MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

Fully Dynamic Detection of Constant-Size Subgraphs

verfasst von / submitted by

Qi Cheng Hua, B.Sc.

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2021 / Vienna, 2021

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	A 066 910
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Computational Science
Betreut von / Supervisor:	Univ.-Prof. Dr. Monika Henzinger
Mitbetreut von / Co-Supervisor:	Dr. Kathrin Hanauer

Acknowledgements

I would like to thank Prof. Dr. Monika Henzinger and Dr. Kathrin Hanauer for their generous support throughout the development of this thesis. Their help and regular feedback was invaluable.

Zusammenfassung

Wir präsentieren Algorithmen für das Zählen zusammenhängender Subgraphen mit vier Knoten in dynamischen Graphen. Im Einzelnen kann mithilfe unserer Algorithmen die Anzahl nicht-induzierter Pfade der Länge drei in amortisierter Zeit $\mathcal{O}(\sqrt{m})$, die Anzahl nicht-induzierter Sterne mit vier Knoten in konstanter Zeit und die Anzahl von jedem anderem Subgraphen mit vier Knoten außer der Clique in amortisierter Zeit $\mathcal{O}(m^{\frac{2}{3}})$ aktualisiert werden. Für zusammenhängende induzierte Subgraphen mit vier Knoten zeigen wir, dass der $\mathcal{O}(m)$ Algorithmus von Eppstein et al. [*Theoretical Computer Science*, 447:44–52, 2012] nicht verbessert werden kann, unter Annahme der Vermutung, dass kein Algorithmus existiert, der das Entscheidungsproblem in Zeit $O(n^{4-\varepsilon})$ löst, ob ein statischer Graph eine Clique der Größe vier enthält.

Abstract

We present algorithms for maintaining the number of all connected four-vertex subgraphs in a dynamic graph. Specifically, our algorithms are capable of maintaining the number of non-induced paths of length three in amortized time $\mathcal{O}(\sqrt{m})$, the number of non-induced claws in constant time and the number of any other non-induced subgraph, which is not a clique, in amortized time $\mathcal{O}(m^{\frac{2}{3}})$. For all connected induced subgraphs with four vertices, we prove that the $\mathcal{O}(m)$ algorithm by Eppstein et al. [*Theoretical Computer Science*, 447:44–52, 2012] cannot be improved unless the conjecture that no algorithm exists which can detect 4-cliques in a static graph in time $\mathcal{O}(n^{4-\varepsilon})$ is wrong.

Contents

Chapter 1

Introduction

1.1 Introduction

The number of certain subgraphs in a network is an important metric with applications in ecology, economics and many other fields. In food webs, graphs represent predator-prey relationships, and the occurrence and absence of certain subgraphs give insight into the ecological interplay between species. For example, the subgraph $X \overrightarrow{\rightarrow Y \rightarrow} Z$ is rare in food webs, hinting that a species' predator rarely hunts for that species' prey [MSOI$^+$02]. In economic studies, company-customer relationships can be represented as bipartite graphs, and Zhang et al. [ZSSH14] show by counting the corresponding subgraphs that companies doing business with the same customer will most likely also attract the customers of the respective other company. Since these graphs change over time as companies acquire new customers, there is a need for algorithms capable of handling dynamic graphs as well.

1.2 Related Work

Algorithms counting numbers of subgraphs and solving related problems have been studied extensively for static graphs. Alon, Yuster and Zwick [AYZ94] developed an algorithm to count the number of triangles and other circles (up to seven vertices) in a graph in time $\mathcal{O}(n^{\omega})$, where n denotes the number of vertices and $\omega < 2.373$ [AW21] is the fast matrix multiplication exponent, ie. the smallest value such that two $n \times n$ matrices can be multiplied in $\mathcal{O}(n^{\omega})$ time. Kloks, Kratsch and Müller [KKM95] showed how to compute the number of 4-cliques in time $\mathcal{O}(m^{(\omega+1)/2})$ (m denotes the number of edges) and the number of any other subgraph of size 4 in time $\mathcal{O}(n^{\omega} + m^{(\omega+1)/2})$. A lot of research has also been done to develop parallel algorithms for counting subgraphs (see eg. [KPP$^+$14]) and to develop approximate algorithms in the streaming setting (see eg. [BFL$^+$06], [BYKS02])

A more recent development is counting subgraph numbers for dynamic graphs, ie. graphs in which edges are inserted or deleted from an initial graph. Here, Kara et al. [KNN$^+$19] provided an algorithm for counting triangles in the insertion-only case in amortized time $O(\sqrt{m})$ per insertion, which Dhulipala et al. [DLS20] extended to the fully-dynamic case, ie. where both insertions and deletions are allowed. They also succeeded in using the algorithm for the triangle case to obtain an algorithm maintaining the number of any k-clique for a fixed k in a fully dynamic graph in amortized time $\mathcal{O}(m\alpha^{k-4})$, where $\alpha \in \mathcal{O}(\sqrt{m})$ is the arboricity of the graph. They also give even faster algorithms, which however rely on fast matrix multiplication. The case of triangles has also been studied by Eppstein and Spiro [ES09], who give an amortized time $\mathcal{O}(h)$ algorithm to maintain the number of triangles, where $h \in \mathcal{O}(\sqrt{m})$ is the h-index of a graph, the maximum number such that the graph contains h vertices with degree at least h. Their algorithm can also maintain the number of any other subgraph on three vertices, and Eppstein et al. [EGST12] extended their method to also maintain counts of any four-vertex subgraph in amortized time

Subgraph	our runtime	runtime in Eppstein et al. [EGST12]
Non-induced path	$\mathcal{O}(m^{\frac{1}{2}})$	$\mathcal{O}(h)$
Non-induced claw	$\mathcal{O}(1)$	$\mathcal{O}(1)$
Non-induced paw	$\mathcal{O}(m^{\frac{2}{3}})$	$\mathcal{O}(h^2)$
Non-induced cycle	$\mathcal{O}(m^{\frac{2}{3}})$	$\mathcal{O}(h^2)$
Non-induced diamond	$\mathcal{O}(m^{\frac{2}{3}})$	$\mathcal{O}(h^2)$
Any connected induced four-vertex subgraph	$\mathcal{O}(m)$	$\mathcal{O}(h^2)$

Table 1.1: Update times for counting different subgraphs in our work and the work by Eppstein et al. [EGST12] $h \in \mathcal{O}(\sqrt{m})$

$\mathcal{O}(h^2)$. Dvorak and Tuma [DT13] gave a different algorithm for maintaining counts for arbitrary subgraphs in amortized time $\mathcal{O}(\log(n)^{0.5n'^2-0.5n'-1})$, where n' denotes the number of vertices in the subgraph to be counted, if the set of graphs arising after each update has bounded expansion.

Note that there is a difference between the number of subgraphs and the number of induced subgraphs, where in the latter case the vertices of a subgraph must induce that subgraph. For example, a triangle contains (multiple) paths of length 2, but not as an induced subgraph. However, it is possible to convert non-induced counts to induced counts and vice versa, and the algorithms by Eppstein et al. maintain counts of induced and non-induced counts in time $\mathcal{O}(h^2)$.

1.3 Our contribution

Our work focuses on algorithms maintaining counts of connected four-vertex-subgraphs and is built upon the work by Eppstein et al. [EGST12]. In the case of graphs with large h-indices, we further reduce the update time required to maintain the number of any non-induced connected four-vertex subgraph which is not the 4-clique. Specifically, we can maintain the number of paws, cycles of length 4 and diamonds in time $\mathcal{O}(m^{\frac{2}{3}})$. (see table 1.1) We also prove that the algorithms for maintaining the number of connected induced four-vertex subgraphs by Eppstein et al. are asymptotically optimal, provided that the number of 4-cliques in a static graph cannot be counted faster than in time $\mathcal{O}(n^4)$, where n is the number of vertices in that graph.

1.4 Structure of this thesis

This thesis is structured the following way: In chapter 2 we will introduce our notation, state the problem and sketch our approach. In chapter 3 we will introduce our data structure, which allows us to speed up counting in the non-induced case. In chapter 4 we will show how we use this data structure to count non-induced subgraphs of size 4. In chapter 5 we will show how we can use this to count induced subgraphs of size 4 and prove conditional lower bounds for this case.

Chapter 2

Preliminaries

Definition 1. (Graph) Let G be a graph. We write V_G for the set of its nodes and E_G for the set of its edges. For a set of nodes V_G and a set of edges E_G we write $G = (V_G, E_G)$ for the graph with these nodes and edges.

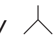
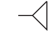
Definition 2. (Dynamic Graph) A dynamic graph is an initial graph G with a sequence of edges (updates) $(e_1, ..., e_k)$. At the i-th edge e_i, if e_i has occured an odd number of times in the sequence before i and e_i is contained in the original graph or if e_i has occured an even number of times in the sequence before i and e_i is not contained in the original graph, e_i is inserted. Conversely, if e_i has occured an odd number of times in the sequence before i and e_i is not contained in the original graph or if e_i has occured an even number of times in the sequence before i and e_i is contained in the original graph, e_i is deleted.

Basically, when iteratively processsing the sequence of updates, e_i is inserted if it does not already exist at that time and deleted if it does.

Definition 3. (Subgraph) Let G be a graph and $S \subset V_G$. We write $G[S]$ for the subgraph of G induced by S.

Definition 4. (Neighbourhood) Let G be a graph and $u \in V_G$ a vertex. We write $\mathcal{N}(u) = \{v \in V_G : \{u, v\} \in E_G\}$ for the set of vertices adjacent to u, ie. its neighbourhood.

Definition 5. (Degree of a Vertex) Let G be a graph and $u \in V_G$ a vertex. We write $\delta(u) = |\mathcal{N}(u)|$ for the degree of u. When considering an update $\{u, v\}$, we write $\delta'(u) = |\mathcal{N}(u) - \{v\}|, \delta'(v) = |\mathcal{N}(v) - \{u\}|$ and $\delta'(w) = \delta(w)$ for all $w \in V_G - \{u, v\}$.

