An $O(1)$-Approximation Algorithm for Dynamic Weighted Vertex Cover with Soft Capacity

Hao-Ting Wei
Department of Industrial Engineering and Engineering Management, National Tsing Hua University, Hsinchu 30013, Taiwan
s104034526@m104.nthu.edu.tw

Wing-Kai Hon
Department of Computer Science, National Tsing Hua University, Hsinchu 30013, Taiwan
wkhon@cs.nthu.edu.tw

Paul Horn
Department of Mathematics, University of Denver, Denver, USA
paul.horn@du.edu

Chung-Shou Liao
Department of Industrial Engineering and Engineering Management, National Tsing Hua University, Hsinchu 30013, Taiwan
cslio@ie.nthu.edu.tw

Kunihiro Sadakane
Department of Mathematical Informatics, The University of Tokyo, Tokyo, Japan
sada@mist.i.u-tokyo.ac.jp

Abstract
This study considers the soft capacitated vertex cover problem in a dynamic setting. This problem generalizes the dynamic model of the vertex cover problem, which has been intensively studied in recent years. Given a dynamically changing vertex-weighted graph $G = (V, E)$, which allows edge insertions and edge deletions, the goal is to design a data structure that maintains an approximate minimum vertex cover while satisfying the capacity constraint of each vertex. That is, when picking a copy of a vertex $v$ in the cover, the number of $v$’s incident edges covered by the copy is up to a given capacity of $v$. We extend Bhattacharya et al.’s work [SODA’15 and ICALP’15] to obtain a deterministic primal-dual algorithm for maintaining a constant-factor approximate minimum capacitated vertex cover with $O(\log n/\epsilon)$ amortized update time, where $n$ is the number of vertices in the graph. The algorithm can be extended to (1) a more general model in which each edge is associated with a non-uniform and unsplittable demand, and (2) the more general capacitated set cover problem.

2012 ACM Subject Classification Theory of computation → Dynamic graph algorithms

Keywords and phrases approximation algorithm, dynamic algorithm, primal-dual, vertex cover

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2018.27

1 Introduction

Dynamic algorithms have received fast-growing attention in the past decades, especially for some classical combinatorial optimization problems such as connectivity [1, 8, 11], vertex cover, and maximum matching [2, 3, 4, 5, 13, 14, 15, 16]. This paper focuses on the fully dynamic model of the vertex cover problem, which has been intensively studied in recent years. Given a vertex-weighted graph $G = (V, E)$ which is constantly updated due to a sequence of edge insertions and edge deletions, the objective is to maintain a subset of vertices $S \subseteq V$ at any given time, such that every edge is incident to at least one vertex in $S$ and the weighted sum of $S$ is minimized. We consider a generalization of the problem, where each vertex is associated with a given capacity. When picking a copy of a vertex $v$ in $S$, the number of its incident edges that can be covered by such a copy is bounded by $v$’s given capacity. The objective is to find a soft capacitated weighted vertex cover $S$ with minimum weight, i.e. $\sum_{v \in S} c_v x_v$ is minimized, as well as an assignment of edges such that the number of edges assigned to a vertex $v$ in $S$ is at most $k_v x_v$, where $c_v$ is the cost of $v$, $k_v$ is the capacity of $v$, and $x_v$ is the number of selected copies of $v$ in $S$. Assume there is no bound on $x_v$. The static model of this generalization is the so-called soft capacitated vertex cover problem, introduced by Guha et al. [9].

Prior work. For the vertex cover problem in a dynamic setting, Ivkovic and Lloyd [12] presented the pioneering work wherein their fully dynamic algorithm maintains a 2-approximation factor to vertex cover with $O((n + m)^{0.7072})$ update time, where $n$ is the number of vertices and $m$ is the number of edges. Onak and Rubinfeld [14] designed a randomized data structure that maintains a large constant approximation ratio with $O(\log^2 n)$ amortized update time in expectation; this is the first result that achieves a constant approximation factor with polylogarithmic update time. Baswana, Gupta, and Sen [2] designed another randomized data structure which improves the approximation ratio to two, and simultaneously improved the amortized update time to $O(\log n)$. Recently, Solomon [16] gave the currently best randomized algorithm, which maintains a 2-approximate vertex cover with $O(1)$ amortized update time.

For deterministic data structures, Onak and Rubinfeld [14] presented a data structure that maintains an $O(\log n)$-approximation algorithm with $O(\log^2 n)$ amortized update time. Bhattacharya et al. [5] proposed the first deterministic data structure that maintains a constant ratio, precisely, a $(2 + \epsilon)$-approximation to vertex cover with polylogarithmic $O(\log n/\epsilon^2)$ amortized updated time. Existing work also considered the worst-case update time. Neiman and Solomon [13] provided a 2-approximation dynamic algorithm with $O(\sqrt{m})$ worst-case update time. Later, Peleg and Solomon [15] improved the worst-case update time to $O(\gamma/\epsilon^2)$, where $\gamma$ is the arboricity of the input graph. Very recently, Bhattacharya et al. [3] extended their hierarchical data structure to achieve the currently best worst-case update time of $O(\log^3 n)$. Note that the above studies only discussed the unweighted vertex cover problem, the objective of which is to find a vertex cover with minimum cardinality.

Consider the dynamic (weighted) set cover problem. Bhattacharya et al. [6] used a hierarchical data structure similar to that reported in [5], and achieved a scheme with $O(f^2)$-approximation ratio and $O(f \log(n + m)/\epsilon^2)$ amortized updated time, where $f$ is the maximum frequency of an element. Very recently, Gupta et al. [10] improved the amortized

\[3\] If each $x_v$ is associated with a bound, it is called the hard capacitated vertex cover problem, introduced by Chuzhoy and Naor [7].
Table 1 Summary of results for unweighted (resp. weighted) minimum vertex cover (UMVC (resp. WMVC)), unweighted (resp. weighted) minimum set cover (UMSC (resp. WMSC)), where \( f \) is the maximum frequency of an element, and weighted minimum capacitated vertex (resp. set) cover (WMCVC (resp. WMCSC)).

