letters*, 107(3-4):125–127, 2008. [3](#), [4](#), [7](#)
- [20] Irit Dinur and Samuel Safra. On the hardness of approximating minimum vertex cover. *Annals of mathematics*, pages 439–485, 2005. [1](#)
- [21] David Easley and Jon Kleinberg. *Networks, crowds, and markets*, volume 8. Cambridge university press Cambridge, 2010. [9](#)
- [22] Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM (JACM)*, 43(2):268–292, 1996. [2](#), [4](#)
- [23] Manuela Fischer and Andreas Noever. Tight analysis of parallel randomized greedy mis. *ACM Transactions on Algorithms (TALG)*, 16(1):1–13, 2019. [2](#), [4](#), [5](#), [11](#), [12](#)
- [24] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990. [1](#), [3](#)
- [25] Naveen Garg, Vijay V Vazirani, and Mihalis Yannakakis. Multiway cuts in node weighted graphs. *Journal of Algorithms*, 50(1):49–61, 2004. [1](#)
- [26] Shivam Garg and Geevarghese Philip. Raising the bar for vertex cover: Fixed-parameter tractability above a higher guarantee. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1152–1166. SIAM, 2016. [1](#)
- [27] Mohsen Ghaffari, Themis Gouleakis, Christian Konrad, Slobodan Mitrović, and Ronitt Rubinfeld. Improved massively parallel computation algorithms for mis, matching, and vertex cover. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, pages 129–138, 2018. [1](#), [4](#), [17](#)
- [28] Mohsen Ghaffari, Ce Jin, and Daan Nilis. A massively parallel algorithm for minimum weight vertex cover. In *Proceedings of the 32nd ACM Symposium on Parallelism in Algorithms and Architectures*, pages 259–268, 2020. [1](#), [4](#), [17](#)
- [29] Lee-Ad Gottlieb, Aryeh Kontorovich, and Robert Krauthgamer. Efficient classification for metric data. *IEEE Transactions on Information Theory*, 60(9):5750–5759, 2014. [1](#)
- [30] Niels Grüttemeier. *Parameterized Algorithmics for Graph-Based Data Analysis*. PhD thesis, Philipps-Universität Marburg, 2022. [5](#), [9](#)
- [31] Niels Grüttemeier and Christian Komusiewicz. On the relation of strong triadic closure and cluster deletion. *Algorithmica*, 82(4):853–880, 2020. [5](#), [9](#), [17](#)

- [32] Eran Halperin. Improved approximation algorithms for the vertex cover problem in graphs and hypergraphs. *SIAM Journal on Computing*, 31(5):1608–1623, 2002. [1](#), [3](#)
- [33] George Karakostas. A better approximation ratio for the vertex cover problem. *ACM Trans. Algorithms*, 5(4), nov 2009. [1](#), [3](#)
- [34] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972. [1](#)
- [35] Richard M Karp and Avi Wigderson. A fast parallel algorithm for the maximal independent set problem. *Journal of the ACM (JACM)*, 32(4):762–773, 1985. [4](#)
- [36] Sreyash Kenkre, Vinayaka Pandit, Manish Purohit, and Rishi Saket. On the approximability of digraph ordering. *Algorithmica*, 78(4):1182–1205, 2017. [1](#), [2](#), [5](#), [14](#)
- [37] 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. [1](#)
- [38] Nathan Klein. On the approximability of dag edge deletion. Bachelor’s thesis, Oberlin College, 2016. [1](#), [14](#)
- [39] Christos Koufogiannakis and Neal E. Young. Distributed and parallel algorithms for weighted vertex cover and other covering problems. In *Proceedings of the 28th ACM Symposium on Principles of Distributed Computing*, PODC ’09, pages 171–179, New York, NY, USA, 2009. Association for Computing Machinery. [1](#)
- [40] Stefan Kratsch. A randomized polynomial kernelization for vertex cover with a smaller parameter. *SIAM Journal on Discrete Mathematics*, 32(3):1806–1839, 2018. [1](#)
- [41] Michael Lampis, Georgia Kaouri, and Valia Mitsou. On the algorithmic effectiveness of digraph decompositions and complexity measures. In *International Symposium on Algorithms and Computation*, pages 220–231. Springer, 2008. [14](#)
- [42] Michael Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM journal on computing*, 15(4):1036–1053, 1986. [4](#)
- [43] Béla Nuendorf. On strong triadic closure with edge insertion. Bachelor’s thesis, Philipps-Universität Marburg, 2020. [5](#), [9](#)
- [44] Doowon Paik, Sudhakar Reddy, and Sartaj Sahni. Deleting vertices to bound path length. *IEEE transactions on computers*, 43(9):1091–1096, 1994. [14](#)
- [45] Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998. [3](#)
- [46] Kihong Park and Heejo Lee. On the effectiveness of route-based packet filtering for distributed dos attack prevention in power-law internets. *ACM SIGCOMM computer communication review*, 31(4):15–26, 2001. [1](#)
- [47] Leonard Brian Pitt. *A simple probabilistic approximation algorithm for vertex cover*. Yale University, Department of Computer Science, 1985. [1](#), [3](#)
- [48] Carla Savage. Depth-first search and the vertex cover problem. *Information processing letters*, 14(5):233–235, 1982. [1](#), [3](#)

- [49] Stavros Sintos and Panayiotis Tsaparas. Using strong triadic closure to characterize ties in social networks. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '14, pages 1466–1475, 2014. [1](#), [2](#), [5](#), [8](#), [9](#), [17](#)
- [50] Nate Veldt. Correlation clustering via strong triadic closure labeling: Fast approximation algorithms and practical lower bounds. In *Proceedings of the 2022 International Conference on Machine Learning*, ICML '22, 2022. [1](#), [5](#), [8](#), [9](#), [11](#), [15](#), [16](#), [17](#)
- [51] Nate Veldt. Optimal LP rounding and fast combinatorial algorithms for clustering edge-colored hypergraphs. *arXiv preprint arXiv:2208.06506*, 2022. [16](#)
- [52] C. K. Wong and M. C. Easton. An efficient method for weighted sampling without replacement. *SIAM Journal on Computing*, 9(1):111–113, 1980. [7](#)
- [53] David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 681–690, 2006. [2](#), [4](#)