Proposition 1. *There are six connected graphs on four nodes: The path* *, the claw*  *, the paw*  *, the cycle* *, the diamond* *and the clique* *.*

Definition 6. (Subgraph Isomorphism) Let G, H be graphs. We call $\phi : V_H \rightarrow V_G$ a subgraph isomorphism from H to G, if for each edge $\{u, v\} \in E_H$ we have $\{\phi(u), \phi(v)\} \in E_G$. We call $\phi : V_H \rightarrow V_G$ an induced subgraph isomorphism from H to G, if ϕ is a subgraph isomorphism from H to G and for any pair of non-adjacent vertices $u, v \in V_H, \{u, v\} \notin E_H$, we have $\{\phi(u), \phi(v)\} \notin E_G$.

Note that for a non-induced subgraph isomorphism we might have $\{\phi(u), \phi(v)\} \in E_G$ despite $\{u, v\} \notin E_H$.

Definition 7. (Subgraph Isomorphism Count) Let G, H be graphs. We denote by $c(H, G)$ the number of subgraph isomorphisms from H to G divided by the number of subgraph automorphisms from H to H. We also denote by $c_I(H, G)$ the number of induced subgraph isomorphisms from H to G divided by the number of subgraph automorphisms from H to H.

The division helps us to get rid of double counting symmetries of H. We will generally write G for the base graph and H for the subgraph of which the count we are interested in.

We are now ready to state the problem we study in this thesis:

Problem 1. For a dynamic graph G, what are upper and lower bounds for the runtime of a fully dynamic algorithm computing $c(H,G)$ or $c_I(H,G)$ for a connected graph H on four nodes in terms of the number of edges $|E_G|$ in G?

In this thesis we will provide upper bounds by giving algorithms computing $c(H,G)$ or $c_I(H,G)$ within that upper bound. We will also show conditional lower bounds for computing $c_I(H,G)$ by reducing the problem of computing $c_I(H,G)$ to the problem of detecting a 4-clique in G, which is conjectured to not be solvable in time $\mathcal{O}(|V_G|^{4-\varepsilon})$ for any $\varepsilon > 0$ (see eg. [BHG+21],[LWW18],[ABW15]).

A common technique used in many subgraph counting algorithms (eg. [AYZ94], [EGST12], [DLS20], [KNN+19]) is partitioning the vertices of G into low and high degree vertices:

Definition 8. (Low and High Degree Vertices) Let G be a graph, $K \in \mathbb{R}$ a threshold. We call a partition of the vertices of G a partition into low and high degree vertices if for all low degree vertices $v \in V_G$ we have $\delta(v) \leq K$, the number of high degree vertices $v \in V_G$ is bounded by $12 \cdot \frac{|E_G|}{K}$ and for each vertex we can check in constant time to which partition it belongs. We write $\mathcal{L}$ for the set of low degree vertices and $\mathcal{H}$ for the set of high degree vertices, ie.

$$V_G = \mathcal{L} \cup \mathcal{H}, \mathcal{L} \cap \mathcal{H} = \emptyset, \delta(v) \leq K \ \forall v \in \mathcal{L}, |\mathcal{H}| \leq 12\frac{|E_G|}{K}.$$

For a vertex $w \in V_G$, we write $\mathcal{N}_{\mathcal{L}}(w) = \mathcal{N}(w) \cap \mathcal{L}$ for the set of neighbours of w with low degree and $\mathcal{N}_{\mathcal{H}}(w) = \mathcal{N}(w) \cap \mathcal{H}$ for the set of neighbours of w with high degree.

We will show in chapter 3 how to get such a partition for a dynamic graph. An immediate consequence is:

Proposition 2. *We have $|\mathcal{N}_{\mathcal{L}}(w)|, |\mathcal{N}_{\mathcal{H}}(w)|, |\mathcal{N}(w)| \leq K$ if w has low degree and $|\mathcal{N}_{\mathcal{H}}(w)| \leq 12\frac{|E_G|}{K}$ for any vertex $w \in V_G$.*

The algorithms of Eppstein et al. [ES09], [EGST12] for maintaining $c(H,G)$ given an updated edge $\{u,v\}$, on which our algorithms are based on, work the following way: If H is isomorphic to a subgraph of G and this subgraph contains $\{u,v\}$, then one edge $e \in E_H$ is mapped to $\{u,v\}$ under this isomorphism. Therefore we get the total number of subgraphs in G isomorphic to H containing $\{u,v\}$ if we count the number of subgraphs of G isomorphic to H, where e is mapped to $\{u,v\}$ for each $e \in E_H$ separately and add these counts. Therefore we have to maintain counts for subgraphs $H - e$ where the nodes incident to e are mapped to u and v. Upon insertion or deletion of $\{u,v\}$, this count has to be added to or subtracted from $c(H,G)$. This procedure can be continued recursively: Instead of counting $H-e$, we can count $H-e-e'$ which allows us to maintain our count for $H-e$ and so on. The cost of maintaining the count of these subgraphs $H-e$ depends on which of its nodes are high or low degree vertices in G, so we have different counts depending on which nodes in H get mapped to low degree vertices and which nodes get mapped to high degree vertices. This motivates the following definition:

Definition 9. (Auxiliary Subgraph Isomorphism Count) Let G be a dynamic graph partitioned into low and high degree vertices as in definition 8, let H be the subgraph we would like to count. Let $I_{\mathcal{L}}, I_{\mathcal{H}}$ be disjoint subsets of V_H and let $I_G = \{v_1, ..., v_{|I_G|}\}$ be another subset of V_H. (For $I_{\mathcal{L}}, I_{\mathcal{H}}$ and I_G, $\mathcal{L}, \mathcal{H}$ and G are just labels and do not refer to the sets of low/high degree vertices or the graph G. However their image under the subgraph isomorphism is related to $\mathcal{L}, \mathcal{H}$ and G which is why we chose this notation.) Given any set of vertices $\{w_1, ..., w_{|I_G|}\} \subset V_G$, we denote the number of subgraph isomorphisms $\phi : V_H \to V_G$ from H to G with $\phi(v) \in \mathcal{L} \ \forall v \in I_{\mathcal{L}}, \phi(v) \in \mathcal{H} \ \forall v \in I_{\mathcal{H}}$ and $\phi(I_G) = \{w_1, ..., w_{|I_G|}\}$, divided by the number of subgraph automorphisms from H to itself by $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{w_1, ..., w_{|I_G|}\})$. (ie. I_G are the vertices in V_H which map to $\{w_1, ..., w_{|I_G|}\}$, $I_{\mathcal{H}}$ are vertices in V_H which map to high degree vertices and $I_{\mathcal{L}}$ are the vertices in V_H which map to low degree vertices.)

In the example of maintaining $H - e$ above, we would set $I_G = \{u_H, v_H\}$ for the nodes u_H, v_H incident to e, ie. $e = \{u_H, v_H\}$. We would then maintain $c(H - e, G, I_G, \emptyset, \emptyset, \{x, y\})$ for any pair of vertices $x, y \in V_G$, and upon insertion/removal of an edge $\{u, v\}$ in G, add/subtract $c(H - e, G, I_G, \emptyset, \emptyset, \{u, v\})$ from $c(H, G)$. As another example, notice $c(H, G) = c(H, G, \emptyset, \emptyset, \emptyset, \emptyset)$. Note that in this example or also in general there might be vertices in $V_H - (I_G \cup I_{\mathcal{H}} \cup I_{\mathcal{L}})$, which are mapped under ϕ to some vertices in $V_G - \{w_1, ..., w_{|I_G|}\}$ and we do not care whether these vertices have high or low degree in G.

The auxiliary counts used by Eppstein et al. [EGST12] have in common that $I_{\mathcal{H}} = \emptyset, I_G \cup I_{\mathcal{L}} = V_H, I_G \cap I_{\mathcal{L}} = \emptyset$. Not having these restrictions allows us to taylor the subgraph maintained more diligently and get better runtime estimates, because we have to iterate through $\mathcal{H}$ less often. Having $I_{\mathcal{H}} \neq \emptyset$ allows us to keep track of subgraphs containing high degree vertices without having to go through $\mathcal{H}$ to find them. Same goes for having vertices in $V_H - (I_G \cup I_{\mathcal{L}})$. The algorithms by Eppstein et al. [EGST12] for counting paws, cycles and diamonds have two nested loops iterating through neighbours of low degree vertices and two nested loops iterating through the set of high degree vertices, yielding a runtime of $\mathcal{O}(\max(K^2, \frac{|E_G|^2}{K^2})$, which is minimized for $K = \sqrt{|E_G|}$, ie. a runtime of $\mathcal{O}(|E_G|)$. We will show in chapter 4 how to skip one loop through the set of high degree vertices, yielding a runtime of $\mathcal{O}(\max(K^2, \frac{|E_G|}{K})$, which is minimized for $K = |E_G|^{\frac{1}{3}}$, ie. a runtime of $\mathcal{O}(|E_G|^{\frac{2}{3}})$.

The algorithms by Eppstein et al. [EGST12] for computing $c_I(H, G)$ use different techniques and just require black-box algorithms for computing $c(H, G)$. We will describe them in chapter 5.

Chapter 3

Auxiliary counts

As described in chapter 2 and shown by Eppstein et al. [EGST12], $c(H, G)$ can be maintained with the help of auxiliary counts $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{w_1, ..., w_{|I_G|}\})$. This chapter is solely dedicated to discuss these auxiliary counts. We start off with a theorem showing how to maintain a partition of the vertices of G into low and high degree vertices as in definition 8 with threshold $K = |E_G|^\alpha$ for some $\alpha \in (0, 1)$ and how this partition allows us to maintain $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{w_1, ..., w_{|I_G|}\})$ in time $\mathcal{O}(|E_G|^\beta)$ for some $\beta \in (0, \infty)$ if we can show the following two claims:

1. $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{w_1, ..., w_{|I_G|}\})$ can be maintained in time $\mathcal{O}(|E_G|^{\beta+\alpha})$ when a vertex changes from being a low degree vertex to being a high degree vertex or vice versa. ($K = |E_G|^\alpha$ is the threshold in the partition into low and high degree vertices.)

2. $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{w_1, ..., w_{|I_G|}\})$ can be maintained in time $\mathcal{O}(|E_G|^\beta)$ when an edge is updated where no vertex changes partition.

Different variations of this theorem are used in a many existing subgraph counting algorithms for dynamic graphs, such as in [ES09], [EGST12], [DLS20], [KNN+19].

The remainders of this chapter will then be dedicated to proving these two things for various auxiliary counts.

