Online Correlation Clustering for Dynamic Complete Signed Graphs

Ali Shakiba

Department of Computer Science, Vali-e-Asr University of Rafsanjan, Rafsanjan, Iran.
ali.shakiba@vru.ac.ir; a.shakiba.iran@gmail.com

Abstract

In the correlation clustering problem for complete signed graphs, the input is a complete signed graph with edges weighted as +1 (denote recommendation to put this pair in the same cluster) or −1 (recommending to put this pair of vertices in separate clusters) and the target is to cluster the set of vertices such that the number of disagreements with these recommendations is minimized.

In this paper, we consider the problem of correlation clustering for dynamic complete signed graphs where (1) a vertex can be added or deleted, and (2) the sign of an edge can be flipped. In the proposed online scheme, the offline approximation algorithm in [CALM+21] for correlation clustering is used. Up to the author’s knowledge, this is the first online algorithm for dynamic graphs which allows a full set of graph editing operations.

The proposed approach is rigorously analyzed and compared with a baseline method, which runs the original offline algorithm on each time step. Our results show that the dynamic operations have local effects on the neighboring vertices and we employ this locality to reduce the dependency of the running time in the Baseline to the summation of the degree of all vertices in $G_t$, the graph after applying the graph edit operation at time step $t$, to the summation of the degree of the changing vertices (e.g. two endpoints of an edge) and the number of clusters in the previous time step. Moreover, the required working memory is reduced to the square of the summation of the degree of the modified edge endpoints rather than the total number of vertices in the graph.

Keywords: Correlation Clustering, Online Algorithms, Signed Graph

1 Introduction

The goal of clustering, as a long-standing problem in machine learning, is to divide a set of elements such that in each partition or cluster, the elements are similar to each other and dissimilar to the elements in other clusters. The problem of correlation clustering is proposed by Bansal, Blum, and Chawla in [BBC04] and found many applications in disambiguation tasks [KCMNT08], community detection [CSX12], cluster ensembles [BGU13], due to its natural simple formulation. In the original correlation clustering formulation, the input is a weighted graph with positive and negative edge weights. The negative edge weights are used to represent dissimilarity between vertices whereas the positive edge weights are used to represent the similarity between two vertices. Then, correlation clustering asks for clustering this graph such that the sum of the absolute values of the weights of the negative edges inside any cluster plus the sum of the weight of the positively weighted edges between any clusters is minimized. Unfortunately, this problem in both maximization and minimization variants is known to be \textsf{NP}-hard. A simple variant of this problem is well-studied in the literature: the weights of the edges are either +1 or −1, i.e. the underlying graph is signed. The maximization instance of this variant is shown to have a polynomial-time approximation scheme in [BBC04]. Moreover, in [CMSY15], a 2.06-approximation algorithm is given for the minimization version of this variant. In [GG05], an approximation algorithm with polynomial running time is given when the number of clusters is bounded to be no more than $k$ clusters.
There is also an $O(\log n)$-approximation for the minimization instance of the problem where the weights are arbitrary and $n$ is the number of vertices, due to [DEFI06]. Also, there is a constant factor approximation scheme for solving the minimization instance of this problem in [CALM+21] when the edge weights are restricted to $\{-1, +1\}$ and the underlying graph is complete.

One main restriction to most of the previously stated algorithms is their reliance on Integer Programming techniques. This makes these techniques less attractive for dynamic graphs. In [CALM+21], a $O(1)$-factor approximation algorithm for solving the correlation minimization variant in signed graphs in a semi-streaming setting is proposed which has $O(1)$ passes over the dataset. Moreover, the same algorithm is then applied in [CALMP22] in an online setting to dynamic signed graphs for complete graphs. In both the aforementioned methods, adding a vertex that reveals all its incident edges to the existing vertices is the only allowed method and after that, no edge might change its sign or a vertex might be deleted. In this paper, we propose the first algorithm, up to the author’s knowledge, for the correlation clustering problem on signed complete graphs where a full set of dynamic operations are allowed.

### 1.1 Our Contribution

Our main contribution is analyzing the behavior of the CorrelationClustering algorithm of [CALM+21] in a full dynamic graph setting where one can add/remove vertices and flip edge signs. This is not the first extension of the CorrelationClustering algorithm to dynamics graphs, as up to the author’s knowledge, it is extended to accomplish consistent clustering for dynamic graphs in [CALMP22]. However, the extension in [CALMP22] only allows adding vertices that reveal all of its edges to existing vertices, whereas in our setting, any change is possible. The main idea in our algorithm is tracking the changes to the underlying graph, which as we will show, is local. Our most important result is the following theorem.

**Theorem 1.** The proposed OnlineCorrelationClustering algorithm (Algorithm 3) maintains the same clustering as the clustering by the offline CorrelationClustering algorithm (Algorithm 1 and described as Baseline in Section 3) on each time step $t = 1, 2, \ldots$ with the following time and space complexities:

- **Adding and deleting vertices with all negative signed incident edges** can be accomplished in constant time with constant extra memory.

- **Flipping the sign of an edge** $e = \{u, v\}$ requires

  $$O \left( |V_t| + |E_t^+| + \left( \deg_{G_t^+}(u) + \deg_{G_t^+}(v) \right)^2 + |G_{t-1}| \right),$$

  time with $O \left( (\deg_{G_t^+}(u) + \deg_{G_t^+}(v))^2 \right)$ extra working memory.

Note that computing the initial offline clustering for the initial graph $G_0$ requires $O (|V_0| + |E_0^+| + D_0)$ time where $D_0 = \sum_{e=\{u,v\} \in E_0^+} \left( \deg_{G_0^+}(u) + \deg_{G_0^+}(v) \right)$ time and $O (|V_0|)$ offline memory.

Up to the author’s knowledge, this is the first online correlation clustering algorithm for signed graphs where edge modification in terms of its sign and vertex addition and deletion are allowed. In comparison with the Baseline algorithm, which requires $O (|V_t| + |E_t^+| + D_t)$ time where

$$D_t = \sum_{e=\{u,v\} \in E_t^+} \left( \deg_{G_t^+}(u) + \deg_{G_t^+}(v) \right),$$

and $O (|V_t|)$ memory on each time step, our proposed approach has the following advantageous:

- Using the locality of changes made during the sequence of operations, we reduced the dependency of the running time in BASELINE to the summation of the degree of all vertices in $G_t$ to the summation of the degree of the changing vertices (e.g. two endpoints of an edge) and the number of clusters in the previous time step. Note that it is expected that the number of clusters at any point in time is much smaller than the number of vertices of the graph.
• Again, using the locality effect, we were able to reduce the required working memory to

\[ O\left( \left( \deg_{G^+}(u) + \deg_{G^+}(v) \right)^2 \right), \]

compared with \( O(|V_i|) \). Note that for real-world data, we expect very low degrees compared to the number of vertices, although \( \deg_{G^+}(v) = O(|V_i|) \).

2 Preliminaries

In the min-disagree variant of the correlation clustering problem, we are given a signed undirected graph \( G = (V, E^+, E^-) \) and the goal is finding a partitioning of the vertices \( V \) into \( \mathcal{C} = \{C_1, \ldots, C_\ell\} \) which minimizes

\[ \text{cost}(\mathcal{C}) = \sum_{\{u,v\} \in E^+} 1 + \sum_{\{u,v\} \in E^-} 1, \]

where \( E^+ \) denotes the set of all edges in \( G \) whose sign is +. We use \( G^+ = G[E^+] \) to denote the induced subgraph of \( G \) with the same set of vertices and \( E^+ \) as the set of edges, i.e. \( G^+ = (V, E^+) \).

**Definition 1 (\( \varepsilon \)-agreement [CALM’21])**. Let \( G^+ = G[E^+] \), \( \varepsilon \in \mathbb{R}^+ \), and \( u, v \in V \). Then, vertices \( u \) and \( v \) are in \( \varepsilon \)-agreement if \( u \) and \( v \) are adjacent in \( G^+ \) and

\[ \text{NonAgreement}_{G}(u, v) = \frac{|N_G(u) \Delta N_G(v)|}{\max \{|N_G(u)|, |N_G(v)|\}} < \varepsilon. \]

**Definition 2 (\( \varepsilon \)-lightness [CALM’21])**. Let \( G^+ = G[E^+] \), \( \varepsilon \in \mathbb{R}^+ \), and \( u \in V \). Then, the vertex \( u \) is said to be \( \varepsilon \)-light, or simply light, in \( G^+ \) if

\[ \frac{\text{AgreeCnt}_{G^+}(u)}{|N_{G^+}(u)|} < \varepsilon, \]

where \( \text{AgreeCnt}_{G^+}(u) = |\{w \in V | u \text{ and } v \text{ are in } \varepsilon \text{-agreement}\}| \). Otherwise, it is called \( \varepsilon \)-heavy, or simply heavy.