<table>
<thead>
<tr>
<th>Problem</th>
<th>Approx. Guarantee</th>
<th>Update Time</th>
<th>Data Structure</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>UMVC</td>
<td>( O(1) )</td>
<td>( O(\log^2 n) ) amortized</td>
<td>randomized</td>
<td>STOC’10 [14]</td>
</tr>
<tr>
<td>UMVC</td>
<td>2</td>
<td>( O(\log n) ) amortized</td>
<td>randomized</td>
<td>FOCS’11 [2]</td>
</tr>
<tr>
<td>UMVC</td>
<td>2</td>
<td>( O(1) ) amortized</td>
<td>randomized</td>
<td>FOCS’16 [16]</td>
</tr>
<tr>
<td>UMVC</td>
<td>2</td>
<td>( O(\sqrt{m}) ) worst-case</td>
<td>deterministic</td>
<td>SODA’13 [13]</td>
</tr>
<tr>
<td>UMVC</td>
<td>2 + ( \epsilon )</td>
<td>( O(\log n/\epsilon^2) ) amortized</td>
<td>deterministic</td>
<td>SODA’15 [5]</td>
</tr>
<tr>
<td>UMVC</td>
<td>2 + ( \epsilon )</td>
<td>( O(\log^2 n) ) worst-case</td>
<td>deterministic</td>
<td>SODA’16 [15]</td>
</tr>
<tr>
<td>WMVC</td>
<td>2 + ( \epsilon )</td>
<td>( O(\log n/\epsilon^2) ) amortized</td>
<td>deterministic</td>
<td>SODA’17 [3]</td>
</tr>
<tr>
<td>WMSC</td>
<td>( O(f^3) )</td>
<td>( O(f^2) ) amortized</td>
<td>deterministic</td>
<td>IPCO’17 [4]</td>
</tr>
<tr>
<td>WMSC</td>
<td>( O(f^2) )</td>
<td>( O(f \log(n + m)/\epsilon^2) ) amortized</td>
<td>deterministic</td>
<td>ICALP’15 [6]</td>
</tr>
<tr>
<td>WMCVC</td>
<td>( O(f^3) )</td>
<td>( O(f^2) ) amortized</td>
<td>deterministic</td>
<td>STOC’17 [10]</td>
</tr>
<tr>
<td>WMCSC</td>
<td>( O(f^2) )</td>
<td>( O(f \log(n + m)/\epsilon) ) amortized</td>
<td>deterministic</td>
<td>This paper</td>
</tr>
<tr>
<td>WMCSC</td>
<td>( O(f^2) )</td>
<td>( O(f \log(n + m)/\epsilon) ) amortized</td>
<td>deterministic</td>
<td>This paper</td>
</tr>
</tbody>
</table>

update time to \( O(f^2) \), albeit the dynamic algorithm achieves a higher approximation ratio of \( O(f^3) \). They also offered another \( O(\log n) \)-approximation dynamic algorithm in \( O(f \log n) \) amortized update time. Bhattacharya et al. [4] simultaneously derived the same outcome with \( O(f^2) \)-approximation ratio and \( O(f^2) \) amortized update time for the unweighted set cover problem. Table 1 presents a summary of the above results.

Our contribution. In this study we investigate the soft capacitated vertex cover problem in the dynamic setting, where there is no bound on the number of copies of each vertex that can be selected. We refer to the primal-dual technique reported in [9], and present the first deterministic algorithm for this problem, which can maintain an \( O(1) \)-approximate minimum capacitated (weighted) vertex cover with \( O(\log n/\epsilon) \) amortized update time. The algorithm can be extended to a more general model in which each edge is associated with a given demand, and the demand has to be assigned to an incident vertex. That is, the demand of each edge is nonuniform and unsplittable. Also, it can be extended to solve the more general capacitated set cover problem, where the input graph is a hyper-graph, and each edge may connect to multiple vertices.

The proposed dynamic mechanism builds on Bhattacharya et al.’s \((\alpha, \beta)\)-partition structure [5, 6], but a careful adaptation has to be made to cope with the newly introduced capacity constraint. Briefly, applying the fractional matching technique in Bhattacharya et al.’s algorithm cannot directly lead to a constant approximation ratio in the capacitated vertex cover problem. The crux of our result is the re-design of a key parameter, weight of a vertex, in the dual model. Details of this modification are shown in the next section.

In addition, if we go back to the original vertex cover problem without capacity constraint, the proposed algorithm is able to resolve the weighted vertex cover problem by maintaining a \((2 + \epsilon)\)-approximate weighted vertex cover with \( O(\log n/\epsilon^2) \) amortized update time. This result achieves the same approximation ratio as the algorithm in [5], but they considered the unweighted model. Details of this discussion are presented in the end of Section 3.
1.1 Overview of our technique

First, we recall the mathematical model of the capacitated vertex cover problem which was first introduced by Guha et al. [9]. In this model, $y_{ev}$ serves as a binary variable that indicates whether an edge $e$ is covered by a vertex $v$. Let $N_v$ be the set of incident edges of $v$, $k_e$ and $c_v$ be the capacity and the cost of a vertex $v$, respectively. Let $x_v$ be the number of selected copies of a vertex $v$. An integer program (IP) model of the problem can be formulated as follows (the minimization program on the left):

Min $\sum_v c_v x_v$

s.t.