3.1 General Framework

Theorem 1. *Let G be a dynamic graph and $\alpha \in (0, 1), \beta \in (0, \infty)$. Set $K = |E_G|^\alpha$. Then there exists a partition of the vertices of G in low and high degree nodes as in definition 8 with threshold K, such that for any sequence of $\frac{|E_G|}{2}$ updates, the only vertices changing from being a low to a high degree vertex or vice versa in an update are the vertices incident to the updated edge, and any vertex changing from being a low to a high degree vertex or vice versa which will not change again for the next $\Omega(K)$ updates during the same sequence of $\frac{|E_G|}{2}$ updates. After $\frac{|E_G|}{2}$ updates, all nodes might change from being a low to a high degree vertex or vice versa. This partition can be calculated initially in $\mathcal{O}(|E_G|)$ time and updated in amortized constant time.*

Moreover, given a count $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{w_1, ..., w_{|I_G|}\})$ as in definition 9 on this partition for all $w_1, ..., w_{|I_G|} \in V_G, w_i \neq w_j$, ie. for all $|I_G|$-tuples of pairwise distinct vertices in G, if we have an algorithm which updates $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{w_1, ..., w_{|I_G|}\})$ for all $w_1, ..., w_{|I_G|} \in V_G, w_i \neq w_j$ when a single vertex w with $\delta'(w) \leq K$ changes from being a low to a high degree vertex or vice versa in time $|E_G|^{\alpha+\beta}$ and if we have an algorithm which updates $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{w_1, ..., w_{|I_G|}\})$ for all $w_1, ..., w_{|I_G|} \in V_G, w_i \neq w_j$ for an updated edge in time $|E_G|^\beta$ under the assumption that no vertex changes from being a low to a high degree vertex or vice versa, then we can maintain $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{w_1, ..., w_{|I_G|}\})$ in amortized time $\mathcal{O}(|E_G|^\beta)$.

Proof. Step 0 (General setup):

Because $K = |E_G|^\alpha$ changes during updates of G, we use a fixed threshold K_{fix} for the next $\frac{|E_G|}{2}$ updates, which we set to $K_{fix} = \frac{1}{4}K = \frac{1}{4}|E_G|^\alpha$.

Step 1 (Initial partition):

We initially partition V_G into low and high degree nodes as follows: For any vertex v with $\delta(v) \leq K_{fix} = \frac{1}{4}K = \frac{1}{4}|E_G|^\alpha$ we insert v in $\mathcal{L}$ and mark v as a low degree vertex. For any vertex v with $\delta(v) \geq 2K_{fix} = \frac{1}{2}K = \frac{1}{2}|E_G|^\alpha$ we insert v in $\mathcal{H}$ and mark v as a high degree vertex. Any other vertex is arbitrarily assigned to either $\mathcal{L}$ or $\mathcal{H}$ and marked accordingly. Whenever we move or insert a vertex in $\mathcal{L}$ or $\mathcal{H}$ from now on, we also implicitly mark v accordingly, so we can determine which partition a vertex belongs in in constant time. Initially, we thus have $\delta(v) < 2K_{fix} = \frac{1}{2}K \ \forall v \in \mathcal{L}$ and $\delta(v) > K_{fix} = \frac{1}{4}K \ \forall v \in \mathcal{H}$. We also have

$$2|E_G| = \sum_{v \in V_G} \delta(v) \geq \sum_{v \in \mathcal{H}} \delta(v) \geq \frac{1}{4}|\mathcal{H}|K,$$

thus $|\mathcal{H}| \leq \frac{8|E_G|}{K}$ as required in definition 8. This initial partitioning takes $\mathcal{O}(|E_G|)$ time.

Step 2 (Changes to partition during updates):

We fix K_{fix} for the next $\frac{|E_G|}{2}$ updates. In this period, we change the partition only if an edge $\{u, v\}$ is inserted with $u \in \mathcal{L}$ and $\delta'(u) = \lfloor 2K_{fix} \rfloor$ (then we move u from $\mathcal{L}$ to $\mathcal{H}$) or $v \in \mathcal{L}$ and $\delta'(v) = \lfloor 2K_{fix} \rfloor$ (then we move v from $\mathcal{L}$ to $\mathcal{H}$), or if an edge $\{u, v\}$ is removed with $u \in \mathcal{H}$ and $\delta'(u) = \lceil K_{fix} \rceil$ (then we move u from $\mathcal{H}$ to $\mathcal{L}$) or $v \in \mathcal{H}$ and $\delta'(v) = \lceil K_{fix} \rceil$ (then we move v from $\mathcal{H}$ to $\mathcal{L}$). After $\frac{|E_G|}{2}$ updates, we reset all vertices and partition them again as we did initially. During the $\frac{|E_G|}{2}$ updates, each update requires constant time as only a constant number of vertices (the ones incident to the updated edge) are affected. The initial partitioning requires $\mathcal{O}(|E_G|)$ time, which can be amortized to constant time, as it only takes place every $\frac{|E_G|}{2}$ updates.

Step 3 (The number of edges can at most change by 50% during the sequence of $\frac{|E_G|}{2}$ updates):

Let G_{old} be the initial graph or the graph after such a reset and G_{new} be any graph arising from G_{old} after $\frac{|E_{G_{old}}|}{2}$ or less updates. Then at most $\frac{|E_{G_{old}}|}{2}$ edges were inserted or removed, thus we have

$$\frac{|E_{G_{old}}|}{2} \leq |E_{G_{new}}| \leq \frac{3|E_{G_{old}}|}{2} \tag{3.1}$$

Step 4 (Proof that a vertex changes partitions at most every $\Omega(K)$ updates):

Obviously, a vertex changing from being a low degree vertex to being a high degree vertex or vice versa does not change again for the next $\lfloor 2K_{fix} \rfloor - \lceil K_{fix} \rceil \geq \lfloor K_{fix} - 2 \rfloor$ updates when no repartitioning occurs. We also have

$$K_{fix} = \frac{1}{4}|E_{G_{old}}|^\alpha \geq \frac{1}{4}\left(\frac{2}{3}|E_{G_{new}}|\right)^\alpha \geq \frac{1}{6}|E_{G_{new}}|^\alpha = \frac{1}{6}K \in \Omega(K)$$

using equation 3.1. Therefore any vertex changing from being a low to a high degree vertex or vice versa which will not change again for the next $\Omega(K)$ updates during the same sequence of $\frac{|E_G|}{2}$ updates.

Step 5 (Proof that the guarantees in definition 8 hold):

Let $v \in \mathcal{L}$ be a low degree vertex. This implies $\delta(v) \leq 2K_{fix} = \frac{1}{2}|E_{G_{old}}|^\alpha$, otherwise v would have been assigned to $\mathcal{H}$. We calculate

$$\delta(v) \leq \frac{1}{2}|E_{G_{old}}|^\alpha \leq \frac{1}{2} \cdot 2^\alpha \cdot |E_{G_{new}}|^\alpha \leq |E_{G_{new}}|^\alpha \leq K.$$

Now let $v \in \mathcal{H}$ be a high degree vertex. This implies $\delta(v) \geq K_{fix} = \frac{1}{4}|E_{G_{old}}|^\alpha$, otherwise v would have been assigned to $\mathcal{L}$. We calculate

$$\delta(v) \geq \frac{1}{4}|E_{G_{old}}|^\alpha \geq \frac{1}{4} \cdot \left(\frac{2}{3}|E_{G_{new}}|\right)^\alpha \geq \frac{1}{6}|E_{G_{new}}|^\alpha = \frac{1}{6}K,$$

which yields

$$2\left|E_{G_{new}}\right| = \sum_{v\in V_{G_{new}}} \delta(v) \geq \sum_{v\in\mathcal{H}} \delta(v) \geq \frac{1}{6}\left|\mathcal{H}\right| K,$$

and therefore $|\mathcal{H}| \leq \frac{12|E_G|}{K}$ as required.

Step 6 (Proof that we can maintain $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{w_1, ..., w_{|I_G|}\})$):

Assume now we have algorithms maintaining $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{w_1, ..., w_{|I_G|}\})$ for all $w_1, ..., w_{|I_G|} \in V_G, w_i \neq w_j$ when a single vertex changes partition or an update occurs without vertices changing partition. Initially, after partitioning the vertices in low and high degree vertices, we perform $|E_G|$ insertions in the graph $(V_G, \emptyset)$. This requires $\sum_{m=1}^{|E_G|} \mathcal{O}(m^\beta) = \mathcal{O}(|E_G|^{\beta+1})$ time. As this happens every $\frac{|E_G|}{2}$ updates, we can amortize this cost to $\mathcal{O}(|E_G|^\beta)$ per update. At each update $\{u, v\}$ we check the degrees of u and v as described before and if necessary, move them from $\mathcal{L}$ to $\mathcal{H}$ or vice versa. If $w \in \{u, v\}$ is moved, we either have $w \in \mathcal{L}$ before the update or $w \in \mathcal{L}$ after the update. In both cases we have $\delta'(w) \leq K$. If u or v change partition, this takes $\mathcal{O}(|E_G|^{\alpha+\beta})$ time, otherwise we just check the degrees of u and v which takes constant time. As a vertex changes partition at most once every $\Omega(K)$ updates, the time can be amortized to $\mathcal{O}(|E_G|^\beta)$ per update. Doing the update itself also requires $\mathcal{O}(|E_G|^\beta)$ time, so we obtain an overall amortized time of $\mathcal{O}(|E_G|^\beta)$ per update. □

3.2 Auxiliary counts

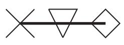
We are now ready to show how to maintain various auxiliary counts. For each auxiliary count, we will show how to maintain $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{w_1, ..., w_{|I_G|}\})$ when a vertex changes partition and when an edge update occurs without changing the partition. Theorem 1 then implies that $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{w_1, ..., w_{|I_G|}\})$ can be maintained.