The basic ingredient of our approach is the \textsc{CorrelationClustering} algorithm introduced in [CALM’21], which is restated in Algorithm 1. This algorithm is a constant-factor approximation scheme for solving the min-disagree correlation clustering problem for a complete undirected signed graph. Let \( C_{\text{Opt}} \) be an optimal solution with cost \( \text{OPT}_G \) (Eq. 1) and \( \mathcal{C} \) be the output of the Algorithm 1 with \( G \) and \( \varepsilon \) as its input. Then, \( \text{cost}(\mathcal{C}) \leq D + \text{OPT}_G \), where \( D \leq 1 + \frac{\beta}{\varepsilon} + \frac{1}{\varepsilon^2} \) (Theorem 3.9 in [CALM’21]). In other words, \textsc{CorrelationClustering}(\( G \)) is a \( 2 + \frac{\beta}{\varepsilon} + \frac{1}{\varepsilon^2} \)-approximation algorithm, or simple a constant factor approximation scheme. Note that in our presentation here, we substituted both \( \beta \) and \( \lambda \) with \( \varepsilon \) and considered \( i = 1 \) in comparison with [CALM’21]. Moreover, \textsc{CorrelationClustering}(\( G \)) is used in [CALMP22], renamed to \textsc{Agreement} algorithm, for solving the min-disagree correlation clustering problem in an online setting. Their online approach only allows adding a vertex to the graph, which reveals all the edges to the previously arrived nodes upon arrival. Moreover, in [CALMP22], they provide the \textsc{Online Agreement} algorithm which uses the \textsc{Correlation-Clustering} after each vertex arrival as an offline re-clustering phase. Their online approach is shown to be a constant-factor approximation algorithm, too (Theorem E.2 in [CALMP22]). An efficient implementation of the algorithms in this paper is applicable using the adjacency list-representation of the graphs. Therefore, the neighborhood of each vertex is accessible in constant-time.
3 Proposed Method

The Online Setting. Consider an initial undirected complete signed graph $G_0 = (V_0, E_0)$ and an infinite sequence of operations, one per time stamp $t = 1, 2, \ldots$ denoted as $\text{OP}_1, \text{OP}_2, \ldots$. We allow three kinds of operations: (1) $\text{FlipSign}(e)$ which flips the sign of an edge $e$, (2) $\text{AddVertex}(v)$ to add a new vertex $v$ to the graph whose all of its incident edges are negatively signed, and (3) $\text{DeleteVertex}(v)$ which deletes an existing vertex $v$ whose all of its incident edges are negatively signed. At first glance, the restriction on the vertex addition/deletion may seem restrictive, however, if one wants to delete a vertex with positively signed incident edges, it can use consecutive calls to $\text{FlipSign}(e)$ for all of its positively signed incident edges and finally, remove it. The same applies for the vertex addition.

Notation in Algorithms. For every vertex $v$ in the graph $G$, we need three quantities: (1) $v.\text{IsLight}_{G^+}$ which is a boolean value denoting whether the vertex $v$ is light or not, (2) $v.\text{AgreeCnt}_{G^+}$ as the number of vertices which are in $\varepsilon$-agreement with node $v$, and (3) $v.\text{Neigh}_{G^+}$ which is equal to $N_{G}(v)$. For each edge $e = \{u, v\}$, we also store its sign as $e.\text{Sign}$ (either + or −), and $e.\text{Agree}$ which is a boolean equal to TRUE if $u$ and $v$ are in $\varepsilon$-agreement in $G^+$, and FALSE otherwise. Note that we do not need to store all of these values explicitly and we just introduced them for simplicity and clearance in expressing the algorithms.

High-level Description of the Algorithm. The proposed online algorithm is fully described in Algorithm 1. At a high-level, we have three phases:

1. Initial clustering, where an initial graph $G_0$ is clustered into $C_0$ by running the offline algorithm CorrelationClustering($G_0, \varepsilon$) for the given parameter $\varepsilon$.

2. During the graph maintenance phase, each $\text{OP}_t$, for $t = 1, 2, \ldots$ is applied to $G_{t-1}$ and the graph $G_t$ is computed. This can be efficiently done by calling the corresponding algorithms, i.e. Algorithm 2 if $\text{OP}_t$ is flipping the sign of an edge, $\text{AddVertex}$ and $\text{DeleteVertex}$ if $\text{OP}_t$ is adding a new or deleting an existing vertex, respectively.

3. After the graph is updated, the clustering $C_{t-1}$ would be refined to calculate $C_t$. This phase is called cluster maintenance.

Time Complexity. The initial clustering phase requires $O(|V_0| + |E_0^+| + D_0)$ time where

$$D_0 = \sum_{e = \{u, v\} \in E_0^+} \left( \deg_{G_0^+}(u) + \deg_{G_0^+}(v) \right),$$

(Theorem 2). For each $\text{OP}_t$, for $t = 1, 2, \ldots$, the time complexity of the graph maintenance phase depends on the operation performed: (1) Adding/deleting a vertex by a call to $\text{AddVertex}/\text{DeleteVertex}$ can be accomplished in time $O(1)$ (Theorem 7). (2) Flipping the sign of a vertex by a call to $\text{FlipSign}(e = \{u, v\})$ is of time complexity $O \left( |V_t| + |E_t^+| + \left( \deg_{G_t^+}(u) + \deg_{G_t^+}(v) \right)^2 \right)$ (Theorem 3).
Algorithm 2 The OnlineCorrelationClustering algorithm

1: procedure OnlineCorrelationClustering(\(G_0, \varepsilon\))
2:    Initialization
3:    \(C_0 \leftarrow \text{CorrelationClustering}(G_0, \varepsilon)\) \(\triangleright\) Algorithm 1
4:    Run for each \(\text{Op}_t\) for \(t = 1, 2, \ldots\)
5:    if \(\text{Op}_t\) is \text{FlipSign}(e) then
6:        Call \text{FlipSign}(e) \(\triangleright\) Algorithm 5
7:    else if \(\text{Op}_t\) is \text{AddVertex}(v) then
8:        Call \text{AddVertex}(v) \(\triangleright\) Algorithm described on Page 12
9:    else if \(\text{Op}_t\) is \text{DeleteVertex}(v) then
10:       Call \text{DeleteVertex}(v) \(\triangleright\) Algorithm described on Page 12
11:    else
12:       return Invalid Operation \(\triangleright\) Ignoring \(\text{Op}_t\) as it is invalid
13: end if
14: end procedure

Space Complexity. As the initial clustering is applied offline, its memory of \(\mathcal{O}(|V_t|)\) (Theorem 2) does not count as the working memory required in the online phase. Combining Theorems 8 and 7, the proposed algorithm requires constant extra memory to handle vertex addition and deletion, and \(\mathcal{O}\left((\text{deg}_{G_t}(u) + \text{deg}_{G_t}(v))^2\right)\) extra memory to handle edge sign flipping.

Baseline Algorithm. The baseline algorithm, denoted as Baseline, is simply running the offline CorrelationClustering, Algorithm 4 for every \(G_t\) for \(t = 1, 2, \ldots\). The time complexity of running CorrelationClustering for one time is as follows: (1) Constructing the graph \(G^+\) can be accomplished in \(\mathcal{O}(|E^+|)\). (2) Discarding all edges of \(G^+\) whose endpoints are not in \(\varepsilon\)-agreement can be done by \(\mathcal{O}(|E^+|)\) evaluation of NonAgreement for two vertices. (3) Deciding on the lightness of vertices can be accomplished in \(\mathcal{O}(|V_t|)\) and the edges whose both endpoints are light can be discarded again in \(\mathcal{O}(|E^+|)\). Finally, finding the connected components of the sparsified graph can be implemented in time \(\mathcal{O}(|V_t| + |E^+_t|)\). To sum it up, each call to CorrelationClustering\((G_t, \varepsilon)\) is of time \(\mathcal{O}(|V_t| + |E^+_t| + D_t)\) where \(D_t = \sum_{e=(u,v) \in E^+} (\text{deg}_{G_t}(u) + \text{deg}_{G_t}(v))\) and \(G_t = (V_t, E_t)\). Moreover, the baseline algorithm requires \(\mathcal{O}(|V_t|)\) working memory in addition to the memory required for representing the graph \(G_t\): Computing the NonAgreement\(_{G^+}(u, v)\) for each edge \(e = \{u, v\} \in E^+_t\) can be accomplished in \(\mathcal{O}(\text{deg}_{G_t}(u) + \text{deg}_{G_t}(v))\) space, which is linear in the number of vertices in \(G_t\) and can be reused for each edge. There is no need to store the NonAgreement of an edge, however, it is required to store the degree of each vertex \(v\) in \(G^+_t\) as well as the number of adjacent vertices in \(G^+_t\) which are in \(\varepsilon\)-agreement with \(v\). Finally, representing the clustering \(C_t\) requires \(\mathcal{O}(|V_t|)\) space.

Theorem 2. The offline CorrelationClustering\((G_t, \varepsilon)\), Algorithm 7 is of time complexity \(\mathcal{O}(|V_t| + |E^+_t| + D)\) where

\[ D = \sum_{e=(u,v) \in E^+_t} (\text{deg}_{G_t}(u) + \text{deg}_{G_t}(v)), \]

and requires \(\mathcal{O}(|V_t|)\) memory plus the memory required to represent the graph \(G_t\).

Analysis. Let \(C_t\) be the output clustering of the Baseline algorithm with \(G_t\) as input. Also, let \(C_t'\) be the clustering given by running OnlineCorrelationClustering after \(\text{Op}_t\). We claim that \(C_t = C_t'\). As discussed in Sections 3.1.3 and 3.3, the graph \(G^+_t\) maintained by the procedures \text{FlipSign}, \text{AddVertex}, and \text{DeleteVertex} is the same as the graph \(G^+_t\) computed in Baseline with the same values of \(\varepsilon -\)
NonAgreement and the same set of $\varepsilon$-lightness vertices. We have also discussed that the clustering maintenance methods described in Sections 3.2 and 3.3 give the same output as the Baseline there.

3.1 Edge Sign Flip

Let $e = \{u, v\}$ be an existing edge in the graph $G$. Then, the procedure FLIPSIGN($e$), Algorithm 5 changes the sign of the edge $e$ and maintains the clustering. Let use $H$ to denote the resulting graph after flipping the sign of the edge $e$. By combining Theorems 4 and 6 we can state the following theorem.