\[ y_{ev} \geq 1, \quad \forall e = \{u, v\} \in E \]
\[ k_v x_v - \sum_{e \in N_v} y_{ev} \geq 0, \quad \forall v \in V \]
\[ x_v \geq y_{ev}, \quad \forall v \in V \]
\[ y_{ev} \in \{0, 1\}, \quad \forall v \in V \]
\[ x_v \in \mathbb{N}, \quad \forall v \in V \]

Max $\sum_{e \in E} \pi_e$

s.t.

\[ k_v q_{ev} + \sum_{n \in N_v} l_{ev} \leq c_v, \quad \forall v \in V \]
\[ q_v \geq 0, \quad \forall v \in E \]
\[ l_{ev} \geq 0, \quad \forall v \in e, \forall e \in E \]
\[ \pi_e \geq 0, \quad \forall e \in E \]

If we allow a relaxation of the above primal form, i.e., dropping the integrality constraints, its dual problem yields a maximization problem. The linear program for the dual can be formulated as shown in the above (the maximization program on the right; also see [9]). One may consider this as a variant of the packing problem, where we want to pack a value of $\pi_e$ for each edge $e$, so that the sum of the packed values is maximized. Packing of $e$ is limited by the sum of $q_v$ and $l_{ev}$, where $q_v$ is the global ability of a vertex $v$ emitted to $v$’s incident edges, and $l_{ev}$ is the local ability of $v$ distributed to its incident edge $e$.

In this study, we incorporate the above IP model with its LP relaxation for capacitated vertex cover into the dynamic mechanism proposed by Bhattacharya et al. [5, 6]. They devised the weight of a vertex $v$ (in the dual model), denoted by $W_v$, to obtain a feasible solution in their dual problem. They also allowed a flexible range for $W_v$ to quickly adjust the solution for dynamic updates while preserving its approximation quality. Due to the additional capacity constraint in our problem, a new weight function is obviously required.

**Technical challenges.** There are two major differences between our algorithm and Bhattacharya et al.’s [5, 6]. First, the capacity constraint in the primal problem leads to the two variables $q_v$ and $l_{ev}$ in the dual problem in which we have to balance their values when approaching $c_v$ to maximize the dual objective. By contrast, the previous work considered one dual variable $l_{ev}$ without the restriction on the coverage of a vertex. We thus re-design $W_v$, the weight of a vertex $v$ to specifically consider the capacitated scenario. Yet, even with the new definition of $W_v$, there is still a second challenge on how to approximate the solution within a constant factor in the dynamic environment. In order to achieve $O(\log n)$ amortized update time, Bhattacharya et al.’s fractional matching approach assigns the value of all $v$’s incident edges to $v$, which, however, may result in a non-constant $h$, hidden in the approximation ratio, where $h$ is the largest number of copies selected in the cover. We observe that we cannot remove $h$ from the approximation guarantee based on the $(\alpha, \beta)$-partition structure if we just select the minimum value of $\alpha$, as it is done in [5, 6]. The key insight is that we show a bound on the value of $\alpha$, which restricts the updates of the dynamic mechanism. With the help of this insight, we are able to revise the setting of $\alpha$ to derive a constant approximation ratio, while maintaining the $O(\log n)$ update time.

2 Level Scheme and its Key Property

The core of Bhattacharya et al.’s $(\alpha, \beta)$-partition structure [5, 6] is a level scheme [14] that is used to maintain a feasible solution in their dual problem. In this section, we demonstrate
A level scheme is an assignment \( \ell : V \to \{0, 1, \ldots, L\} \) such that every vertex \( v \in V \) has a level \( \ell(v) \). Let \( c_{\min} \) and \( c_{\max} \) denote the minimum and maximum costs of a vertex, respectively. For our case, we set \( L = \lceil \log_2(n \mu / c_{\min}) \rceil \) for some \( \alpha, \beta > 1 \) and \( \mu > c_{\max} \). Based on \( \ell \), each edge \((u, v)\) is also associated with a level \( \ell(u, v) \), where \( \ell(u, v) = \max\{\ell(u), \ell(v)\} \). An edge is assigned to the higher-level endpoint, and ties are broken arbitrarily if both endpoints have the same level.

Each edge \((u, v)\) has a weight \( w(u, v) \) according to its level, such that \( w(u, v) = \mu \beta^{-\ell(u, v)} \). Each vertex \( v \) also has a weight \( W_v \), which is defined based on the incident edges of \( v \) and their corresponding levels. Before giving details on \( W_v \), we first define some notations. Let \( N_v = \{u \mid (u, v) \in E\} \) be the set of vertices adjacent to \( v \) (i.e., the neighbors of \( v \)). Let \( N_v(i) \) denote the set of level-i neighbors of \( v \), and \( N_v(i, j) \) denote the set of \( v \)'s neighbors whose levels are in the range \([i, j]\). That is, \( N_v(i) = \{u \mid (u, v) \in E \land \ell(u) = i\} \) and \( N_v(i, j) = \{u \mid (u, v) \in E \land \ell(u) \in [i, j]\} \). The degree of a vertex \( v \) is denoted by \( D_v = |N_v| \). Similarly, we define \( D_v(i) = |N_v(i)| \) and \( D_v(i, j) = |N_v(i, j)| \). Finally, we use \( \delta(v) \) to denote the set of edges assigned to a vertex \( v \). Now, the weight \( W_v \) of a vertex \( v \) is defined as follows:

**Case 1** \( D_v(0, \ell(v)) > k_v \):

\[
W_v = k_v \mu \beta^{-\ell(v)} + \sum_{i > \ell(v)} \min\{k_v, D_v(i)\} \mu \beta^{-i}
\]

**Case 2** \( D_v(0, \ell(v)) \leq k_v \):

\[
W_v = D_v(0, \ell(v)) \mu \beta^{-\ell(v)} + \sum_{i > \ell(v)} \min\{k_v, D_v(i)\} \mu \beta^{-i}
\]