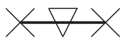
Given a vertex changing the partition, we will have to prove that we can count all subgraphs of G isomorphic to H, in which any node in $I_{\mathcal{H}} \cup I_{\mathcal{L}}$ is mapped to the vertex changing partition under the subgraph isomorphism and we will treat each vertex in $I_{\mathcal{H}} \cup I_{\mathcal{L}}$ separately. Given an edge update $\{u, v\}$ we will have to count all subgraphs of G isomorphic to H, in which any edge in H is mapped to $\{u, v\}$ under the subgraph isomorphism and we will treat each edge in H separately. Given $I_{\mathcal{H}}$ and $I_{\mathcal{L}}$ and which partition u and v are assigned to, many edges in H will not be mapped to $\{u, v\}$, because the vertices incident to the edge in H are not mapped to the same partition. Eg. if $I_{\mathcal{H}} = V_H$, then any updated edge $\{u, v\}$ with $u \in \mathcal{L}$ or $v \in \mathcal{L}$ will not change $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{w_1, ..., w_{|I_G|}\})$.

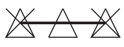
Figure 3.1 illustrates the subgraphs which we are going to count in this chapter. The path in definition 11 (see figure 3.1b, $s2$ in the work by Eppstein et al.), 13 (see figure 3.1d, $s3$ in the work by Eppstein et al.), the claw in definition 15 (see figure 3.1f, $s7$ in the work by Eppstein et al.) and the paw in definition 17 (see figure 3.1h, $s5$ in the work by Eppstein et al.) are also used in the work by Eppstein et al. [EGST12]. We will repeat their proofs for the maintainance of these auxiliary subgraph counts. Some other subgraphs differ (slightly) from the subgraphs in the work of Eppstein et al. The path in definition 10 (see figure 3.1a) corresponds to $s0$, the claw in definition 14 (see figure 3.1e) corresponds to $s4$ and the triangle in definition 16 (see figure 3.1g) corresponds to $s1$ in the work of Eppstein et al. In these subgraphs, we replace some low-degree vertices with arbitrary vertices, and show that we can maintain their counts with the same methods. Finally, the path in definition 12 (see figure 3.1c) has no counterpart in the work by Eppstein et al., and together with our modifications in the other auxiliary subgraphs will allow us to skip looping through the high degree vertices later on when counting subgraphs of size four.

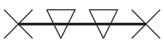
3.2.1 Auxiliary paths

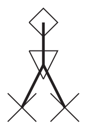
Definition 10. Let G be a dynamic graph partitioned into low and high degree vertices as in theorem 1 with threshold K. We define H to be a path of length 2, ie. $V_H = \{v_1, v_2, v_3\}$ and

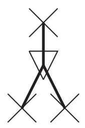

(a) Path in definition 10.

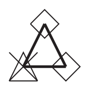

(b) Path in definition 11.


(c) Path in definition 12.


(d) Path in definition 13.


(e) Claw in definition 14.


(f) Claw in definition 15.


(g) Triangle in definition 16.

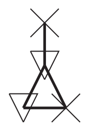

(h) Paw in definition 17.

Figure 3.1: The subgraphs H in the discussed auxiliary counts. ▽ denotes a vertex in $I_{\mathcal{L}}$, △ denotes a vertex in $I_{\mathcal{H}}$, × denotes a vertex in I_G, ⊠ denotes a vertex in $I_{\mathcal{H}} \cap I_G$, and ◇ denotes a vertex in $V_H - (I_{\mathcal{L}} \cup I_{\mathcal{H}} \cup I_G)$. Edges — are drawn bold.

$E_H = \{\{v_1, v_2\}, \{v_2, v_3\}\}$. We also define $I_G = \{v_1\}, I_{\mathcal{L}} = \{v_2\}, I_{\mathcal{H}} = \emptyset$, ie. the center node has low degree and we fix one endpoint (see figure 3.1a).

Lemma 1. *Let $H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}$ be as in definition 10. Then given a vertex w which changes from being a low degree vertex to a high degree vertex or vice versa with $\delta'(w) \leq K$ we can maintain $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{x\})$ in time $\mathcal{O}(K)$ for all $x \in V_G$.*

Proof. First assume w was a low degree vertex and turns into a high degree vertex. Then for each neighbour x of w we decrement $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{x\})$ by $\delta'(w) - 1$.

Now assume w was a high degree vertex and turns into a low degree vertex. Then for each neighbour x of w we increment $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{x\})$ by $\delta'(w) - 1$.

Because of $\delta'(w) \leq K$, the number of neighbours of w is in $\mathcal{O}(K)$. □

Lemma 2. *Let $H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}$ be as in definition 10. Then given an update $\{u, v\}$ where no vertex changes partition we can maintain $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{x\})$ in time $\mathcal{O}(K)$ for all $x \in V_G$.*

Proof. We distinguish four cases:

1. $u, v \in \mathcal{L}$: For each neighbour $w \neq v$ of u, we increment/decrement $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{w\})$ by 1. For each neighbour $w \neq u$ of v, we increment/decrement $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{w\})$ by 1. As u and v are low-degree vertices, they have $\mathcal{O}(K)$ neighbours. We also increment/decrement $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{u\})$ by $\delta'(v)$ and $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{v\})$ by $\delta'(u)$.

2. $u \in \mathcal{L}, v \in \mathcal{H}$: For each neighbour $w \neq v$ of u, we increment/decrement $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{w\})$ by 1. We increment/decrement $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{v\})$ by $\delta'(u)$. As u is a low-degree vertex, it has $\mathcal{O}(K)$ neighbours.

3. $u \in \mathcal{H}, v \in \mathcal{L}$: Analogous to the previous case with swapped roles for u and v.

4. $u, v \in \mathcal{H}$: Nothing needs to be done here.

□

Corollary 1. *Let $H, G, I_G, I_\mathcal{L}, I_\mathcal{H}$ be as in definition 10. Then we can maintain $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{x\})$ in amortized time $\mathcal{O}(K)$ for all $x \in V_G$.*

Proof. Follows from lemma 1, 2 and theorem 1 □

Definition 11. Let G be a dynamic graph partitioned into low and high degree vertices as in theorem 1 with threshold K. We define H to be a path of length 2, ie. $V_H = \{v_1, v_2, v_3\}$ and $E_H = \{\{v_1, v_2\}, \{v_2, v_3\}\}$. We also define $I_G = \{v_1, v_3\}, I_\mathcal{L} = \{v_2\}, I_\mathcal{H} = \emptyset$, ie. the center node has low degree and we fix both endpoints (see figure 3.1b).

Lemma 3. *Let $H, G, I_G, I_\mathcal{L}, I_\mathcal{H}$ be as in definition 11. Then given a vertex w which changes from being a low degree vertex to a high degree vertex or vice versa with $\delta'(w) \leq K$ we can maintain $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{x, y\})$ in time $\mathcal{O}(K^2)$ for all pairs of distinct vertices $x, y \in V_G$.*

Proof. First assume w was a low degree vertex and turns into a high degree vertex. For each unordered pair of distinct neighbours x, y we decrement $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{x, y\})$ by 1.

Now assume w was a high degree vertex and turns into a low degree vertex. For each unordered pair of distinct neighbours x, y we increment $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{x, y\})$ by 1.

Because of $\delta'(w) \leq K$, the number of pairs of neighbours of w is in $\mathcal{O}(K^2)$. □

Lemma 4. *Let $H, G, I_G, I_\mathcal{L}, I_\mathcal{H}$ be as in definition 11. Then given an update $\{u, v\}$ where no vertex changes partition we can maintain $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{x, y\})$ in time $\mathcal{O}(K)$ for all pairs of distinct vertices $x, y \in V_G$.*

Proof. We distinguish four cases:

1. $u, v \in \mathcal{L}$: For each neighbour $w \neq v$ of u, we increment/decrement $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{w, v\})$ by 1. For each neighbour $w \neq u$ of v, we increment/decrement $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{w, u\})$ by 1. As u and v are low-degree vertices, they have $\mathcal{O}(K)$ neighbours.

2. $u \in \mathcal{L}, v \in \mathcal{H}$: For each neighbour $w \neq v$ of u, we increment/decrement $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{w, v\})$ by 1. As u is a low-degree vertex, it has $\mathcal{O}(K)$ neighbours.

3. $u \in \mathcal{H}, v \in \mathcal{L}$: Analogous to the previous case with swapped roles for u and v.

4. $u, v \in \mathcal{H}$: Nothing needs to be done here.

□

Corollary 2. *Let $H, G, I_G, I_\mathcal{L}, I_\mathcal{H}$ be as in definition 11. Then we can maintain $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{x, y\})$ in time $\mathcal{O}(K)$ for all pairs of distinct vertices $x, y \in V_G$.*

Proof. Follows from lemma 3, 4 and theorem 1 □

Definition 12. Let G be a dynamic graph partitioned into low and high degree vertices as in theorem 1 with threshold K. We define H to be a path of length 2, ie. $V_H = \{v_1, v_2, v_3\}$ and $E_H = \{\{v_1, v_2\}, \{v_2, v_3\}\}$. We also define $I_G = \{v_1, v_3\}, I_\mathcal{L} = \emptyset, I_\mathcal{H} = \{v_1, v_2, v_3\}$, ie. all nodes have high degree and we fix both endpoints (see figure 3.1c).

Lemma 5. *Let $H, G, I_G, I_\mathcal{L}, I_\mathcal{H}$ be as in definition 12. Then given a vertex w which changes from being a low degree vertex to a high degree vertex or vice versa with $\delta'(w) \leq K$ we can maintain $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{x, y\})$ in time $\mathcal{O}(\max(K^2, |E_G|))$ for all pairs of distinct vertices $x, y \in V_G$.*

Proof. First assume w was a low degree vertex and turns into a high degree vertex. For each unordered pair of distinct high degree neighbours x, y of w we increment $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{x, y\})$ by 1. Because of $\delta'(w) \leq K$, the number of pairs of high degree neighbours of w is in $\mathcal{O}(K^2)$. For every high degree neighbour $x \in \mathcal{N}(w) \cap \mathcal{H}$ and every high degree vertex $y \in \mathcal{H}$ which is adjacent to x we increment $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{w, y\})$ by 1. As there are $\mathcal{O}(\frac{|E_G|}{K})$ high degree vertices and w has $\mathcal{O}(K)$ neighbours, this requires at most $\mathcal{O}(|E_G|)$ time.