**Theorem 3.** The procedure FLIPSIGN($e = \{u, v\}$) stated in Algorithm 5 flips the sign of the edge $e$ and maintains the graph and the clustering with

$$O\left(|V_t| + |E_t^+| + \left(\deg_{G_t^+}(u) + \deg_{G_t^+}(v)\right)^2\right),$$

time using $O\left(\left(\deg_{G_t^+}(u) + \deg_{G_t^+}(v)\right)^2\right)$ working memory in addition to the memory required for representing clustering output, e.g. $C_t$.

3.1.1 Flipping from $-$ to $+$

Suppose that FLIPSIGN($e = \{u, v\}$) flips the sign of an edge $e$ from $-$ to $+$. As it is illustrated in Figure 1, this change will have two different effects on the $\varepsilon$-agreement of neighboring vertices. If a vertex $w$ belongs to the neighborhood of both $u$ and $v$, the effect of edge sign flipping would be different compared to the case where $w$ belongs to either neighborhood and not the other.

![Figure 1: The effect of calling FLIPSIGN($e = \{u, v\}$) on the graph $G$ where the sign of edge $e$ is flipped from $-$ to $+$: (a) $G^+$ (b) $H^+$ (c) $G^+$ (d) $H^+$.

**Lemma 1.** Suppose the sign of edge $u = \{u, v\}$ is changed from $-$ to $+$ by a single call to FLIPSIGN$_G(e)$. Then,

1. If $w \in N_{G^+}(u) \cap N_{G^+}(v)$, then NonAgreement$_{G^+}(u, w) \geq$ NonAgreement$_{H^+}(u, w)$.
2. If $w \in N_{G^+}(u) \setminus N_{G^+}(v)$, $|N_{G^+}(u)| < |N_{G^+}(w)|$, and $w \neq v$, then NonAgreement$_{G^+}(u, w) <$ NonAgreement$_{H^+}(u, w)$.
3. Let $\text{Threshold} = |N_{G^+}(u) \Delta N_{G^+}(w)|$. If $w \in N_{G^+}(u) \setminus N_{G^+}(v)$, $|N_{G^+}(u)| \geq |N_{G^+}(w)|$, and $w \neq v$. Then,

   (a) $\text{Threshold} < |N_{G^+}(u)|$ implies NonAgreement$_{G^+}(u, w) <$ NonAgreement$_{H^+}(u, w)$.

   (b) $\text{Threshold} = |N_{G^+}(u)|$ implies NonAgreement$_{G^+}(u, w) =$ NonAgreement$_{H^+}(u, w)$.

   (c) $\text{Threshold} > |N_{G^+}(u)|$ implies NonAgreement$_{G^+}(u, w) >$ NonAgreement$_{H^+}(u, w)$.

4. Let $w \notin N_{G^+}(u) \cup N_{G^+}(v)$, then NonAgreement$_{G^+}(x, w) =$ NonAgreement$_{H^+}(x, w)$ for $x \in N_{G^+}(w)$.
At first, it seems that maintaining the graph $G^t$ and the status of $\varepsilon$-agreement after an edge sign flipping may require a large amount of processing. However, as we show in the following two propositions, flipping the sign of a vertex has local effects.

**Proposition 1.** Suppose the sign of an edge $u = \{u, v\}$ is changed from $-\varepsilon$ to $+\varepsilon$ by a single call to FLIPSIGN$_G(e)$ and $H$ is the resulting graph. Let $w \in N_{G^+}(u)$ and \textsc{threshold} = $|N_{G^+}(u)\Delta N_{G^+}(w)|$.

1. If $u$ and $w$ are in $\varepsilon$-agreement in $G^+$, then they are in $\varepsilon$-agreement in $H^+$ if any of the following cases hold:
   
   (a) $w \in N_{G^+}(u) \cap N_{G^+}(v)$.
   
   (b) $w \in N_{G^+}(u) \setminus N_{G^+}(v)$, $w \neq v$, $|N_{G^+}(u)| > |N_{G^+}(w)|$, and \textsc{threshold} \geq $|N_{G^+}(u)|$.

2. If $u$ and $w$ are not in $\varepsilon$-agreement in $G^+$, then they are not in $\varepsilon$-agreement in $H^+$ if any of the following cases hold:

   (a) $w \in N_{G^+}(u) \setminus N_{G^+}(v)$, $w \neq v$, and $|N_{G^+}(u)| < |N_{G^+}(w)|$.

   (b) $w \in N_{G^+}(u) \setminus N_{G^+}(v)$, $w \neq v$, $|N_{G^+}(u)| \geq |N_{G^+}(w)|$, and \textsc{threshold} \leq $|N_{G^+}(u)|$.

**Proposition 2.** Suppose the sign of an edge $u = \{u, v\}$ is changed from $-\varepsilon$ to $+\varepsilon$ by a single call to FLIPSIGN$_G(e)$. Let $w \notin N_{G^+}(u) \cup N_{G^+}(v)$, then the $\varepsilon$-agreement of $x$ and $w$ is the same in both $G^+$ and $H^+$ for $x \in N_{G^+}(w)$.

Propositions 1 and 2 and Lemma 1 cover all possible configurations for $\varepsilon$-agreement of two vertices $u$ and $v$ whose positive sign edge $e = \{u, v\}$ is present in $G^+$. These results are summarized in Table 1.

| Edge        | Neighborhood | $\varepsilon$-agreement $G^+$ | $\varepsilon$-agreement $H^+$ | Exact Conditions |
|-------------|--------------|-------------------------------|-------------------------------|-----------------|
| $\{x, w\}$ | $w \notin N_{G^+}(u) \cup N_{G^+}(v)$ | ✓                             | ✓                             | Propositions 2 & 1 |
| $\{x, w\}$ | $w \notin \{u, v\}$                     | ×                             | ×                             |                  |

**Table 1:** All different configurations for $\varepsilon$-agreement of any two vertices whose edge is present in $H^+$ after FLIPEDGE$(u, v)$ is called. The marks ✓, and x are used to denote that the vertices in that row are in $\varepsilon$-agreement or not. The mark ? is used to denote that identifying $\varepsilon$-agreement requires re-computation of the Eq. [2].
3.1.2 Flipping from $+$ to $-$

Similar to the flipping the sign of an edge from $-$ to $+$, we can obtain a similar set of results, however with different conditions.

**Lemma 2.** Suppose the sign of an edge $u = \{u, v\}$ is changed from $+$ to $-$ by a single call to $\text{FlipSign}_{G}(e)$. Then,

1. If $w \in N_{G^+}(u) \cap N_{G^+}(v)$, then $\text{NonAgreement}_{H^+}(u, w) > \text{NonAgreement}_{G^+}(u, w)$.
2. If $w \in N_{G^+}(u) \setminus N_{G^+}(v)$, $|N_{G^+}(u)| \leq |N_{G^+}(w)|$, and $w \neq v$, then $\text{NonAgreement}_{H^+}(u, w) < \text{NonAgreement}_{G^+}(u, w)$.
3. Let $\text{Threshold} = |N_{G^+}(u) \Delta N_{G^+}(w)|$. If $w \in N_{G^+}(u) \setminus N_{G^+}(v)$, $|N_{G^+}(u)| > |N_{G^+}(w)|$, and $w \neq v$. Then,
   
   (a) $\text{Threshold} < |N_{G^+}(u)|$ implies $\text{NonAgreement}_{G^+}(u, w) > \text{NonAgreement}_{H^+}(u, w)$.
   
   (b) $\text{Threshold} = |N_{G^+}(u)|$ implies $\text{NonAgreement}_{G^+}(u, w) = \text{NonAgreement}_{H^+}(u, w)$.
   
   (c) $\text{Threshold} > |N_{G^+}(u)|$ implies $\text{NonAgreement}_{G^+}(u, w) < \text{NonAgreement}_{H^+}(u, w)$.
4. Let $w \notin N_{G^+}(u) \cup N_{G^+}(v)$, then $\text{NonAgreement}_{G^+}(x, w) = \text{NonAgreement}_{H^+}(x, w)$ for $x \in N_{G^+}(w)$.

As we will show in the following two propositions, the effect of edge sign flipping from $+$ to $-$ is local similar to flipping the edge sign from $-$ to $+$, although has different effects.

**Proposition 3.** Suppose the sign of an edge $u = \{u, v\}$ is changed from $+$ to $-$ by a single call to $\text{FlipSign}_{G}(e)$. Let $w \in N_{G^+}(u)$ and $\text{Threshold} = |N_{G^+}(u) \Delta N_{G^+}(w)|$.

1. If $u$ and $w$ are in $\varepsilon$-agreement in $G^+$, then they are in $\varepsilon$-agreement in $H^+$ if any of the following cases hold:
   
   (a) $w \in N_{G^+}(u) \setminus N_{G^+}(v)$, $w \neq v$, and $|N_{G^+}(u)| \leq |N_{G^+}(w)|$.
   
   (b) $w \in N_{G^+}(u) \setminus N_{G^+}(v)$, $w \neq v$, $|N_{G^+}(u)| > |N_{G^+}(w)|$, and $\text{Threshold} \leq |N_{G^+}(u)|$.
2. If $u$ and $w$ are not in $\varepsilon$-agreement in $G^+$, then they are not in $\varepsilon$-agreement in $H^+$ if any of the following cases hold:
   
   (a) $w \in N_{G^+}(u) \cap N_{G^+}(v)$.
   
   (b) $w \in N_{G^+}(u) \setminus N_{G^+}(v)$, $w \neq v$, $|N_{G^+}(u)| > |N_{G^+}(w)|$, and $\text{Threshold} > |N_{G^+}(u)|$.