Due to the capacity constraint, we consider whether the number of level-i neighbors of \( v \), \( 0 \leq i \leq \ell(v) \), is larger than the capacity of \( v \), to define the weight of a vertex \( v \). Note that the total weight of the edges that are assigned to \( v \) or incident to \( v \) can contribute at most \( k_v w(u, v) \) to \( W_v \). Briefly, the weight of a vertex has two components: one that is dependent on the incident edges with level \( \ell(v) \), and the other that is dependent on the remaining incident edges. For convenience, we call the former component \( \text{Internal}_v \) and the latter component as \( \text{External}_v \). Moreover, we have:

\[
\text{External}_v \leq k_v \sum_{i > \ell(v)} \mu \beta^{-i} \leq (1/(\beta - 1)) k_v \mu \beta^{-\ell(v)}.
\]

In general, an arbitrary level scheme cannot be used to solve our problem. What we need is a valid level scheme, which is defined as follows.

**Definition 1.** A level scheme is valid if \( W_v \leq c_v \) for every vertex \( v \).

**Lemma 2.** Let \( V_0 \) denote the set of level-0 vertices in a valid level scheme. Then, \( V \setminus V_0 \) forms a vertex cover of \( G \).

**Proof.** Consider any edge \((u, v)\) \( \in E \). We claim that at least one of its endpoints must be in \( V \setminus V_0 \). Suppose that the claim is false which implies that \( \ell(u) = \ell(v) = 0 \) and \( w(u, v) = \mu > c_{\max} \). Since \( w(u, v) \) appears in \( \text{Internal}_v \), we have \( W_v \geq w(u, v) \). As a result, \( c_v \geq W_v \geq \mu > c_{\max} \), which leads to a contradiction. The claim thus follows, and so does the lemma. \( \blacksquare \)
The above lemma implies that no edge is assigned to any level-0 vertex. In our mechanism, we will maintain a valid level scheme, based on which each vertex in \( V \setminus V_0 \) picks enough copies to cover all the edges assigned to it; this forms a valid capacitated vertex cover.

Next, we define the notion of tightness, which is used to measure how good a valid level scheme performs.

**Definition 3.** A valid level scheme with an associated edge assignment is \( \varepsilon \)-tight if for every vertex \( v \) with \(|\delta(v)| > 0\), \( W_v \in (c_v/\varepsilon, c_v] \).

**Lemma 4.** Given an \( \varepsilon \)-tight valid level scheme, we can obtain an \( \varepsilon (2(\beta/(\beta - 1)) + 1) \)-approximation solution to the weighted minimum capacitated vertex cover (WMCVC) problem.

**Proof.** First, we fix an arbitrary edge assignment that is consistent with the given valid level scheme. For each vertex \( v \) with \(|\delta(v)| > 0\), we pick \( |\delta(v)|/k_v \) copies to cover all the \(|\delta(v)|\) edges assigned to it. To analyze the total cost of this capacitated vertex cover, we relate it to the value \( \sum \pi_e \) of a certain feasible solution of the dual problem, whose corresponding values of \( q_v \) and \( l_{ev} \) are as follows:

For every vertex \( v \):

- if \(|\delta(v)|/k_v > 1\): \( q_v = \mu \beta^{-\ell(v)} \), and \( l_{ev} = 0 \);
- if \(|\delta(v)|/k_v \leq 1\): \( q_v = \mu \sum_{i|D_v(i) > k_v} \beta^{-i} \), \( l_{ev} = 0 \) if \( D_v(\ell(e)) > k_v \), and \( l_{ev} = \mu \beta^{-\ell(e)} \) otherwise.

For every edge \( e \): \( \pi_e = \mu \beta^{-\ell(e)} \).

It is easy to verify that the above choices of \( q_v \), \( l_{ev} \), and \( \pi_e \) give a feasible solution to the dual problem.

For the total cost of our solution, we separate the analysis into two parts, based on the multiplicity of the vertex:

**Case 1 \(|\delta(v)|/k_v > 1\):** In this case, the external component of \( W_v \) is at most \( 1/(\beta - 1) \) of the internal component, so \( W_v \leq (\beta/(\beta - 1)) k_v q_v \). Then, the cost of all copies of \( v \) is:

\[
|\delta(v)|/k_v \cdot c_v \leq |\delta(v)|/k_v \cdot \varepsilon \cdot W_v \\
\leq 2 \cdot \left( |\delta(v)|/k_v \cdot \varepsilon \cdot (\beta/(\beta - 1)) k_v q_v \right) = 2 \varepsilon (\beta/(\beta - 1)) \cdot \sum_{e \in \delta(v)} \pi_e.
\]

**Case 2 \(|\delta(v)|/k_v = 1\):** In this case, we pick one copy of vertex \( v \), whose cost is:

\[
c_v \leq \varepsilon \cdot W_v \leq \varepsilon \cdot \sum_{e \sim v} \pi_e = \varepsilon \cdot \left( \sum_{e \in \delta(v)} \pi_e + \sum_{e \not\in \delta(v), e \sim v} \pi_e \right),
\]

where \( e \sim v \) denotes \( e \) is an edge incident to \( v \).

In summary, the total cost is bounded by:

\[
\sum_v \left( \max\{\varepsilon, 2\varepsilon (\beta/(\beta - 1))\} \sum_{e \in \delta(v)} \pi_e + \varepsilon \sum_{e \not\in \delta(v), e \sim v} \pi_e \right) \\
= \sum_v \left( 2\varepsilon (\beta/(\beta - 1)) \sum_{e \in \delta(v)} \pi_e + \varepsilon \sum_{e \not\in \delta(v), e \sim v} \pi_e \right) \\
= \varepsilon (2(\beta/(\beta - 1)) + 1) \sum_e \pi_e \\
\leq \varepsilon (2(\beta/(\beta - 1)) + 1) \cdot OPT,
\]
where \( OPT \) denotes the optimal solution of the dual problem, which is also a lower bound of the cost of any weighted capacitated vertex cover.