Now assume w was a high degree vertex and turns into a low degree vertex. For each unordered pair of distinct high degree neighbours x, y of w we decrement $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{x, y\})$ by 1. Because of $\delta'(w) \leq K$, the number of pairs of high degree neighbours of w is in $\mathcal{O}(K^2)$. For every high degree vertex x we set $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{x, w\}) = 0$. As there are $\mathcal{O}(\frac{|E_G|}{K})$ high degree vertices, this requires at most $\mathcal{O}(\frac{|E_G|}{K})$ time. □

Lemma 6. *Let $H, G, I_G, I_\mathcal{L}, I_\mathcal{H}$ be as in definition 12. Then given an update $\{u, v\}$ where no vertex changes partition we can maintain $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{x, y\})$ in time $\mathcal{O}(\frac{|E_G|}{K})$ for all pairs of distinct vertices $x, y \in V_G$.*

Proof. Assume u and v have high degree, otherwise nothing needs to be done. For each high degree vertex $w \neq v$ adjacent to u, we increment/decrement $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{v, w\})$ by 1. For each high degree vertex $w \neq u$ adjacent to v, we increment/decrement $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{u, w\})$ by 1. By the definition of high degree vertices, there are at most $12\frac{|E_G|}{K}$ high degree vertices. □

Corollary 3. *Let $H, G, I_G, I_\mathcal{L}, I_\mathcal{H}$ be as in definition 12. Then we can maintain $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{x, y\})$ in time $\mathcal{O}(\max(\frac{|E_G|}{K}, K))$ for all pairs of distinct vertices $x, y \in V_G$.*

Proof. Follows from lemma 5, 6 and theorem 1 □

Definition 13. Let G be a dynamic graph partitioned into low and high degree vertices as in theorem 1 with threshold K. We define H to be a path of length 3, ie. $V_H = \{v_1, v_2, v_3, v_4\}$ and $E_H = \{\{v_1, v_2\}, \{v_2, v_3\}, \{v_3, v_4\}\}$. We also define $I_G = \{v_1, v_4\}, I_\mathcal{L} = \{v_2, v_3\}, I_\mathcal{H} = \emptyset$, ie. both internal nodes have low degree and we fix both endpoints (see figure 3.1d).

Lemma 7. *Let $H, G, I_G, I_\mathcal{L}, I_\mathcal{H}$ be as in definition 13. Then given a vertex w which changes from being a low degree vertex to a high degree vertex or vice versa with $\delta'(w) \leq K$ we can maintain $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{x, y\})$ in time $\mathcal{O}(K^3)$ for all pairs of distinct vertices $x, y \in V_G$.*

Proof. First assume w was a low degree vertex and turns into a high degree vertex. Then w has $k_\mathcal{L} \leq K + 1$ low degree neighbours $w_\mathcal{L}^1, ..., w_\mathcal{L}^{k_\mathcal{L}}$ and $k_\mathcal{H} \leq K + 1$ high degree neighbours $w_\mathcal{H}^1, ..., w_\mathcal{H}^{k_\mathcal{H}}$. For each neighbour $x \neq w$ of a low degree neighbour $w_\mathcal{L}^j$ and for all $y \in \{w_\mathcal{L}^1, ..., w_\mathcal{L}^{k_\mathcal{L}}, w_\mathcal{H}^1, ..., w_\mathcal{H}^{k_\mathcal{H}}\} - \{w_\mathcal{L}^j\}$ we decrement $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{x, y\})$ by 1. Each low degree vertex has at most K neighbours, therefore the number of such pairs x, y is in $\mathcal{O}(K^3)$.

Now assume w was a high degree vertex and turns into a low degree vertex. Then w has $k_\mathcal{L} \leq K+1$ low degree neighbours $w_\mathcal{L}^1, ..., w_\mathcal{L}^{k_\mathcal{L}}$ and $k_\mathcal{H} \leq K+1$ high degree neighbours $w_\mathcal{H}^1, ..., w_\mathcal{H}^{k_\mathcal{H}}$. For each neighbour $x \neq w$ of a low degree neighbour $w_\mathcal{L}^j$ and for all $y \in \{w_\mathcal{L}^1, ..., w_\mathcal{L}^{k_\mathcal{L}}, w_\mathcal{H}^1, ..., w_\mathcal{H}^{k_\mathcal{H}}\} - \{w_\mathcal{L}^j\}$ we increment $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{x, y\})$ by 1. Each low degree vertex has at most K neighbours, therefore the number of such pairs x, y is in $\mathcal{O}(K^3)$. □

Lemma 8. *Let $H, G, I_G, I_\mathcal{L}, I_\mathcal{H}$ be as in definition 13. Then given an update $\{u, v\}$ where no vertex changes partition we can maintain $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{x, y\})$ in time $\mathcal{O}(K^2)$ for all pairs of distinct vertices $x, y \in V_G$.*

Proof. Let $\mathcal{N}_{\mathcal{L},u}^2 = \{w \in V_G - \{u, v\} : \exists w' \in \mathcal{N}_\mathcal{L}(u) - \{v\} : \{w, w'\} \in E_G\}$ be the set of neighours of low degree neighbours of u without u and v and analogously, let $\mathcal{N}_{\mathcal{L},v}^2 = \{w \in V_G - \{u, v\} : \exists w' \in \mathcal{N}_\mathcal{L}(v) - \{u\} : \{w, w'\} \in E_G\}$ be the set of neighours of low degree neighbours of v without u and v.

We distinguish four cases:

1. $u, v \in \mathcal{L}$: For all $x \in \mathcal{N}^2_{\mathcal{L},u}$, increment/decrement $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{x, v\})$ by 1. For all $x \in \mathcal{N}^2_{\mathcal{L},v}$, increment/decrement $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{x, u\})$ by 1. For all $x \in \mathcal{N}(u) - \{v\}, y \in \mathcal{N}(v) - \{u\}$, increment/decrement $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{x, y\})$ by 1. As u and v are low-degree vertices, they have $\mathcal{O}(K)$ neighbours and thus $\left|\mathcal{N}^2_{\mathcal{L},u}\right|, \left|\mathcal{N}^2_{\mathcal{L},v}\right| \in \mathcal{O}(K^2)$.

2. $u \in \mathcal{L}, v \in \mathcal{H}$: For all $x \in \mathcal{N}^2_\mathcal{L}(u)$, increment/decrement $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{x, v\})$ by 1. As u is a low-degree vertex, it has $\mathcal{O}(K)$ neighbours and thus $\left|\mathcal{N}^2_{\mathcal{L},u}\right| \in \mathcal{O}(K^2)$.

3. $u \in \mathcal{H}, v \in \mathcal{L}$: Analogous to the previous case with swapped roles for u and v.

4. $u, v \in \mathcal{H}$: Nothing needs to be done here.

□

Corollary 4. *Let $H, G, I_G, I_\mathcal{L}, I_\mathcal{H}$ be as in definition 13. Then we can maintain $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{x, y\})$ in time $\mathcal{O}(K^2)$ for all pairs of distinct vertices $x, y \in V_G$.*

Proof. Follows from lemma 7, 8 and theorem 1 □

3.2.2 Auxiliary Claws

Definition 14. Let G be a dynamic graph partitioned into low and high degree vertices as in theorem 1 with threshold K. We define H to be the claw, ie. $V_H = \{v_1, v_2, v_3, v_4\}$ and $E_H = \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_1, v_4\}\}$. (ie. v_1 is the center node) We also define $I_G = \{v_2, v_3\}, I_\mathcal{L} = \{v_1\}, I_\mathcal{H} = \emptyset$, ie. the center node has low degree and we fix two out of the three outer vertices. (see figure 3.1e).

Lemma 9. *Let $H, G, I_G, I_\mathcal{L}, I_\mathcal{H}$ be as in definition 14. Then given a vertex w which changes from being a low degree vertex to a high degree vertex or vice versa with $\delta'(w) \leq K$ we can maintain $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{x, y\})$ in time $\mathcal{O}(K^2)$ for all pairs of distinct vertices $x, y \in V_G$.*

Proof. First assume w was a low degree vertex and turns into a high degree vertex. For each unordered pair of distinct neighbours x, y of w we decrement $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{x, y\})$ by $\delta'(w) - 2$.

Now assume w was a high degree vertex and turns into a low degree vertex. For each unordered pair of distinct neighbours x, y of w we increment $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{x, y\})$ by $\delta'(w) - 2$.

Because of $\delta'(w) \leq K$, the number of pairs of neighbours of w is in $\mathcal{O}(K^2)$. □

Lemma 10. *Let $H, G, I_G, I_\mathcal{L}, I_\mathcal{H}$ be as in definition 14. Then given an update $\{u, v\}$ where no vertex changes partition we can maintain $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{x, y\})$ in time $\mathcal{O}(K^2)$ for all pairs of distinct vertices $x, y \in V_G$.*

Proof. We distinguish four cases:

1. $u, v \in \mathcal{L}$: For each unordered pair of distinct neighbours $w, w' \in \mathcal{N}(u) - \{v\}$ of u, we increment/decrement $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{w, w'\})$ by 1. For each unordered pair of distinct neighbours $w, w' \in \mathcal{N}(v) - \{u\}$ of v, we increment/decrement $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{w, w'\})$ by 1. For each neighbour $w \in \mathcal{N}(u) - \{v\}$, we increment/decrement $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{w, v\})$ by $\delta'(u) - 1$. For each neighbour $w \in \mathcal{N}(v) - \{u\}$, we increment/decrement $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{w, u\})$ by $\delta'(v) - 1$. As u and v are low-degree vertices, they have $\mathcal{O}(K)$ neighbours, thus this requires $\mathcal{O}(K^2)$ time.

2. $u \in \mathcal{L}, v \in \mathcal{H}$: For each unordered pair of neighbours $w, w' \in \mathcal{N}(u) - \{v\}$ of u, we increment/decrement $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{w, w'\})$ by 1. For every neighbour $w \in \mathcal{N}(u) - \{v\}$ of u, we increment/decrement $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{w, v\})$ by $\delta'(u) - 1$. As u is a low-degree vertex, it has $\mathcal{O}(K)$ neighbours, thus this requires $\mathcal{O}(K^2)$ time.

3. $u \in \mathcal{H}, v \in \mathcal{L}$: Analogous to the previous case with swapped roles for u and v.

4. $u, v \in \mathcal{H}$: Nothing needs to be done here.