**Proposition 4.** Suppose the sign of an edge $u = \{u, v\}$ is changed from $+$ to $-$ by a single call to $\text{FlipSign}_{G}(e)$. Let $w \notin N_{G^+}(u) \cup N_{G^+}(v)$, then the $\varepsilon$-agreement of $x$ and $w$ is the same in both $G^+$ and $H^+$ for $x \in N_{G^+}(w)$.

3.1.3 Maintenance Algorithm for $G^+$

After building our basic tools in Sections 3.1.2 and 3.1.1, we are ready to state an algorithm for maintaining the graph $G^+$. We can benefit from the locality of the effect of the $\text{FlipSign}$, as we show in the following corollary.

**Corollary 1.** Suppose the sign of an edge $u = \{u, v\}$ is changed by a single call to $\text{FlipSign}_{G}(e)$. Then, $\text{AgreeCnt}_{G^+}(w) = \text{AgreeCnt}_{H^+}(w)$ for all $w \in V(G) \setminus S$ where $S = N_{G^+}(u) \cup N_{G^+}(v) \cup \{u, v\}$.
In other words, Corollary 1 states that after a call to FlipSign\(_G(v)\), we just need to take care of the AgreeCnt\(_{H^+}(w)\) in the subgraph induced by set \(S\), e.g. \(w \in S\).

The neighborhood of a vertex \(w \in V\) in \(G^+\), i.e. \(N_{G^+}(w)\), can be partitioned into the following 4 parts: (1) \(A = N_{G^+}(w) \cap N_{G^+}(u) \cap N_{G^+}(v)\), (2) \(B = N_{G^+}(w) \cap (N_{G^+}(u) \setminus N_{G^+}(v))\), (3) \(C = N_{G^+}(w) \cap (N_{G^+}(v) \setminus N_{G^+}(u))\), and (4) \(D = N_{G^+}(w) \setminus (N_{G^+}(u) \cup N_{G^+}(v))\). If the vertex \(w\) and a vertex in \(D\) are in \(\varepsilon\)-agreement in \(G^+\), then, they would be in \(\varepsilon\)-agreement in \(H^+\), too. This is implied by the Proposition 1 (the second case) and the Proposition 4, i.e. the \(\varepsilon\)-agreement of \(w\) and the vertices in the set \(D\) is equal in both \(G^+\) and \(H^+\). However, for the other three parts, \(A\), \(B\) and \(C\), we need to consider the exact flip:

- to +: The \(\varepsilon\)-agreement between the vertex \(w\) and the vertices in either sets \(B\) and \(C\) requires verification, depending on extra conditions (Table 1 rows 3-10). However, the \(\varepsilon\)-agreement of the vertex \(w\) with vertices of \(A\) in \(G^+\) implies their \(\varepsilon\)-agreement in \(H^+\) by the Proposition 2.

+ to -: By the Table 1 the \(\varepsilon\)-agreement between the vertex \(w\) and the vertices in either sets \(A\), \(B\) and \(C\) requires verification, except for the vertices in \(A\) which are not in \(\varepsilon\)-agreement with \(w\) in \(G^+\), remain in non-\(\varepsilon\)-agreement with \(w\) in \(H^+\).

Considering this discussion, we can state Theorem 4.

**Theorem 4.** Suppose the sign of edge \(f = \{u, v\}\) is changed by a single call to FlipSign\(_G(v)\) and Let \(S = N_{G^+}(u) \cup N_{G^+}(v) \cup \{u, v\}\).

1. For all vertices \(w \notin S\), if \(w\) is \(\varepsilon\)-light in \(G^+\), then it would remain \(\varepsilon\)-light in \(H^+\), too.

2. For all vertices \(w \notin S\), if \(w\) is not a \(\varepsilon\)-light vertex in \(G^+\), then it would not be \(\varepsilon\)-light in \(H^+\), also.

3. The \(\varepsilon\)-lightness of vertices in \(S\) requires re-computation and verification in \(H^+\).

**Algorithm Intuition.** The major role of the Algorithms 3 and 6 is keeping the lightness of vertices valid during the FlipSign updates. The naive way of doing this requires a re-computation of all \(\varepsilon\)-agreements throughout the updated graph. However, using Corollary 1 and Theorem 4 we know that the lightness of vertices, except the ones in the neighborhood of the edge endpoints, does not change in a graph after a single call to FlipSign. In other words, the changes are only in the subgraph with vertex set \(S = \{u, v\} \cup N_{G^+}(u) \cup N_{G^+}(v)\). Therefore, both the Algorithms 3 and 6 need to recompute the number of vertices in \(\varepsilon\)-agreement with the set \(S\). To accomplish this efficiently, these Algorithms use Table 1 and apply a pruning strategy. Note that while verifying the \(\varepsilon\)-agreement of two vertices by calling VerifyEdge (Algorithm 4), the number of vertices in \(\varepsilon\)-agreement with the edge endpoints is updated. This is later used to update the lightness of vertices in \(S\).

After updating the graph \(G^+\) to accommodate the flipped edge, we need to update the underlying clusters. We will defer this discussion to Section 5.2 However, for now we just call the MaintainClusteringAfterFlipSign algorithm.

**Time Complexity.** It is easy to verify that evaluating the NonAgreement\(_{G^+}(u, v)\) requires \(O(\deg_{G^+}(u) + \deg_{G^+}(v)) = O(\Delta_{G^+})\), operations at the worst-case, where \(\Delta_{G^+}\) is the maximum degree of the graph \(G^+\). Therefore, calling VerifyEdge\((f = \{u, v\})\) is of \(O(\deg_{G^+}(u) + \deg_{G^+}(v))\) or \(O(\Delta_{G^+})\) time complexity, in the worst case.

Lines 4-6 in Update\(_G^+\)NegativeToPositive (Algorithm 3) can be accomplished in time \(O(\deg_{G^+}(u) + \deg_{G^+}(v))\). The For loop on lines 7-15 calls the VerifyEdge procedure for at most \(O(|A|)\) times. Moreover, the For loop on lines 18-32 calls the VerifyEdge procedure for at most \(O(|B|)\) and \(O(|C|)\) times for \(X = B\) and \(X = C\), respectively. As \(A\), \(B\) and \(C\) are a partition of \(N_{G^+}(u) \cup N_{G^+}(v)\), the procedure Update\(_G^+\)NegativeToPositive calls VerifyEdge for at most \(O(\deg_{G^+}(u) + \deg_{G^+}(v))\) times. Lines 35-41 in Update\(_G^+\)PositiveToNegative requires an evaluation of NonAgreement\(_{G^+}(u, v)\), which can be accomplished in \(O(\deg_{G^+}(u) + \deg_{G^+}(v))\) time. Moreover, lines 42-45 requires \(O(|S|)\) time where \(|S| = O(\deg_{G^+}(u) + \deg_{G^+}(v))\). As the same analysis applies to Update\(_G^+\)PositiveToNegative (Algorithm 6), we can state Theorem 5.
Space Complexity. A careful examination of the FlipSign\(e = \{u,v\}\) algorithm shows that this algorithm requires \(O\left(\deg_{G^+}(u) + \deg_{G^+}(v)\right)\) working memory in addition to the memory required for representing the graph \(G^+[S]\) where \(S = N_{G^-}(u) \cup N_{G^-}(v) \cup \{u,v\}\): (1) There is no need to construct the graph \(G^+\), as we can simply substitute \(G\) with \(G^+\), every edge not present in \(G^+\) has a negative sign. (2) Moreover, we do not need to load the complete graph \(G^+\) into memory, thanks to the locality of edge flipping (Theorem 3), we just need to load \(G^+[S]\). (3) We need to store the set \(S\), which in the worst-case can require \(O\left(|V|\right)\) memory (consider a complete graph with all positive edges). As this worst-case scenario is very pessimistic, we use the size of the set \(S\), i.e. \(O\left(\deg_{G^+}(u) + \deg_{G^+}(v)\right)\) memory. (4) Verifying the \(\varepsilon\)-agreement between two vertices also requires \(O\left(\deg_{G^+}(u) + \deg_{G^+}(v)\right)\) memory which is reused for every edge in \(G^+[S]\). Note that the time and space complexity stated in Theorem 5 only applies to maintaining the graph \(G^+_t\). The time and space required for maintaining the clustering is stated in Theorem 6.

**Theorem 5.** Suppose the sign of an edge \(f = \{u,v\}\) is changed by a single call to FlipSign\(G(e)\). Then, updating the \(\varepsilon\)-lightness of vertices in \(G\) can be accomplished in \(O\left((\deg_{G^+}(u) + \deg_{G^+}(v))^2\right)\) time and \(O\left(\deg_{G^+}(u) + \deg_{G^+}(v)\right)\) working memory in addition to the memory required for representing \(G^+[S]\) where \(S = N_{G^+}(u) \cup N_{G^+}(v) \cup \{u,v\}\).

### 3.2 Maintaining Clusters in Edge Sign Flip

Let \(C_t\) be the clustering at time step \(t\) for \(t = 0,1,\ldots\). Also, let \(O_1, O_2, \ldots\) be a sequence of operations where \(O_t\) is applied to the graph \(G_{t-1}\) and \(G_t\) is its result.

**Definition 3** (**\(\varepsilon\)-critical**). Let \(G\) be a signed graph and \(G^+ = G[E^+]\) and assume that \(\overline{G^+}\) be the sparsified graph of \(G^+\) (Line 5 of the Algorithm 1). Then, a vertex \(v\) (edge \(e\)) whose removal modifies the connected components in \(\overline{G^+}\) is called an \(\varepsilon\)-critical vertex (edge).