The next section discusses how to dynamically maintain an \( \varepsilon \)-tight level scheme, for some constant factor \( \varepsilon \) and with amortized \( O(\log n/\varepsilon) \) update time. Before that, we show a greedy approach to get a \((\beta + 1)\)-tight level scheme to the static problem as a warm up.

First, we have the following definition.

\textbf{Definition 5.} A valid level scheme \( \lambda \) is \textit{improvable} if some vertex can drop its level to get another level scheme \( \lambda' \) such that \( \lambda' \) is valid; otherwise, we say \( \lambda \) is \textit{non-improvable}.

\textbf{Lemma 6.} If a valid level scheme \( \lambda \) is non-improvable, then \( \lambda \) is \((\beta + 1)\)-tight.

If we set the level of every vertex to \( L \) initially, it is easy to check that by our choice of \( L \) as \( \lceil \log_{\beta}(n\mu\alpha/c_{\min}) \rceil \), such a level scheme is valid. Next, we examine each vertex one by one, and drop its level as much as possible while the scheme remains valid. In the end, we will obtain a non-improvable scheme, so that by the above lemma, the scheme is \((\beta + 1)\)-tight. This implies a \((\beta + 1)(2(\beta/(\beta - 1)) + 1)\)-approximate solution for the WMCVC problem.

\section{Maintaining an \( \alpha(\beta + 1) \)-tight Level Scheme Dynamically}

In this section, we present our \( O(1) \)-approximation algorithm for the WMCVC problem, with amortized \( O(\log n) \) update time for each edge insertion and edge deletion. We first state an invariant that is maintained throughout by our algorithm, and show how the latter is done. Next, we analyze the time required to maintain the invariant with the potential method, and show that our proposed method can be updated efficiently as desired. To obtain an \( O(\log n) \) amortized update time, we relax the flexible range of the weight of a vertex \( W_v \) by multiply a constant \( \alpha \). Let \( c^*_v \) be \( c_v/\alpha(\beta + 1) \). The invariant that we maintain is as follows.

\textbf{Invariant 7.} (1) For every vertex \( v \in V \setminus V_0 \), it holds that \( c^*_v \leq W_v \leq c_v \), and (2) for every vertex \( v \in V_0 \), it holds that \( W_v \leq c_v \).

By maintaining the above invariant, we will automatically obtain an \( \alpha(\beta + 1) \)-tight valid scheme. As mentioned, we will choose a value for \( \alpha \) in order to remove \( h \) from the approximation ratio. In particular, we will set \( \alpha = (2\beta + 1)/\beta + 2\varepsilon \), where \( 0 < \varepsilon < 1 \) to balance the update time, and \( \beta = 2.43 \) to minimize the approximation ratio, so that we achieve the following theorem.

\textbf{Theorem 8.} There exists a dynamic level scheme \( \lambda \) which can achieve a constant approximation ratio \((\approx 36)\) for the WMCVC problem with \( O(\log n/\varepsilon) \) amortized update time.

The remainder of this section is devoted to proving Theorem 8.

\subsection{The algorithm: Handling insertion or deletion of an edge}

We now show how to maintain the invariant under edge insertions and deletions. A vertex is called \textit{dirty} if it violates Invariant 7, and \textit{clean} otherwise. Initially, the graph is empty, so that every vertex is clean and is at level zero. Assume that at the time instant just prior to the \( t^{th} \) update, all vertices are clean. When the \( t^{th} \) update takes place, which either inserts or deletes an edge \( e = (u, v) \), we need to adjust the weights of \( u \) and \( v \) accordingly. Due to this adjustment, the vertices \( u \), or \( v \), or both may become dirty. To recover from this, we call the procedure Fix. The pseudo codes of the update algorithm (Algorithm 1) and the procedure Fix are shown in the next page.
Algorithm 1

1: if an edge \( e = (u, v) \) has been inserted then
2: Set \( \ell(e) = \max \{\ell(u), \ell(v)\} \) and set \( w(u, v) = \mu \beta^{-\ell(e)} \)
3: Update \( W_u \) and \( W_v \)
4: else if an edge \( e = (u, v) \) has been deleted then
5: Update \( W_u \) and \( W_v \)
6: end if
7: Run procedure \text{Fix}

procedure \text{Fix}:

1: while there exists a dirty vertex \( v \) do
2: if \( W_v > c_v \) then
3: Increment the level of \( v \) by setting \( \ell(v) \leftarrow \ell(v) + 1 \)
4: Update \( W_u \) and \( W_v \) for all affected \( v \)'s neighboring vertices \( u \)
5: else if \( W_v < c_v^* \) and \( \ell(v) > 0 \) then
6: Decrement the level of \( v \) by setting \( \ell(v) \leftarrow \ell(v) - 1 \)
7: Update \( W_v \) and \( W_u \) for all affected \( v \)'s neighboring vertices \( u \)
8: end if
9: end while

Algorithm 1 ensures that Invariant 7 is maintained after each update, so that the dynamic scheme is \( \alpha(\beta + 1) \)-tight as desired. To complete the discussion, as well as the proof of Theorem 8, it remains to show that each update can be performed efficiently, in amortized \( O(\log n) \) time.