□

Corollary 5. *Let $H, G, I_G, I_\mathcal{L}, I_\mathcal{H}$ be as in definition 14. Then we can maintain $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{x, y\})$ in time $\mathcal{O}(K^2)$ for all pairs of distinct vertices $x, y \in V_G$.*

Proof. Follows from lemma 9, 10 and theorem 1 □

Definition 15. Let G be a dynamic graph partitioned into low and high degree vertices as in theorem 1 with threshold K. We define H to be the claw, ie. $V_H = \{v_1, v_2, v_3, v_4\}$ and $E_H = \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_1, v_4\}\}$. (ie. v_1 is the center node) We also define $I_G = \{v_2, v_3, v_4\}, I_\mathcal{L} = \{v_1\}, I_\mathcal{H} = \emptyset$, ie. the center node has low degree and we fix all outer vertices. (see figure 3.1f).

Lemma 11. *Let $H, G, I_G, I_\mathcal{L}, I_\mathcal{H}$ be as in definition 15. Then given a vertex w which changes from being a low degree vertex to a high degree vertex or vice versa with $\delta'(w) \leq K$ we can maintain $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{x, y, z\})$ in time $\mathcal{O}(K^3)$ for all triples of distinct vertices $x, y, z \in V_G$.*

Proof. First assume w was a low degree vertex and turns into a high degree vertex. For each unordered triple of distinct neighbours x, y, z of w we decrement $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{x, y, z\})$ by 1.

Now assume w was a high degree vertex and turns into a low degree vertex. For each unordered triple of distinct neighbours x, y, z of w we increment $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{x, y, z\})$ by 1.

Because of $\delta'(w) \leq K$, the number of triples of neighbours of w is in $\mathcal{O}(K^3)$. □

Lemma 12. *Let $H, G, I_G, I_\mathcal{L}, I_\mathcal{H}$ be as in definition 15. Then given an update $\{u, v\}$ where no vertex changes partition we can maintain $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{x, y, z\})$ in time $\mathcal{O}(K^2)$ for all triples of distinct vertices $x, y, z \in V_G$.*

Proof. We distinguish four cases:

1. $u, v \in \mathcal{L}$: For each unordered pair of distinct neighbours $w, w' \in \mathcal{N}(u) - \{v\}$ of u, we increment/decrement $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{w, w', v\})$ by 1. For each unordered pair of distinct neighbours $w, w' \in \mathcal{N}(v) - \{u\}$ of v, we increment/decrement $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{w, w', u\})$ by 1. As u and v are low-degree vertices, they have $\mathcal{O}(K)$ neighbours and thus $\mathcal{O}(K^2)$ pairs of neighbours.

2. $u \in \mathcal{L}, v \in \mathcal{H}$: For each unordered pair of distinct neighbours $w, w' \in \mathcal{N}(u) - \{v\}$ of u, we increment/decrement $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{w, w', v\})$ by 1. As u is a low-degree vertex, it has $\mathcal{O}(K)$ neighbours and thus $\mathcal{O}(K^2)$ pairs of neighbours.

3. $u \in \mathcal{H}, v \in \mathcal{L}$: Analogous to the previous case with swapped roles for u and v.

4. $u, v \in \mathcal{H}$: Nothing needs to be done here.

□

Corollary 6. *Let $H, G, I_G, I_\mathcal{L}, I_\mathcal{H}$ be as in definition 15. Then we can maintain $c(H, G, I_G, I_\mathcal{L}, I_\mathcal{H}, \{x, y, z\})$ in time $\mathcal{O}(K^2)$ for all triples of distinct vertices $x, y, z \in V_G$.*

Proof. Follows from lemma 11, 12 and theorem 1 □

3.2.3 Auxiliary Triangles

Definition 16. Let G be a dynamic graph partitioned into low and high degree vertices as in theorem 1 with threshold K. We define H to be a triangle, ie. $V_H = \{v_1, v_2, v_3\}$ and $E_H = \{\{v_1, v_2\}, \{v_2, v_3\}, \{v_1, v_3\}\}$. We also define $I_G = \{v_1\}, I_{\mathcal{L}} = \emptyset, I_{\mathcal{H}} = \{v_1\}$, ie. one vertex has high degree and we fix that vertex. (see figure 3.1g).

Lemma 13. *Let $H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}$ be as in definition 16. Then given a vertex w which changes from being a low degree vertex to a high degree vertex or vice versa with $\delta'(w) \leq K$ we can maintain $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{x\})$ in time $\mathcal{O}(K^2)$ for all $x \in V_G$.*

Proof. First assume w was a low degree vertex and turns into a high degree vertex. Then for each unordered pair x, y of distinct neighbours of w with $\{x, y\} \in E_G$ we increment $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{w\})$ by 1.

Now assume w was a high degree vertex and turns into a low degree vertex. Then we set $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{w\}) = 0$.

Because of $\delta'(w) \leq K$, the number of pairs of neighbours of w is in $\mathcal{O}(K^2)$. □

Lemma 14. *Let $H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}$ be as in definition 16. Then given an update $\{u, v\}$ where no vertex changes partition we can maintain $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{x\})$ in time $\mathcal{O}(\max(K, \frac{|E_G|}{K}))$ for all $x \in V_G$.*

Proof. For each w in $\mathcal{H}$, we check whether w is adjacent to both u and v and if so, increment/decrement $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{w\})$ by 1. If $u \in \mathcal{H}$ or $v \in \mathcal{H}$, we also increment/decrement $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{u\})$ or $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{v\})$, respectively, by 1 for every such w. As there are $\mathcal{O}(\frac{|E_G|}{K})$ high degree vertices, this requires $\mathcal{O}(\frac{|E_G|}{K})$ time. If $u \in \mathcal{H}$, we further increment/decrement $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{u\})$ by the number of paths of length two connecting u and v via a low degree node. This number can be maintained in $\mathcal{O}(K)$ amortized time using corollary 2. If $v \in \mathcal{H}$, we also increment/decrement $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{v\})$ by the number of paths of length two connecting u and v via a low degree node. □

Corollary 7. *Let $H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}$ be as in definition 16. Then we can maintain $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{x\})$ in amortized time $\mathcal{O}(\max(K, \frac{|E_G|}{K}))$ for all $x \in V_G$.*

Proof. Follows from lemma 13, 14 and theorem 1 □

3.2.4 Auxiliary Paws

Definition 17. Let G be a dynamic graph partitioned into low and high degree vertices as in theorem 1 with threshold K. We define H to be a paw, ie. $V_H = \{v_1, v_2, v_3, v_4\}$ and $E_H = \{\{v_1, v_2\}, \{v_2, v_3\}, \{v_2, v_4\}, \{v_3, v_4\}\}$. We also define $I_G = \{v_1, v_3\}, I_{\mathcal{L}} = \{v_2, v_4\}, I_{\mathcal{H}} = \emptyset$, ie. the center vertex has low degree, as well as another vertex which is part of the triangle and we fix the other two vertices (see figures 3.1h and 3.2a).

Lemma 15. *Let $H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}$ be as in definition 17. Then given a vertex w which changes from being a low degree vertex to a high degree vertex or vice versa with $\delta'(w) \leq K$ we can maintain $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{x, y\})$ in time $\mathcal{O}(K^3)$ for all pairs of distinct vertices $x, y \in V_G$.*

Proof. First assume w was a low degree vertex and turns into a high degree vertex. For all ordered pairs x, y of neighbours of w with $\{x, y\} \in E_G$, where x is a low degree vertex, we decrement $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{z, y\})$ by 1 for every $z \in \mathcal{N}(x) - \{w, y\}$ and we decrement $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{z', y\})$ by 1 for every $z' \in \mathcal{N}(w) - \{x, y\}$.

Now assume w was a high degree vertex and turns into a low degree vertex. For all ordered pairs x, y of neighbours of w with $\{x, y\} \in E_G$, where x is a low degree vertex, we increment $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{z, y\})$ by 1 for every $z \in \mathcal{N}(x) - \{w, y\}$ and we increment $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{z', y\})$ by 1 for every $z' \in \mathcal{N}(w) - \{x, y\}$. (see figures 3.2b and 3.2c)

Because of $\delta'(w) \le K$, $\mathcal{N}(w) \in \mathcal{O}(K)$, the number of pairs of neighbours of w is in $\mathcal{O}(K^2)$ and $\mathcal{N}(x) \in \mathcal{O}(K)$ (because x is a low degree vertex), yielding a runtime of $\mathcal{O}(K^3)$. □

Lemma 16. *Let $H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}$ be as in definition 17. Then given an update $\{u, v\}$ where no vertex changes partition we can maintain $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{x, y\})$ in time $\mathcal{O}(K^2)$ for all pairs of distinct vertices $x, y \in V_G$.*

Proof. We distinguish four cases:

1. $u, v \in \mathcal{L}$: We distinguish the counts for each edge in the paw to which $\{u, v\}$ may be mapped to.

   (a) $\{v_1, v_2\}$ is mapped to $\{u, v\}$. (see figure 3.2d): For each ordered pair of neighbours $w, w' \in \mathcal{N}(u) - \{v\}$ of u with $\{w, w'\} \in E_G$ where w' is a low degree vertex we increment/decrement $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{v, w\})$ by 1. For each ordered pair of neighbours $w, w' \in \mathcal{N}(v) - \{u\}$ of v with $\{w, w'\} \in E_G$ where w' is a low degree vertex we increment/decrement $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{u, w\})$ by 1. As u and v are low-degree vertices, both have $\mathcal{O}(K^2)$ pairs of neighbours and this requires $\mathcal{O}(K^2)$ time.

   (b) $\{v_2, v_3\}$ is mapped to $\{u, v\}$. (see figure 3.2e): For each ordered pair of distinct neighbours $w, w' \in \mathcal{N}(u) - \{v\}$ of u, such that w is a low degree vertex and adjacent to v, we increment/decrement $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{v, w'\})$ by 1. For each ordered pair of distinct neighbours $w, w' \in \mathcal{N}(v) - \{u\}$ of v, such that w is a low degree vertex and adjacent to u, we increment/decrement $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{u, w'\})$ by 1. As u and v are low-degree vertices, both have $\mathcal{O}(K^2)$ pairs of neighbours and this requires $\mathcal{O}(K^2)$ time.

   (c) $\{v_2, v_4\}$ is mapped to $\{u, v\}$. (see figure 3.2f): For each ordered pair of distinct neighbours $w, w' \in \mathcal{N}(u) - \{v\}$ of u, such that w is adjacent to v, we increment/decrement $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{w, w'\})$ by 1. For each ordered pair of distinct neighbours $w, w' \in \mathcal{N}(v) - \{u\}$ of v, such that w is adjacent to u, we increment/decrement $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{w, w'\})$ by 1. As u and v are low-degree vertices, both have $\mathcal{O}(K^2)$ pairs of neighbours and this requires $\mathcal{O}(K^2)$ time.