Note that we used the term “modifying the connected components”, rather than talking about the change in the number of components in Definition 3. Assume the graph in Figure 2 with the following setting: all the edges have positive signs, vertices \(v, w, x, y, z\) are \(\varepsilon\)-light, and vertex \(u\) is not \(\varepsilon\)-light. Moreover, assume that any path between vertices \(x\) and \(y\) passes the vertex \(u\), any path between vertices \(w\) and \(z\) passes the vertex \(v\) and any path between \(w\) and \(z\) passes through the edge \(\{u,v\}\). In this setting, the edges \(\{v,w\}\) and \(\{v,z\}\) would be removed in the corresponding sparsified graph. Without loss of generality, assume that the sparsified graph consisted of 3 components. It is clear that flipping the sign of edge \(\{u,v\}\) would result in its removal. However, consider removing the edge \(\{u,v\}\): it makes \(u\) an \(\varepsilon\)-light vertex, however changes the vertex \(v\) to a non-\(\varepsilon\)-light vertex. In this case, the edges \(\{x,u\}\) and \(\{y,u\}\) would be removed in the corresponding sparsified graph whereas the edges \(\{v,w\}\) and \(\{v,z\}\) remain. In this case, the number of connected components remains unchanged, e.g. 3, but the underlying clustering is changed.

![Figure 2](image.png)

Figure 2: A \(\varepsilon\)-critical edge whose removal does not change the number of connected components in the sparsified graph.

Moreover, it is notable that any cut edge of \(G^+\) is a \(\varepsilon\)-critical edge, however, there may be an edge \(e = \{u,v\}\) which is not a cut edge of \(G^+\), but is \(\varepsilon\)-critical. For example, consider the graph \(G^+\) in Figure 1a.
a), consisting of a single connected component, where vertices \( w \) and \( x \) are both \( \varepsilon \)-light and \( u \) and \( v \) are not \( \varepsilon \)-light. Also assume that by removing the edge \( e = \{ u, v \} \) the vertices \( u \) and \( v \) would become \( \varepsilon \)-light. In this case, the edge \( e = \{ u, v \} \) is not a cut, however, if deleted, the components \( C_u, C_v, C_w \) and \( C_x \) will not be connected in the corresponding sparsified graph. The same holds for \( \varepsilon \)-critical vertices, i.e. a cut vertex of \( G^+ \) is always \( \varepsilon \)-critical, however, the converse is not necessarily true. For example, in Figure 3(b), the vertex \( y \) is not a cut vertex, but assuming that its deletion causes vertices \( u \) and \( v \) to become an \( \varepsilon \)-light vertex. Prior to deleting the vertex \( y \), assume that vertex \( w \) was \( \varepsilon \)-light. Then, removal of \( y \) causes removal of the edges \( \{ u, w \} \) and \( \{ v, w \} \), which causes separation of \( C_w \) from the rest of the graph.

Figure 3: Counterexamples where (a) \( e = \{ u, v \} \) is a \( \varepsilon \)-critical edge, but not a cut edge of \( G^+ \). (b) \( y \) is a \( \varepsilon \)-critical vertex, but not a cut vertex of \( G^+ \).

Summing up the previous discussion regarding \( \varepsilon \)-critical vertices and edges, we can state the following result.

**Lemma 3.** Let \( \text{Op}_t = \text{FlipSign}(e) \) for an edge \( e = \{ u, v \} \) and \( S = N_{G^+_{i-1}}(u) \cup N_{G^+_{i-1}}(v) \cup \{ u, v \} \). Then, for all \( w \not\in S \), the presence of all edges \( f = \{ x, w \} \) with positive sign in \( \tilde{G}^+_t \) and \( \tilde{G}^+_t \) does not change, except for the following case where a change is possible:

- \( w \) is an \( \varepsilon \)-light vertex in \( G^+_{i-1} \), and
- \( x \in S \), and
- \( w \) and \( x \) are in \( \varepsilon \)-agreement.

Moreover, we can use the following result to prune the search space.

**Lemma 4.** Let \( \text{Op}_t = \text{FlipSign}(e) \) for an edge \( e = \{ u, v \} \). If there exists a cluster \( C \in \mathcal{C}_{i-1} \) such that \( C \subseteq V \setminus \left( N_{G^+_{i-1}}(u) \cup N_{G^+_{i-1}}(v) \cup \{ u, v \} \right) \), we have \( C \in \mathcal{C}_i \).

**Maintaining Clustering After an Edge Sign Flip** The basic idea in maintaining the clustering after a \( \text{FlipSign}(e = \{ u, v \}) \) is updating the underlying graph \( G^+_i \) and then, computing the connected components of \( G^+_i[S] \) for the set \( S = N_{G^+_{i-1}}(u) \cup N_{G^+_{i-1}}(v) \cup \{ u, v \} \). Let \( D_1, \ldots, D_k \) be the connected components of
\( G_t^+ [S] \) and use \( D^w \) to denote the connected component which vertex \( w \) belongs. Moreover, we define a status ordering for clusters \( C \subseteq C_{t-1} \) as (1) \textsc{Unprocessed}, (2) \textsc{CopyCandidate}, (3) \textsc{MergeCandidate}, and (4) \textsc{SplitCandidate}. When marking, a mark would be applied only if the order of the new mark is strictly greater than the current mark. The procedure \textsc{Mark}(C, \textsc{Mark}, t-1) \textsc{Mark} to the cluster \( C \) in \( C_{t-1} \). The aim of these statuses and their ordering would become clear in the sequel. Now, we are ready to state the cluster maintenance algorithm after a \textsc{FlipSign}(\( e = \{u, v\} \)).

1. At first, we mark all clusters \( C \in C_{t-1} \) as \textsc{Unprocessed}.

2. For all clusters \( C \in C_{t-1} \) which satisfy \( C \subseteq V_t \setminus S \), we mark them as \textsc{CopyCandidate}. These clusters do not change and would be present in \( C_t \) by Lemma 4.

3. For each pair of vertices \( w, w' \in S \), mark \( C^w \) and \( C^{w'} \) as either \textsc{MergeCandidate} if \( D^w = D^{w'} \) and \( C^w \neq C^{w'} \), or \textsc{CopyCandidate} if \( D^w = D^{w'} \) and \( C^w = C^{w'} \). Moreover, just for clarity and ease of notation, assume that with each \textsc{MergeCandidate} cluster, we store the set of pairs \( (w, w') \) which caused \( C^w \) and \( C^{w'} \) to be marked as \textsc{MergeCandidate}. A set of clusters marked as \textsc{MergeCandidate} with shared vertex pairs is called a cluster group, since we intend to merge them and construct a new cluster. Moreover, if \( D^w \neq D^{w'} \) and \( C^w = C^{w'} \), then mark \( C^w \) as \textsc{SplitCandidate}. Note that when marking a cluster as \textsc{SplitCandidate}, we store its cluster group as the algorithm tries to apply \textsc{MergeCandidate} to it, although we do not change the mark.

At the end of this process, no cluster is marked \textsc{Unprocessed} in \( C_{t-1} \). Starting with an empty collection \( C_t \), we first add all the clusters of \( C \in C_{t-1} \) which are marked \textsc{CopyCandidate} to \( C_t \) with their own ids. Then, we merge each cluster group and the resulting cluster gets a new id, but we postpone its addition to \( C_t \) to the time after processing clusters with \textsc{SplitCandidate} mark. Next, for each cluster \( C \) with \textsc{SplitCandidate} mark, we need to compute the connected components of a modification of \( G_{t-1}^+ [C] \) where all edges with both endpoints in \( S \) are removed. If this subgraph is connected, then we merge \( C \) with its corresponding cluster group and continue to the next cluster with \textsc{SplitCandidate}. However, suppose there are more than one connected components, e.g. \( C_1, \ldots, C_o \). Then, for each connected component \( C_i \), if \( C_i \subseteq V_t \setminus S \), we add it directly to the \( C_t \) with a new id. Otherwise, let \( w \in C \cap S \). Then, we simply merge it to the cluster group containing a pair \( (w, x) \) for any \( x \in S \). The new cluster replaces the old merged cluster in \( C_{t-1} \) and inherits its mark and cluster group.

Whenever there is no more clusters marked \textsc{SplitCandidate} in \( C_{t-1} \), we append all the remaining clusters (which are marked \textsc{MergeCandidate}) in \( C_{t-1} \) to \( C_t \). We call this algorithm \textsc{MaintainClusteringAfterFlipSign}.

**Theorem 6.** The algorithm \textsc{MaintainClusteringAfterFlipSign} gives the same output as the Baseline at any time step \( t = 1, 2, \ldots \) with \( \text{OP}_t = \text{FlipSign}(e) \) and runs in

\[
\mathcal{O} \left( |V_t| + |E_t^+| + \left( \deg_{G_t^+}(u) + \deg_{G_t^+}(v) \right)^2 + |C_{t-1}| \right),
\]

time with \( \mathcal{O} \left( \left( \deg_{G_t^+}(u) + \deg_{G_t^+}(v) \right)^2 \right) \) working memory in addition to memory required for representing clustering \( G_t^+ \).