3.2 Time complexity

Each update involves two steps, namely the adjustment of weights of the endpoints, and the running of procedure \text{Fix}. We now give the time complexity analysis, where the main idea is to prove the following two facts: (Fact 1) the amortized cost of the adjustment step is \( O(\log n) \), and (Fact 2) the amortized cost of the procedure \text{Fix} is zero, irrespective of the number of vertices or edges that are affected during this step. Once the above two facts are proven, the time complexity analysis follows.

We use the standard potential method in our amortized analysis. Imagine that we have a bank account \( B \). Initially, the graph is empty, and the bank account \( B \) has no money. For each adjustment step during an edge insertion or deletion, we deposit some money into the bank account \( B \); after that, we use the money in \( B \) to pay for the cost of the procedure \text{Fix}. Some proofs are omitted in the following due to space limit.

Following the definition of \([6]\), we say a vertex \( v \in V \) is active if its degree in \( G \) is non-zero, and passive otherwise. Now, the value of \( B \) is set by the following formula:

\[
B = \frac{1}{\epsilon} \cdot \left( \sum_{e \in E} \phi(e) + \sum_{v \in V} \psi(v) \right),
\]

where \( 0 < \epsilon < 1 \), and \( \phi \) and \( \psi \) are functions defined as follows:

\[
\phi(e) = \left( \frac{\beta}{(\beta - 1) + \epsilon} \right) (L - \ell(e)).
\]
\[
\psi(v) = \begin{cases} 
\beta^{\ell(v)+1} \max \{0, c_v^* - W_v\}, & \text{if } v \text{ is active.} \\
\mu (\beta - 1) \cdot \max \{0, \alpha c_v^* - W_v\}, & \text{otherwise.}
\end{cases}
\]

The following lemma proves Fact 1.

Lemma 9. After the adjustment step, the potential \(B\) increases by at most \(O(\log n/\epsilon)\).

We now switch our attention to Fact 2. Observe that the procedure \(\text{Fix}\) performs a series of level up and level down events. For each such event, the level of a specific vertex \(v\) will be changed, which will then incur a change in its weight, and changes in the weights of some of the incident edges and their endpoints. Let \(t_0\) denote the moment before a level up or a level down event, and \(t_1\) denote the moment after the weights of the edges and vertices are updated due to this event. Let \(\text{COUNT}\) denote the number of times an edge in the graph \(G\) is updated (for simplicity, we assume that in one edge update, the weight and the assignment of the edge may be updated, and so do the weights of its endpoints, where all these can be done in \(O(1)\) time).

For ease of notation, in the following, a superscript \(t\) in a variable denotes the variable at moment \(t\). For instance, \(W_{t_0}^v\) stands for the weight \(W_v\) of \(v\) at moment \(t_0\). Also, we use \(\Delta x\) to denote the quantity \(x_{t_0} - x_{t_1}\), so that

\[|\Delta \text{COUNT}| = |\text{COUNT}^{t_0} - \text{COUNT}^{t_1}| = \text{COUNT}^{t_1} - \text{COUNT}^{t_0}\]

represents the number of incident edges whose weights are changed between \(t_0\) and \(t_1\).

Briefly speaking, based on the level scheme and the potential function \(B\), we can show:

- For each level up event, each of the affected edges \(e\) would have its \(\phi(e)\) value dropped, so that an \(\epsilon\) fraction can pay for the weight updates of itself and its endpoints, while the remaining fraction can be converted into the increase in \(\psi(v)\) value.
- For each level down event, the reverse happens, where the vertex \(v\) would have its \(\psi(v)\) value dropped, so that an \(\epsilon\) fraction can pay for the weight updates of the affected edges and their endpoints, while the remaining fraction can be converted into the increase in \(\phi(e)\) values of the affected edges. The \(\alpha\) value controls the frequency of the level down events, while trading this off with the approximation guarantee.

Sections 3.2.1 and 3.2.2 present the details of the amortized analysis of these two types of events, respectively. Finally, note that there is no money (potential) input to the bank \(B\) after the adjustment step, so that the analysis implies that the procedure \(\text{Fix}\) must stop (as the money in the bank is finite).

3.2.1 Amortized cost of level up

Let \(v\) be the vertex that undergoes the level up event, and \(i = \ell(v)\) denote its level at moment \(t_0\). By our notation, \(\Delta B = B^{t_0} - B^{t_1}\) denotes the potential drop in the bank \(B\) from moment \(t_0\) to moment \(t_1\). To show that the amortized cost of a level up event is at most zero, it is equivalent to show that \(\Delta B \geq |\Delta \text{COUNT}|\).

Recall that after a level up event, only the value of \(\psi(v)\), the values of \(\phi(e)\) and \(\psi(u)\) for an edge \((u,v)\) may be affected. In the following, we will examine carefully the changes in such values, and derive the desired bound for \(\Delta B\). First, we have the following simple lemma.

Lemma 10. \(|\Delta \text{COUNT}| \leq D^\nu_v(0,i)\).
An $O(1)$-Approximation Alg. for Dynamic Weighted Vertex Cover with Soft Capacity

**Proof.** When $v$ changes from level $i$ to $i+1$, only those incident edges with levels $i$ will be affected.

The next three lemmas examine, respectively, the changes $\Delta \psi(v)$, $\Delta \phi(e)$, and $\Delta \psi(u)$.

- **Lemma 11.** $\Delta \psi(v) = 0$.
- **Lemma 12.** For every edge $e$ incident to $v$,
  \[ \Delta \phi(e) = \begin{cases} \frac{\beta}{(\beta - 1)} + \epsilon, & \text{if } \ell(e) \in [0,i], \\ 0, & \text{otherwise}. \end{cases} \]
- **Lemma 13.** For every vertex $u \in N_v^{10}$, $\Delta \psi(u) \geq -\beta/(\beta - 1)$.