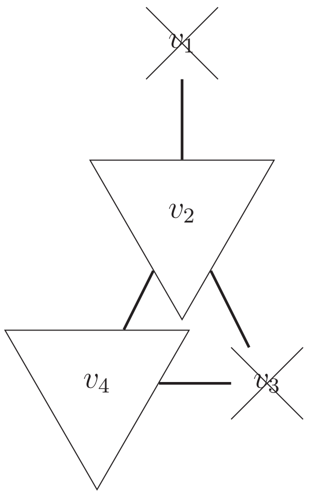
   (d) $\{v_3, v_4\}$ is mapped to $\{u, v\}$. (see figure 3.2g): For every low degree neighbour w of u, which is also adjacent to v, we increment/decrement $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{u, w'\})$ and $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{v, w'\})$ by 1 each for every $w' \in \mathcal{N}(w) - \{u, v\}$. As u is a low-degree vertex, it has $\mathcal{O}(K)$ low degree neighbours, which in turn also have $\mathcal{O}(K)$ neighbours, so this requires $\mathcal{O}(K^2)$ time.

2. $u \in \mathcal{L}, v \in \mathcal{H}$: Again, we distinguish the counts for each edge in the paw to which $\{u, v\}$ may be mapped to.

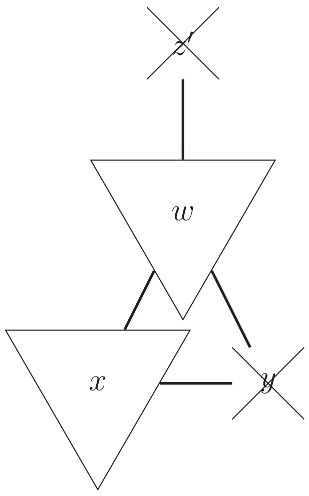
   (a) $\{v_1, v_2\}$ is mapped to $\{u, v\}$. (see figure 3.2d): For each ordered pair of neighbours $w, w' \in \mathcal{N}(u) - \{v\}$ of u with $\{w, w'\} \in E_G$ where w' is a low degree vertex we increment/decrement $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{v, w\})$ by 1. As u is a low-degree vertex, it has $\mathcal{O}(K^2)$ pairs of neighbours and this requires $\mathcal{O}(K^2)$ time.

   (b) $\{v_2, v_3\}$ is mapped to $\{u, v\}$. (see figure 3.2e): For each ordered pair of distinct neighbours $w, w' \in \mathcal{N}(u) - \{v\}$ of u, such that w is a low degree vertex and adjacent to v, we increment/decrement $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{v, w'\})$. As u is a low-degree vertex, it has $\mathcal{O}(K^2)$ pairs of neighbours and this requires $\mathcal{O}(K^2)$ time.

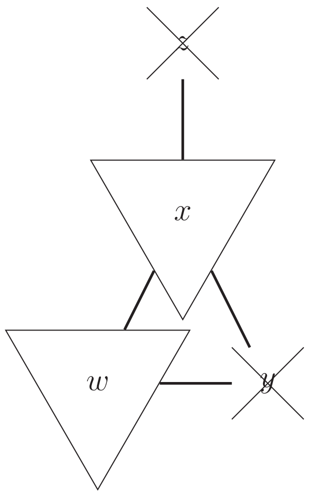
   (c) $\{v_3, v_4\}$ is mapped to $\{u, v\}$. (see figure 3.2g): For every low degree neighbour w of u, which is also adjacent to v, we increment/decrement $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{v, w'\})$ by 1 for every $w' \in \mathcal{N}(w) - \{u, v\}$. As u is a low-degree vertex, it has $\mathcal{O}(K)$ low



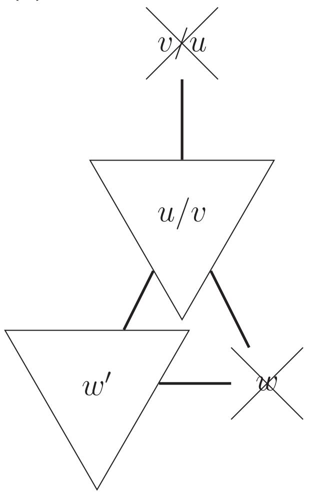
(a) Paw in definition 17



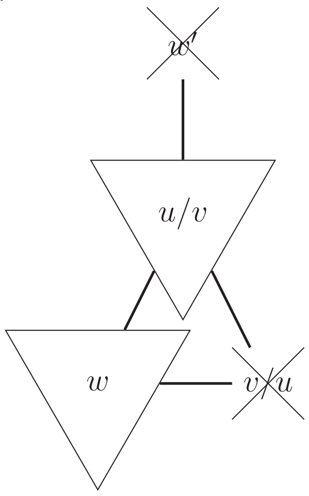
(b) Paw in proof of lemma 15



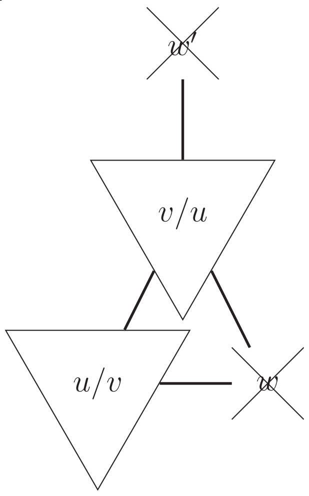
(c) Paw in proof of lemma 15



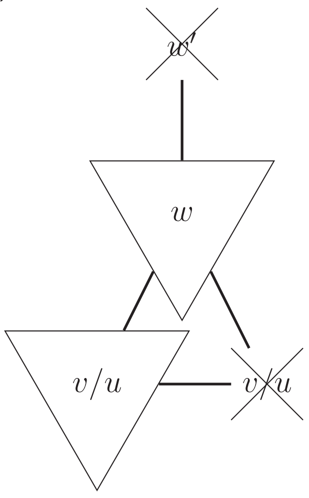
(d) Paw in proof of lemma 16



(e) Paw in proof of lemma 16



(f) Paw in proof of lemma 16



(g) Paw in proof of lemma 16

Figure 3.2: Paws used in subsection 3.2.4. Crossed out vertices are mapped to I_G under the subgraph isomorphism, upside down triangles are mapped to $I_{\mathcal{L}}$ under the subgraph isomorphism.

degree neighbours, which in turn also have $\mathcal{O}(K)$ neighbours, so this requires $\mathcal{O}(K^2)$ time.

3. $u \in \mathcal{H}, v \in \mathcal{L}$: Analogous to the previous case with swapped roles for u and v.
4. $u, v \in \mathcal{H}$: Nothing needs to be done here.

$\square$

Corollary 8. *Let $H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}$ be as in definition 17. Then we can maintain $c(H, G, I_G, I_{\mathcal{L}}, I_{\mathcal{H}}, \{x, y\})$ in amortized time $\mathcal{O}(K^2)$ for all pairs of distinct vertices $x, y \in V_G$.*

Proof. Follows from lemma 15, 16 and theorem 1 $\square$

Chapter 4

Non-induced Subgraph Isomorphism

Our algorithms for maintaining $c(H, G)$ given an updated edge e work the following way: If H is isomorphic to a subgraph of G and this subgraph contains e, then one edge $e' \in E_H$ is mapped to e under this isomorphism. Therefore we get the total number of subgraphs in G isomorphic to H if we count the number of subgraphs of G isomorphic to H, where e' is mapped to e for each $e' \in E_H$ separately and add these counts. These auxiliary counts are maintained using the procedures described in chapter 3 and they are updated before we update $c(H, G)$.

This is the same technique that is also used in the work by Eppstein et al. [EGST12], so our proofs are very similar. We repeat their algorithm for paths and claws, as we could not find an improvement. In the case of paw, cycle and diamond, we can skip one iteration through $\mathcal{H}$ compared to Eppstein et al. when counting certain subgraphs containing three or more high degree vertices, but otherwise copy their algorithm. As mentioned in chapter 2, skipping this iteration yields faster runtimes, as instead of having two nested loops over the neighbours of low degree vertices and two nested loops over the set of high degree vertices yielding a runtime of $\mathcal{O}\left(\max\left(K^2, \left(\frac{|E_G|}{K}\right)^2\right)\right) = \mathcal{O}(|E_G|)$ (for $K = \sqrt{E_G}$) we have two nested loops over the neighbours of low degree vertices and a single loop over the set of high degree vertices yielding a runtime of $\mathcal{O}\left(\max\left(K^2, \frac{|E_G|}{K}\right)\right) = \mathcal{O}(|E_G|^{\frac{2}{3}})$ (for $K = |E_G|^{\frac{1}{3}}$).

We will use theorem 1 to partition the vertices in G into low and high degree vertices. Unlike the auxiliary counts in chapter 3, we do not care when vertices switch the partition, because the total count $c(H, G)$ does not depend on the partition.

4.1 Paths

Theorem 2. *Let G be a dynamic graph. We can maintain $c(\ulcorner\!\!/, G)$ in amortized time $\mathcal{O}(\sqrt{|E_G|})$.*

Proof. We partition G into low and high degree vertices as in theorem 1 with threshold $K = |E_G|^{\alpha}$.

When an edge $\{u, v\}$ is inserted/deleted, three types of paths can arise/vanish: paths in which u is the endpoint, paths in which v is the endpoint and paths in which neither u nor v is the endpoint.

In the first case, $c(\ulcorner\!\!/, G)$ changes by the number of paths of length 2, in which v is the endpoint and the central node has low degree, which we can maintain in $O(K)$ amortized time by corollary 1, plus the number of paths of length 2, in which v is the endpoint and the central node has high degree. We can count the number of paths with endpoint v and a high degree central node in time $\mathcal{O}(\frac{|E_G|}{K})$ by adding $\delta(w) - 1$ for every $w \in (\mathcal{N}(v) \cap \mathcal{H}) - \{u\}$. In the case of inserting $\{u, v\}$, we have to correct for paths of length 2 which contain the edge $\{u, v\}$, if u is a low degree vertex. This can be done by subtracting $\delta'(u)$.

The second case is analogous to the first case.