### 3.3 Adding and Removing Vertices

Adding or removing a vertex with all negative edges is very simple, as it has no positive edges appearing in \( G_t^+ \). To be precise, to add a new (remove an existing) vertex to the graph \( G \) with no positive incident edges, just add (remove) it. In the case of adding a new vertex \( v \), this would create a new singleton cluster consisting of that single vertex, i.e. \( C_t = C_{t-1} \cup \{v\} \). Similarly, removing an existing vertex \( v \) from \( G \) with no positive edges results in deleting its singleton cluster, i.e. \( C_t = C_{t-1} \setminus \{v\} \). Both of these operations are of constant time complexity, i.e. \( \mathcal{O}(1) \). By the way, to add a new vertex with some positive incident edges, we can first add the vertex as a vertex with no positive edges and then, add its positive edges by flipping the
corresponding edges. Similarly, if we need to delete a vertex with positive incident edges, we can first flip the sign of the positive edges and when there are no more positive edges incident to a vertex, remove the vertex. Therefore, we can state the following theorem.

**Theorem 7.** The procedure AddVertex\((v)\) (DeleteVertex\((v)\)) stated here, adds a new vertex \(v\) (deletes an existing vertex \(v\)) with all negative signed incident edges and maintains the graph and the clustering in \(O(1)\) time using \(O(1)\) working memory in addition to the memory required for representing clustering output, e.g. \(C_t\).

Note that if the Baseline algorithm runs over \(G_t\), the output is the same as \(C_t\) plus (minus) a singleton cluster containing \(v\) if we add (delete) the vertex \(v\) with all negatively signed incident edges.

### 4 Conclusion and Future Research Directions

In this paper, we proposed the first, up to the author’s knowledge, online correlation clustering algorithm for dynamic complete signed graphs with full set of operations: (1) adding/deleting vertices, and (2) flipping the edge sign. The proposed approach employs the approximation scheme of [CALM\([21]\)] and make it dynamic by studying its local behavior. The time and space complexity of the proposed approach is rigorously analyzed. In comparison with the Baseline algorithm, our proposed approach reduced the dependency of the running time in Baseline to the summation of the degree of all vertices in \(G_t\) to the summation of the degree of the changing vertices (e.g. two endpoints of an edge) and the number of clusters in the previous time step. Moreover, using the locality effect, the required working memory of the proposed method is reduced to \(O\left((\text{deg}_{G_t^n}(u) + \text{deg}_{G_t^n}(v))^2\right)\) compared with \(O(|V_t|)\). A future research direction would applying the same technique to weighted graphs. Also, improving the memory and time requirements for this problem is another research direction.

### A Appendix

**Lemma 5.** Suppose the sign of edge \(e = \{u, v\}\) is changed from \(-\) to \(+\) by a single call to FlipSign\(_G(e)\). Then,

1. \(w \in N_{G_t^n}(u)\) implies \(w \in N_{H_t^n}(u)\).
2. If \(w \in N_{G_t^n}(u) \setminus N_{G_t^n}(v)\) and \(w \neq v\), then \(|N_{H_t^n}(u) \Delta N_{H_t^n}(w)| = |N_{G_t^n}(u) \Delta N_{G_t^n}(w)| + 1\).
3. If \(w \in N_{G_t^n}(u) \cap N_{G_t^n}(v)\), then \(|N_{H_t^n}(u) \Delta N_{H_t^n}(w)| = |N_{G_t^n}(u) \Delta N_{G_t^n}(w)| - 1\).
4. We have \(|N_{H_t^n}(u) \Delta N_{H_t^n}(w)| = |N_{G_t^n}(u) \Delta N_{G_t^n}(v)| + 2\).
5. Let \(w \notin N_{G_t^n}(u) \cup N_{G_t^n}(v)\), then \(|N_{H_t^n}(x) \Delta N_{H_t^n}(w)| = |N_{G_t^n}(x) \Delta N_{G_t^n}(w)|\) for all \(x \in N_{G_t^n}(w)\).

The same holds when \(u\) and \(v\) are exchanged in statements (1) to (3).

**Proof.** (1) The first claim is clear, since just the sign of edge \(e = \{u, v\}\) is changed.

(2) By the hypothesis, we have \(w \in N_{G_t^n}(u)\), however \(w \notin N_{G_t^n}(v)\) (Figure 1(a)). So, \(v \in N_{H_t^n}(u) \Delta N_{H_t^n}(w)\) by the definition of the neighborhood operator. However, \(v \notin N_{G_t^n}(u) \Delta N_{G_t^n}(w)\) (Figure 1(b)). On the other hand, the only change made to the neighborhoods of vertices by this FlipSign is for vertices \(u\) and \(v\). Therefore, the desired result is obtained.

(3) In this case, the hypothesis implies \(v \in N_{G_t^n}(w)\), since the graph is undirected. However, \(v \notin N_{G_t^n}(u)\), and so, \(v \in N_{G_t^n}(u) \Delta N_{G_t^n}(w)\) (Figure 1(c)). By this call to FlipSign, we have \(v \in N_{H_t^n}(u)\) which implies \(v \notin N_{H_t^n}(u) \Delta N_{H_t^n}(w)\) (Figure 1(d)). Similar to the second case, the neighborhood of no other vertex is updated, so the desired result is obtained.

(4) Note that neither \(u \in N_{G_t^n}(v)\) nor \(v \in N_{G_t^n}(u)\), since the sign of the edge \(e = \{u, v\}\) is \(-\) prior
calling \textsc{FlipSign}(e). However, after calling \textsc{FlipSign}(e), we have $u \in N_{H^+}(v)$ or $v \in N_{H^+}(u)$, therefore $u, v \in N_{H^+}(u) \Delta N_{H^+}(v)$. As nothing else changes in their neighborhood in $H^+$ compared to $G^+$, the claim holds.

(5) As flipping the edge $e = \{u, v\}$ from $-\to$ to $+\to$ just changes the neighborhood of vertices $v$ and $u$, this claim holds. Note that $N_{G^+}(w) = N_{H^+}(w)$. \hfill \Box

Proof of Lemma 1

\textbf{Proof.} (1) By the third part of the Lemma 5 and the definition of $\text{NonAgreement}_X(u,w)$ for $X = G^+, H^+$, we have $|N_{H^+}(u) \Delta N_{H^+}(w)| = |N_{G^+}(u) \Delta N_{G^+}(w)| + 1$ for all $w \in N_{G^+}(u) \cap N_{G^+}(v)$, i.e. the numerator of the $\text{NonAgreement}_{H^+}(u,w)$ is decreased by one compared to the numerator for $\text{NonAgreement}_{G^+}(u,w)$. Now, there are two cases to consider: (a) $N_{G^+}(u) < N_{G^+}(w)$, and (b) $N_{G^+}(u) \geq N_{G^+}(w)$. In the first case, (a), the denominator does not change in $\text{NonAgreement}_{H^+}(u,w)$ compared to $G^+$, so the claim holds. However, on the second case, (b), the denominator would increase by one, since $v$ is the new neighbor of $u$. This way, the value of $\text{NonAgreement}_{H^+}(u,w)$ is smaller than the one in $G^+$. Therefore, the desired result is obtained.

(2) This case is very similar to the first case, since the numerator increases by one according to the second part of the Lemma 5, however, the denominator does not change. So, the desired result is obtained.

(3) This case complements two previous cases for neighbors of one of the endpoints of the flipped edge, i.e., $w \in N_{G^+}(u) \cap N_{G^+}(v), |N_{G^+}(u)| \geq |N_{G^+}(w)|$, and $w \neq v$. In this case, both the numerator and the denominator increase by one. Hence, the value of $\text{NonAgreement}_{G^+}(u,w)$ can be bigger, smaller or even equal to $\text{NonAgreement}_{H^+}(u,w)$, depending the relation of $\text{Threshold}$ and $|N_{G^+}(u)|$.

(4) The numerator for $\text{NonAgreement}_{H^+}(u,w)$ is increased by two compared to $G^+$ (Lemma 5 case (4)) and the denominator increases by one, since $v$ is added as the new neighbors of vertex $u$ in $H^+$ and vice versa. Hence, based on the relation between $\text{Threshold} = |N_{G^+}(u) \Delta N_{G^+}(v)|$ and $2 \max \{|N_{G^+}(u)|, |N_{G^+}(v)|\}$, the claim is obtained.

(5) By the fifth case of the Lemma 5, the numerator does not change in $\text{NonAgreement}_{H^+}(u,w)$ compared to $G^+$, however, the denominator might change, whether $N_{G^+}(w)$ is the maximum or $N_{G^+}(u)$. If $N_{G^+}(w)$ is the maximum, the denominator does not change, so the claim (5-a) is proved. If the maximum is $N_{G^+}(w)$, then the denominator is increased by one in $H^+$, so the claim (5-b) is obtained, too.

(6) This is the direct consequence of case (6) in Lemma 5 and the fact that no neighborhood, except for $u$ and $v$, are updated by flipping the sign of the edge $e = \{u, v\}$ from $-\to$ to $+\to$. \hfill \Box

Proof of Proposition 1

\textbf{Proof.} (1) Given that $u$ and $w$ are in $\varepsilon$-agreement in $G^+$, we know that $\text{NonAgreement}_{G^+}(u,w) < \varepsilon$. This pair of vertices remain $\varepsilon$-agreement in $H^+$ if $\text{NonAgreement}_{H^+}(u,w) < \varepsilon$, too. In the case (1-a), $\text{NonAgreement}_{H^+}(u,w) \leq \text{NonAgreement}_{G^+}(u,w) < \varepsilon$ by the first case of the Lemma 1. The last case, i.e. (1-b), hold due to case (3-c) of the Lemma 1.