Based on the above lemmas, we derive the following and finish the proof for the case of level up.

\[
\Delta B = \frac{1}{\epsilon} \left( \Delta \psi(v) + \sum_{e \in E} \Delta \phi(e) + \sum_{u \in N_v^{10}} \Delta \psi(u) \right) \\
\geq \frac{1}{\epsilon} \left( 0 + \left( \frac{\beta}{(\beta - 1)} + \epsilon \right) D_v^{10}(0,i) - \frac{\beta}{\beta - 1} D_v^{10}(0,i) \right) \\
= D_v^{10}(0,i) \geq |\Delta \text{Count}|.
\]

### 3.2.2 Amortized cost of level down

We now show that the amortized cost of level down for a vertex $v$ is at most zero. Similar to the case of level up, we examine $\Delta \psi(v)$, $\Delta \phi(e)$, and $\Delta \psi(u)$, and show that $\Delta B \geq |\Delta \text{Count}|$.

Before starting the proof of the level down case, recall that we have mentioned a parameter $h$ at the end of Introduction, where $h$ is the largest number of selected copies of all the vertices. That is, $h = \max_v \{[\delta_v^{10}(v)/k_v]\}$. Also, we let $h' = \max_v \{[D_v^{10}(0, \ell(v))/k_v]\}$, where $h' \geq h$, and set $\xi \geq 0$ such that $h' = h + \xi$.

- **Lemma 14.** $|\Delta \text{Count}| \leq D_v^{10}(0,i) < h' \cdot \frac{\beta \cdot \epsilon \cdot N}{\rho}$.

Now, we are ready to examine $\Delta \psi(v)$, $\Delta \phi(e)$, and $\Delta \psi(u)$, through the following lemmas.

- **Lemma 15.** For every vertex $u \in N_v^{10}$, $\Delta \psi(u) \geq -1/(\beta - 1)$.

Next, we partition $N_v^{10}$ into three subsets: $X$, $Y_1$ and $Y_2$, i.e. $N_v^{10} = X \cup Y_1 \cup Y_2$, where

\[
X = \{ u \mid u \in N_v^{10}(0,i-1) \}, \\
Y_1 = \{ u \mid u \in N_v^{10}(i) \}, \\
Y_2 = \{ u \mid u \in N_v^{10}(i+1,L) \}.
\]

- **Lemma 16.** For every edge $(u,v)$ incident to a vertex $v$,
  \[ \Delta \phi(u,v) = \begin{cases} \frac{\beta}{(\beta - 1)} + \epsilon, & \text{if } u \in X \\ 0, & \text{if } u \in Y_1 \cup Y_2. \end{cases} \]

Next, let $W_v^{10} = x + y_1 + y_2$, where $x, y_1$ and $y_2$ on the right-hand-side correspond to the weights generated by the subsets $X$, $Y_1$, $Y_2$, respectively. So, we get the following lemmas:
Lemma 17. $\sum_{u \in N_v^0} \Delta \phi(u, v) \leq - \left( \frac{\beta}{(\beta - 1)} + \epsilon \right) (hx/\mu)$.

Lemma 18. $\Delta \psi(v) = (\alpha c_v^* - x - y_1 - y_2) \cdot \frac{\beta^{t/2}}{\mu(\beta - 1)} - \max \{0, \alpha c_v^* - \beta x - y_1 - y_2\} \cdot \frac{\beta^t}{\mu(\beta - 1)}$.

Finally, depending upon the value of $\alpha c_v^* - \beta x - y_1 - y_2$, we consider two possible scenarios, where we show that in each case, $\Delta B \geq \beta^{t/2} c_v^*/\mu$. This in turn implies $\Delta B \geq |\Delta \text{COUNT}|$ as desired. Thus, the level scheme remains $\alpha(\beta + 1)$-tight after a level down event. However, the value of $h$ is bounded by $n$, and $h$ appears inside $\alpha$, so that the approximation ratio of the scheme may become $n$ in the worst-case. Fortunately, with the help of the following lemma, we can choose $\alpha$ carefully, which in turn improves the approximation ratio from $n$ to $O(1)$.

Lemma 19. Suppose that we set $\alpha \geq \beta/(\beta - 1)$. By the time a level down event occurs at $v$ at moment $t_0$, exactly one copy of $v$ is selected. That is, $|\delta^0(v)|/k_v = 1$.

Proof. Assume to the contrary that $v$ could decrease its level even if more than one copy of $v$ is selected. Since $v$ undergoes level down, its weight $W_v$ must have decreased; this can happen only in one of the following cases:

Case 1: An incident edge whose level is in the range $[0, \ell(v)]$ is deleted. In this case, since more than one copy of $v$ is selected, $W_v$ is unchanged. Thus, this case cannot happen.

Case 2: An incident edge whose level is in the range $[\ell(v) + 1, L]$ is deleted. In this case, the weight $W^{t_0}_v$ at moment $t_0$ is less than $c_v^*$. On the other hand, at the moment $t'$ when $v$ attains the current level $\ell(v)$ (from level $\ell(v) - 1$), its weight $W^{t'}_v$ was at least $c_v$ before level up, and became at least $c_v/(\beta + 1)$ after the level up. (The reason is from the proof of Lemma 6: the weight change between consecutive levels is at most a factor of $\beta + 1$.) This implies that:

$$c_v^* > W^{t_0}_v \geq k_v \mu \beta^{-\ell(v)} \quad \therefore \text{more than one copy of } v \text{ is selected}$$

Combining, we would have

$$c_v^*/\alpha(\beta + 1) = c_v^*/k_v \mu \beta^{-\ell(v)} \geq \frac{c_v(\beta - 1)}{\beta(\beta + 1)},$$

so that $\alpha < \beta/(\beta - 1)$. A contradiction occurs.

Thus, the lemma follows.