(2) Since $u$ and $w$ are not in $\varepsilon$-agreement in $G^+$, we have $\text{NonAgreement}_{G^+}(u,w) \geq \varepsilon$. The case (2-a) holds by the case (2) in Lemma 1 i.e. $\text{NonAgreement}_{H^+}(u,w) \geq \text{NonAgreement}_{G^+}(u,w) \geq \varepsilon$. The case (2-b) holds by the case (3-a) of the Lemma 1. \hfill \Box

Proof of Proposition 2

\textbf{Proof.} This is a direct consequence of the definition of $\varepsilon$-agreement and the cases (4-a) to (4-c) of Lemma 1. Note that by case (4-b) of Lemma 1 $u$ and $v$ might be in $\varepsilon$-agreement in $G^+$, but not in $\varepsilon$-agreement in $H^+$. \hfill \Box

\textbf{Lemma 6.} Suppose the sign of edge $u = \{u,v\}$ is changed from $+\to$ to $-\to$ by a single call to \textsc{FlipSign}_{G}(e). Then,

1. $w \in N_{G^+}(u)$ and $w \neq v$ implies $w \in N_{H^+}(u)$.

2. If $w \in N_{G^+}(u) \setminus N_{G^+}(v)$ and $w \neq v$, then $|N_{G^+}(u) \Delta N_{G^+}(v)| = |N_{H^+}(u) \Delta N_{H^+}(w)| + 1$. 

14
3. If \( w \in N_{G^+}(u) \cap N_{G^+}(v) \), then \( |N_{G^+}(u) \Delta N_{G^+}(w)| = |N_{H^+}(u) \Delta N_{H^+}(w)| - 1 \).

4. Let \( w \notin N_{G^+}(u) \cup N_{G^+}(v) \), then \( |N_{H^+}(x) \Delta N_{H^+}(w)| = |N_{G^+}(x) \Delta N_{G^+}(w)| \) for all \( x \in N_{G^+}(w) \).

The same holds when \( u \) and \( v \) are exchanged in statements (1) to (3).

Proof. The proof is similar to the proof of Lemma 5, i.e. swapping \( G^+ \) and \( H^+ \) in the proofs.

Proof of Lemma 3

Proof. (1) By the third case of Lemma 6, we have the numerator of \( \text{NonAgreement}_{H^+}(u, w) \) increases by one compared to \( \text{NonAgreement}_{G^+}(u, w) \). The denominator does not change in \( H^+ \) if \( |N_{G^+}(w)| \geq |N_{G^+}(u)| \), otherwise it would also decrease by one. So, the \( \text{NonAgreement}_{H^+}(u, w) > \text{NonAgreement}_{G^+}(u, w) \).

(2,3) The proof of these parts is very similar to the second and the third parts in Lemma 1. Note the slight difference in the hypothesis of these parts compared to the case 2 and 3 in Lemma 1 respectively.

(4) This is the direct implication of the last part of Lemma 6 and the fact that no neighborhood other than \( u \) and \( v \) have changes in \( H^+ \).

Proof of Proposition 3

Proof. The proof is similar to the proof of the Proposition 1 using Lemma 2.

Proof of Proposition 4

Proof. This can be shown by a similar argument to the Proposition 2 based on the Lemma 2.

Proof of Corollary 1

Proof. Let the sign of edge \( e \) changed from \(-\) to \(+\). Assume that \( w \notin S \) and \( x \) be an arbitrary neighbor of \( w \) in \( G^+ \), i.e. \( x \in N_{G^+}(x) \). Given that \( x \) and \( w \) are in \( \varepsilon \)-agreement in \( G^+ \), then by the Proposition 2 case (4), they would be in \( \varepsilon \)-agreement in \( H^+ \), too. If they are not in \( \varepsilon \)-agreement in \( G^+ \), nothing changes in their \( \varepsilon \)-agreement in \( H^+ \). Moreover, the degree of \( w \) in \( G^+ \) and \( H^+ \) is the same. Therefore, the claim is proved for flipping from \(-\) to \(+\). A similar argument using the Proposition 4 shows the claim holds for flippinf the edge sign from \(+\) to \(-\). Hence, the claim is proven.

Proof of Theorem 4

Proof. (1,2) Given that \( w \notin S \), the fourth case of Proposition 2 implies that the number of vertices which are in \( \varepsilon \)-agreement with \( w \) in \( H^+ \) does not change for this vertex in comparison with \( G^+ \). Therefore, if \( w \) is not a \( \varepsilon \)-light vertex in \( G^+ \), it remains non-\( \varepsilon \)-light in \( H^+ \), too. If \( w \) is a \( \varepsilon \)-light vertex in \( G^+ \), then by the same argument it remains a \( \varepsilon \)-light vertex in \( H^+ \), also.

(3) This is simply implied by Table 1 and Corollary 1.

Proof of Theorem 5

Proof. Considering the discussion just before this theorem for time and space analysis, one can simply call either Algorithm 3 or 6, whether the sign of edge \( f \) is changed from \(-\) to \(+\) or \(+\) to \(-\), respectively.

Proof of Lemma 3

Proof. The discussion just before the Lemma 3 is its proof.

Proof of Lemma 4

Proof. This is a direct consequence of Lemma 3 since all the edges inside \( \widehat{G^+_{t-1}}[C] \) does not belong to \( S \). Therefore, \( \widehat{G^+_{t-1}}[C] = \widehat{G^+_{t}}[C] \) and the claim is shown.

Proof of Theorem 6
Proof. For flipping the sign of an edge \( e = \{u, v\} \), we first compute the connected components of \( \tilde{G}_{t-1} \) where \( S = \{u, v\} \cup N_{G_{t-1}}(u) \cup N_{G_{t-1}}(v) \) and then, decide to merge, split or simply transfer each cluster in \( C_{t-1} \). A split possibility is taken into account whenever there are two vertices \( w, w' \in S \) which are in different connected components. As they are now separated in \( S \), we need to verify their connection by paths out of \( S \). Therefore, we compute the connected components of \( \tilde{G}_{t-1}[C] \) where all edges with both endpoints in \( S \) removed. If there is one connected component, then no split occurs as the vertices \( w \) and \( w' \) are still connected by paths with no intersection with \( S \). However, we need to fall-back to our merging decision if there are any or even copy decision. On the other hand, if there are more than one connected components, we split \( C \) into its connected components. Next, we need to take care of merging these split new clusters with existing ones. To do this, we fall-back again to cluster groups and assign the new split clusters either a copy decision or a merge decision. If take a high-level look, in fact we are reconstructing the connected components of the graph \( G_t^+ \), which is exactly the same as running the Baseline.

For the time complexity, both steps one and two in marking can be accomplished in time \( |C_{t-1}| \). The third step can be accomplished in \( \mathcal{O}(|S| \times |S|) \), too. Moreover, we need to compute the connected components of the clusters with split possibility as well as the subgraph with vertices in \( S \), which accumulation is \( \mathcal{O}(|V_t| + |E_t^+|) \). Therefore, the claimed running time is obtained.

For the space complexity, consider storing cluster groups which are \( \mathcal{O}(|S| \times |S|) \) in the worst case. Each pair is used for at most two clusters, so the claimed space complexity is obtained.

References

[BBC04] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. Machine learning, 56(1):89–113, 2004.

[BGU13] Francesco Bonchi, Aristides Gionis, and Antti Ukkonen. Overlapping correlation clustering. Knowledge and Information Systems, 35(1):1–32, 2013.

[CALM+21] 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.

[CALMP22] Vincent Cohen-Addad, Silvio Lattanzi, Andreas Maggiori, and Nikos Parotsidis. Online and consistent correlation clustering. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, Proceedings of the 39th International Conference on Machine Learning, volume 162 of Proceedings of Machine Learning Research, pages 4157–4179. PMLR, 17–23 Jul 2022.

[CMSY15] 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 symposium on Theory of computing, pages 219–228, 2015.

[CSX12] Yudong Chen, Sujay Sanghavi, and Huan Xu. Clustering sparse graphs. Advances in neural information processing systems, 25, 2012.

[DEFI06] Erik D Demaine, Dotan Emanuel, Amos Fiat, and Nicole Immorlica. Correlation clustering in general weighted graphs. Theoretical Computer Science, 361(2-3):172–187, 2006.

[GG05] Ioannis Giotsis and Venkatesan Guruswami. Correlation clustering with a fixed number of clusters. arXiv preprint cs/0504023, 2005.

[KCMNT08] Dmitri V Kalashnikov, Zhaoli Chen, Sharad Mehrotra, and Rabia Nuray-Turan. Web people search via connection analysis. IEEE Transactions on Knowledge and Data Engineering, 20(11):1550–1565, 2008.
Algorithm 3 \textbf{UpdateG\textsuperscript{+}NegativeToPositive} procedure runs after every call to \textbf{FlipSign\textsubscript{G}(e)} which changed \(e\).\textit{Sign} from \(-\) to \(+\).

1: \textbf{procedure} \textbf{UpdateG\textsuperscript{+}NegativeToPositive}(\(e = \{u, v\}, i\))
2: \hspace{1cm} Let \(G_t\) be the graph after the \(t^{th}\) modification command which was \textbf{FlipSign\textsubscript{G}(e)}.
3: \hspace{1cm} \(G^+ \leftarrow G_t[E^+]\)
4: \hspace{1cm} Let \(A \leftarrow N_{G^+}(u) \cap N_{G^+}(v)\)
5: \hspace{1cm} Let \(B = (N_{G^+}(u) \setminus N_{G^+}(v)) \setminus \{v\}\)
6: \hspace{1cm} Let \(C = (N_{G^+}(v) \setminus N_{G^+}(u)) \setminus \{u\}\)
7: \hspace{1cm} \textbf{for all} \(w \in A\) \hspace{1cm} \(\triangleright \) Applying Lemma \(1\) (1)
8: \hspace{1.2cm} Let \(f = \{u, w\}\) and \(g = \{v, w\}\)
9: \hspace{1.2cm} \textbf{if not} \(f.\text{Agree}\) \textbf{then}
10: \hspace{1.4cm} Call \textbf{VerifyEdge}(\(f = \{u, w\}\))
11: \hspace{1.2cm} \textbf{end if}
12: \hspace{1.2cm} \textbf{if not} \(g.\text{Agree}\) \textbf{then}
13: \hspace{1.4cm} Call \textbf{VerifyEdge}(\(g = \{v, w\}\))
14: \hspace{1.2cm} \textbf{end if}
15: \hspace{1cm} \textbf{end for}
16: \hspace{1cm} \textbf{for} \((x, X)\) in \([(u, B), (v, C)]\) \hspace{1cm} \(\triangleright \) Run once with \(x \leftarrow u\) and \(X \leftarrow B\), another time with \(x \leftarrow v\) and \(X \leftarrow C\)
17: \hspace{2cm} \textbf{for all} \(w \in X\)
18: \hspace{3cm} Let \(f = \{x, w\}\)
19: \hspace{3cm} \textbf{if} \(f.\text{Agree}\) \textbf{then}
20: \hspace{4cm} \textbf{if} \(\text{deg}_{G^+}(x) < \text{deg}_{G^+}(w)\) \textbf{then}\n21: \hspace{5cm} Call \textbf{VerifyEdge}(\(f = \{x, w\}\))\n22: \hspace{4cm} \textbf{else if} \(|N_{G^+}(x) \Delta N_{G^+}(w)| < \text{deg}_{G^+}(x)\) \textbf{then}\n23: \hspace{5cm} Call \textbf{VerifyEdge}(\(f = \{x, w\}\))\n24: \hspace{4cm} \textbf{end if}
25: \hspace{3cm} \textbf{else}\n26: \hspace{4cm} \textbf{if} \(\text{deg}_{G^+}(x) \geq \text{deg}_{G^+}(w)\) \textbf{and} \(|N_{G^+}(x) \Delta N_{G^+}(w)| > \text{deg}_{G^+}(x)\) \textbf{then}\n27: \hspace{5cm} Call \textbf{VerifyEdge}(\(f = \{x, w\}\))\n28: \hspace{5cm} \textbf{end if}\n29: \hspace{3cm} \textbf{end if}
30: \hspace{3cm} \textbf{end if}
31: \hspace{2cm} \textbf{end for}
32: \hspace{2cm} \textbf{end for}
33: \hspace{2cm} \textbf{end for}
34: \hspace{2cm} \textbf{if} \text{NonAgreement}_{G^+}(u, v) \leq \varepsilon \textbf{then}
35: \hspace{3cm} \(f.\text{Agree} \leftarrow \text{True}\)
36: \hspace{3cm} \(u.\text{AgreeCnt}_{G^+} \leftarrow u.\text{AgreeCnt}_{G^+} + 1\)
37: \hspace{3cm} \(v.\text{AgreeCnt}_{G^+} \leftarrow v.\text{AgreeCnt}_{G^+} + 1\)
38: \hspace{2cm} \textbf{else}
39: \hspace{3cm} \(f.\text{Agree} \leftarrow \text{False}\)
40: \hspace{2cm} \textbf{end if}
41: \hspace{2cm} \textbf{end if}
42: \hspace{2cm} \textbf{Let} \(S \leftarrow \{u, v\} \cup u.\text{Neigh}_{G^+} \cup v.\text{Neigh}_{G^+}\)
43: \hspace{2cm} \textbf{for all} \(w \in S\) \hspace{1cm} \(\triangleright \) Fixing the lightness of the vertices in \(G^+\)
44: \hspace{3cm} \(w.\text{IsLight} \leftarrow \left(\frac{w.\text{AgreeCnt}_{G^+}}{\text{deg}_{G^+}(w)} < \varepsilon\right)\)
45: \hspace{2cm} \textbf{end for}
46: \hspace{2cm} \textbf{return} \(G^+\) as \(H^+\).
47: \textbf{end procedure}
Algorithm 4  The $\text{VerifyEdge}(f = \{x, y\})$ maintains $f.\text{Agree}$, $x.\text{AgreeCnt}$ and $y.\text{AgreeCnt}$ whenever an update occurs.

1:  procedure $\text{VerifyEdge}(f = \{x, y\})$
2:    if not $f.\text{Agree}$ and $\text{NonAgreement}_{G^+}(x, y) < \varepsilon$ then
3:      $f.\text{Agree} \leftarrow \text{True}$
4:      $x.\text{AgreeCnt} \leftarrow x.\text{AgreeCnt} + 1$
5:      $y.\text{AgreeCnt} \leftarrow y.\text{AgreeCnt} + 1$
6:    else if $f.\text{Agree}$ and $\text{NonAgreement}_{G^+}(x, y) \geq \varepsilon$ then
7:      $f.\text{Agree} \leftarrow \text{False}$
8:      $x.\text{AgreeCnt} \leftarrow x.\text{AgreeCnt} - 1$
9:      $y.\text{AgreeCnt} \leftarrow y.\text{AgreeCnt} - 1$
10:  end if
11: end procedure

Algorithm 5  $\text{FlipSign}(e)$ flips the sign of an edge $e$ and updates the corresponding clusters

1:  procedure $\text{FlipSign}(e)$
2:    $G_t \leftarrow G_{t-1}$
3:    PreviousSign $\leftarrow e.\text{Sign}$
4:    Change the sign of edge $e$ in $G_t$
5:    Let $G^+ = G_t[E^+]$
6:    if PreviousSign is + then
7:      Call $\text{Update}^+\text{PositiveToNegative}(e)$ \hspace{1cm} $\triangleright$ Updates $G_t$ to reflect sign flip of edge
8:    else
9:      Call $\text{Update}^+\text{NegativeToPositive}(e)$ \hspace{1cm} $\triangleright$ PreviousSign is –
10:  end if
11:  Call $\text{MaintainClusteringAfterFlipSign}$ \hspace{1cm} $\triangleright$ Described on Page 11
12: end procedure
Algorithm 6 Update\(G^+\)PositiveToNegative procedure runs after every call to FlipSign\(_G(e)\) which changed \(e.\text{Sign}\) from + to –.

1: procedure Update\(G^+\)PositiveToNegative\((e = \{u, v\})\)
2: Let \(G_t\) be the graph after the \(t^{th}\) modification command which was FlipSign\(_G(e)\).
3: \(G^+ \gets G_t[E^+]\)
4: Let \(f = \{u, v\}\)
5: if \(f.\text{Agree}\) then \(\triangleright\) After deleting \(f\), \(\varepsilon\)-agreement no more means for \(u\) and \(v\)
6: \(u.\text{AgreeCnt}_{G^+} \gets u.\text{AgreeCnt}_{G^+} - 1\)
7: \(v.\text{AgreeCnt}_{G^+} \gets v.\text{AgreeCnt}_{G^+} - 1\)
8: end if
9: Let \(A \gets N_{G^+}(u) \cap N_{G^+}(v)\)
10: Let \(B = (N_{G^+}(u) \setminus N_{G^+}(v)) \setminus \{v\}\)
11: Let \(C = (N_{G^+}(v) \setminus N_{G^+}(u)) \setminus \{u\}\)
12: for all \(w \in A\) do \(\triangleright\) Applying Lemma\(2\) (1)
13: Let \(f = \{u, w\}\) and \(g = \{v, w\}\)
14: if \(f.\text{Agree}\) then
15: Call VerifyEdge\((f = \{u, w\})\)
16: end if
17: if \(g.\text{Agree}\) then
18: Call VerifyEdge\((g = \{v, w\})\)
19: end if
20: end for
21: for \((x, X)\) in \([u, B], (v, C)\) do \(\triangleright\) Run once with \(x \leftarrow u\) and \(X \leftarrow B\), another time with \(x \leftarrow v\) and \(X \leftarrow C\)
22: for all \(w \in X\) do
23: Let \(f = \{x, w\}\)
24: if \(f.\text{Agree}\) then
25: if \(\text{deg}_{G^+}(x) \leq \text{deg}_{G^+}(w)\) then \(\triangleright\) Applying Lemma\(2\) (2)
26: Call VerifyEdge\((f = \{x, w\})\)
27: else if \(|N_{G^+}(x) \Delta N_{G^+}(w)| > \text{deg}_{G^+}(x)\) then \(\triangleright\) Applying Lemma\(2\) (3-c)
28: Call VerifyEdge\((f = \{x, w\})\)
29: end if
30: else \(\triangleright\) \(f.\text{Agree} = \text{False}\)
31: if \(\text{deg}_{G^+}(x) > \text{deg}_{G^+}(w)\) and \(|N_{G^+}(x) \Delta N_{G^+}(w)| < \text{deg}_{G^+}(x)\) then \(\triangleright\) Applying Lemma\(2\) (3-a)
32: Call VerifyEdge\((f = \{x, w\})\)
33: end if
34: end if
35: end for
36: end for
37: end for
38: Let \(S \gets \{u, v\} \cup u.\text{NEIGH}_{G^+} \cup v.\text{NEIGH}_{G^+}\)
39: for all \(w \in S\) do \(\triangleright\) Fixing the lightness of the vertices in \(G^+\)
40: \(w.\text{IsLight} \gets \left(\frac{w.\text{AgreeCnt}_{G^+}}{\text{deg}_{G^+}(w)} < \varepsilon\right)\)
41: end for
42: return \(G^+\) as \(H^+\).
43: end procedure

19