The above lemma states that if we choose $\alpha \geq \beta/(\beta - 1)$, then level down of $v$ occurs only when $|\delta^0(v)|/k_v$ is one. Then, Case 2 inside the proof of Lemma 14 will not occur, so that we can strengthen Lemma 14 to get $|\Delta \text{COUNT}| \leq D^{\text{up}}_0(0, i) < \beta^2 c_v^*/\mu$. Similarly, the proof of Lemma 17 can be revised, so that we can strengthen Lemma 17 by replacing $h$ with one. On the other hand, we need $\alpha \geq (2\beta + 1)/\beta + 2\epsilon$ to satisfy the amortized cost analysis. Consequently, we set $\alpha = (2\beta + 1)/\beta + 2\epsilon$, and we can achieve the desired bound $\Delta B \geq \beta^2 c_v^*/\mu \geq |\Delta \text{COUNT}|$. The proof for the level down case is complete.

3.3 Summary and extensions

With the appropriate setting of $\alpha = (2\beta + 1)/\beta + 2\epsilon$, where $0 < \epsilon < 1$, we get an $\alpha(\beta + 1)$-tight level scheme. Then, by setting $\beta = 2.43$, Theorem 8 is proven so that we get an approximation solution of ratio close to 36 with $O((\log n)/\epsilon)$ amortized update time. Note that if we focus on the non-capacitated case, that is, each vertex is weighted and has unlimited capacity,
the problem becomes the weighted vertex cover problem. Our dynamic scheme can easily be adapted to maintain an approximate solution, based on the following changes. First, we define the weight of a vertex \( W_v \) as \( W_v = \sum_{e \sim v} \mu \beta^{-\ell(e)} \). Next, we let \( \alpha = 1 + 3\epsilon \) and \( \beta = 1 + \epsilon \) and revise \( \phi(e) \) as \( \phi(e) = (1 + \epsilon)(L - \ell(e)) \). After these changes, we can go through a similar analysis, and obtain a \((2 + \epsilon)\)-approximate weighted vertex cover with \( O(\log n/\epsilon^2) \) amortized update time.

Finally, we consider two natural extensions of the capacitated vertex cover problem, and show how to adapt the proposed level scheme to handle these extensions.

**Capacitated set cover.** First, we consider the capacitated set cover problem, which is equivalent to the capacitated vertex cover problem in hyper-graphs. A hyper-graph \( G = (V,E) \) has \(|V| = n\) vertices and \(|E| = m\) hyper-edges, where each hyper-edge is incident to a set of vertices. Suppose each hyper-edge is incident to at most \( f \) vertices. In this case, we can obtain a level scheme that maintains an \( O(f^2) \) approximation solution to the dynamic set cover problem with \( O(f \log(m + n)/\epsilon) \) amortized update time.

**Capacitated vertex cover with non-uniform unsplittable demand.** Next, we consider a more general case of the capacitated vertex cover problem in which each edge has an unsplittable demand. That is, the demand of each edge must be covered by exactly one of its endpoints. In this case, we found that it is difficult to adapt the proposed level scheme and derive similar results as before. Briefly speaking, one may re-design the weight of a vertex \( W_v \) to keep the approximation ratio, by then it becomes hard to cope with the edge insertion and deletion and maintain the \( O(\log n) \) amortized update time. Nevertheless, we present two simple algorithms, one with \( O(\log_2 d_{\max}) \) approximation ratio and \( O(\log k_{\max}/\epsilon) \) amortized update time, where \( d_{\max} = \max_e\{d_e\} \), \( k_{\max} = \max_v\{k_v\} \), and another with \( O(1) \) approximation ratio and \( O(d_{\max} \log k_{\max}/\epsilon) \) amortized update time, by re-using our proposed scheme for capacitated vertex cover.

### 4 Concluding Remarks

We have extended dynamic vertex cover to the more general WMCVC problem, and developed a constant-factor dynamic approximation algorithm with \( O(\log n/\epsilon) \) amortized update time, where \( n \) is the number of the vertices. Note that, with minor adaptions to the greedy algorithm reported in Gupta et al.’s very recent paper [10] is also able to work for the dynamic capacitated vertex cover problem, but only to obtain a logarithmic-factor approximation algorithm with \( O(\log n) \) amortized update time. Moreover, our proposed algorithm can also be extended to solve the soft capacitated set cover problem, and the capacitated vertex cover problem with non-uniform unsplittable edge demand.

We conclude this paper with some open problems. First, recall that in the static model, the soft capacitated vertex cover problem [9] can be approximated within a factor of two and three for the uniform and non-uniform edge demand cases, respectively. Here, we have shown that it is possible to design a dynamic scheme with \( O(1) \) approximation ratio with polylogarithmic update time for the uniform edge demand case. Thus, designing an \( O(1) \)-approximation ratio algorithm with \( O(\log k_{\max}) \), or polylogarithmic, update time for the non-uniform edge demand case seems promising.

Recall that in the non-capacity case, both of [4, 10] achieved a large constant approximation ratio (\( \approx 1000 \)) with \( O(1) \) amortized update time. However, when applying their approaches directly, it seems hard to remove the coefficient \( h \), so that the approximation ratio may be
up to $O(n)$. On the other hand, very recently, Bhattacharya et al. [3] derived a scheme with polylogarithmic worst-case update time and $(2 + \epsilon)$-approximation ratio. They created six states for dynamic updates. Nevertheless, we cannot extend their approach directly, since some of these states do not satisfy the capacity constraint. It would be of significant to consider the above approaches to soft capacity vertex cover.

Another open problem is to consider hard capacity vertex cover, where most of the previous studies in the literature used different techniques, such as rounding and patching, from the primal-dual approach in this paper. It would be worthwhile to explore the dynamic model with hard capacity constraints.

References