In the third case, $c(\ulcorner\!\!/, G)$ changes by $\delta'(u) \cdot \delta'(v)$.

The choice $K = \sqrt{|E_G|}$ yields the desired runtime. $\square$

4.2 Claws

Theorem 3. *Let G be a dynamic graph. We can maintain $c(\curlywedge, G)$ in constant time.*

Proof. The number of claws with a vertex v as center node is $\binom{\delta(v)}{3}$ with the convention $\binom{n}{k} = 0$ for $n < k$. The total number of claws is thus given by

$$c(\curlywedge, G) = \sum_{v \in V_G} \binom{\delta(v)}{3}.$$

Maintaining $\delta(v)$ can be done in constant time. Thus, when inserting an edge $\{u, v\}$, the count has to be increased by $\binom{\delta'(u)+1}{3} - \binom{\delta'(u)}{3} + \binom{\delta'(v)+1}{3} - \binom{\delta'(v)}{3}$ and when deleting an edge $\{u, w\}$, the count has to be decreased by $\binom{\delta'(u)}{3} - \binom{\delta'(u)-1}{3} + \binom{\delta'(v)}{3} - \binom{\delta'(v)-1}{3}$ $\square$

4.3 Paws

Lemma 17. *Let G be a dynamic graph partitioned into low and high degree vertices as in theorem 1 with threshold $K = |E_G|^{\alpha}$. Given an update $\{u, v\}$, we can calculate the number of triangles containing $\{u, v\}$ (ie. the number of triangles after inserting or before removing $\{u, v\}$) in $\mathcal{O}(\max(K, \frac{|E_G|}{K}))$ amortized time.*

Proof. There are two types of triangles containing $\{u, v\}$: Triangles $\{u, v, w\}$, where w is a low degree vertex and triangles $\{u, v, w\}$, where w is a high degree vertex. The number of triangles, where w is a low degree vertex is the number of paths of length 2 between u and v, where the center node has low degree. This can be counted in $\mathcal{O}(K)$ amortized time by corollary 2. The number of triangles where w is a high degree vertex can be calculated by iterating through the set of high degree vertices $\mathcal{H}$ and for each of them checking whether it is adjacent to both u and v. As $|\mathcal{H}| \leq 12\frac{|E_G|}{K}$, this is done in $O(\frac{|E_G|}{K})$ time. $\square$

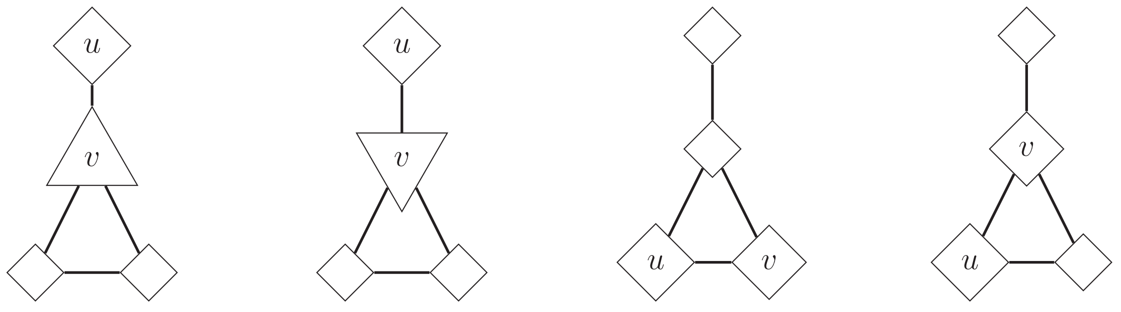
Theorem 4. *Let G be a dynamic graph. We can maintain $c(-\!\!\lhd, G)$ in amortized time $\mathcal{O}(|E_G|^{\frac{2}{3}})$.*

Proof. We partition G into low and high degree vertices as in theorem 1 with threshold $K = |E_G|^{\alpha}$.

When an edge $\{u, v\}$ is inserted/deleted, three types of paws can arise/vanish: $\{u, v\}$ can be the edge connecting the two vertices with degrees one and three (see figure 4.1a and 4.1b), $\{u, v\}$ can be the edge connecting the two vertices with degree two (see figure 4.1c) or $\{u, v\}$ can be one of the two edges connecting two vertices with degrees two and three (see figure 4.1d).

For the first type, wlog assume u is the vertex corresponding to the vertex with degree one in the paw and v is the vertex corresponding to the vertex with degree three in the paw, we can update analogously in the other case. The number of paws then changes by the number of triangles containing v but not u. If v is a high degree vertex (see figure 4.1a), using corollary 7 we can maintain the number of triangles containing v in time $\mathcal{O}(\max(K, \frac{|E_G|}{K}))$. If $\{u, v\}$ was inserted, this includes triangles containing $\{u, v\}$, so we subtract the number of triangles containing $\{u, v\}$ in time $\mathcal{O}(\max(K, \frac{|E_G|}{K}))$ using lemma 17. If v is a low degree vertex (see figure 4.1b), then for each unordered pair of vertices in $\mathcal{N}(v) - \{u\}$ we check whether this unordered pair forms a triangle with v, and calculate the number of triangles containing v but not u this way in time $\mathcal{O}(K^2)$. Thus counting the first type of paws requires $\mathcal{O}(\max(K^2, \frac{|E_G|}{K}))$ time.

For the second type (see figure 4.1c), we add/subtract the number of claws in which u and v are two of the three outer vertices to our count. The number of such claws with a low degree center node are maintained in $\mathcal{O}(K^2)$ time due to corollary 5, the number of such claws with a



(a) First Paw (b) Second Paw (c) Third Paw (d) Fourth Paw

Figure 4.1: Paws in the proof of theorem 4. $\bigtriangledown$ denotes low degree vertices, $\bigtriangleup$ denotes high degree vertices and $\diamond$ denotes arbitrary vertices.

high degree center node can be counted by checking for each high degree vertex $w \in \mathcal{H}$ whether it is adjacent to both u and v and if so, adding/subtracting $\delta(w) - 2$, which requires $\mathcal{O}(\frac{|E_G|}{K})$ time as there are $\mathcal{O}(\frac{|E_G|}{K})$ high degree vertices.

For the third type (see figure 4.1d), we calculate the number of triangles containing $\{u, v\}$ in time $\mathcal{O}(\max(K, \frac{|E_G|}{K}))$ using lemma 17. Each triangle contributes to $\delta'(u) - 1 + \delta'(v) - 1$ paws, so the product is added/subtracted from our overall count.

The choice $K = |E_G|^{\frac{1}{3}}$ yields the desired runtime. □

4.4 Cycles

Definition 18. Let $(\{a, b, c, d\}, \{\{a, b\}, \{b, c\}, \{c, d\}, \{d, a\}\})$ be a cycle. We refer to $\{c, d\}$ as the opposite edge of $\{a, b\}$

Theorem 5. *Let G be a dynamic graph. We can maintain $c(\diamond, G)$ in amortized time $\mathcal{O}(|E_G|^{\frac{2}{3}})$.*

Proof. We partition G into low and high degree vertices as in theorem 1 with threshold $K = |E_G|^{\alpha}$.

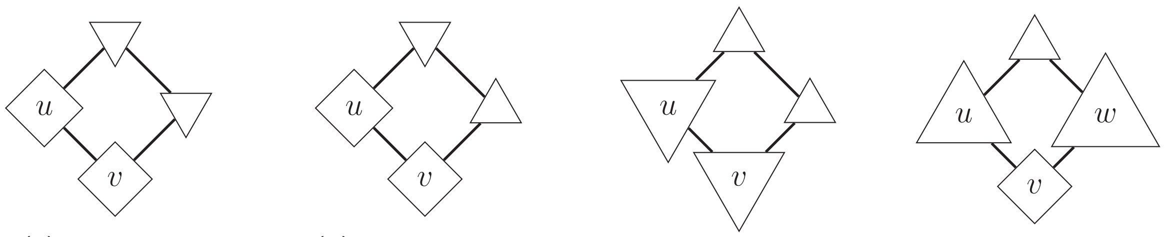
For an updated edge $\{u, v\}$ there are three types of cycles to account for:

1. Cycles in which the opposite edge is incident to two low degree vertices (see figure 4.2a): The number of these cycles is counted using corollary 4 in amortized time $\mathcal{O}(K^2)$ time.

2. Cycles in which the opposite edge is incident to a low and a high degree vertex (see figure 4.2b): Wlog assume u is adjacent to the low degree vertex in the opposite edge. (Ie. we run the following algorithm also with u and v swapped.) For each high degree vertex $w \in \mathcal{H} - \{u, v\}$ adjacent to v we add/subtract the number of paths of length 2 to u with a low degree vertex in the center of the path, which can be maintained using corollary 2 in $\mathcal{O}(K)$ amortized time. If v is a low-degree vertex, this also adds all paths through v, so for each high degree vertex adjacent to v, one has to be subtracted/added back. Since there are $\mathcal{O}(\frac{|E_G|}{K})$ high degree vertices, this can be done in $\mathcal{O}(\frac{|E_G|}{K})$ time.

   Counting this type of cycles thus requires $\mathcal{O}(\max(K, \frac{|E_G|}{K}))$ time.

3. Cycles in which the opposite edge is incident to two high degree vertices: If u and v are both low degree vertices (see figure 4.2c), then for all $x \in \mathcal{N}_{\mathcal{H}}(u) - \{v\}, y \in \mathcal{N}_{\mathcal{H}}(v) - \{u\}$ we increment/decrement the cycle count by 1 if x and y are adjacent. As u and v have low degree, this takes $\mathcal{O}(K^2)$ time. Otherwise wlog. assume u has high degree (see figure 4.2d). For every high degree vertex w which is adjacent to v we increment/decrement the cycle count by the number of paths of length 2 with a high degree vertex in the center



(a) First Cycle (b) Second Cycle (c) Third Cycle (d) Fourth Cycle

Figure 4.2: Cycles in the proof of theorem 5. ▽ denotes low degree vertices, △ denotes high degree vertices and ◇ denotes arbitrary vertices.

of the path connecting u and w. The number of paths of length 2 with a high degree vertex in the center of the path can be counted using corollary 3in $\mathcal{O}(\frac{|E_G|}{K})$ amortized time. If v is also a high degree vertex, the paths include the path $u - v - w$, so 1 must be subtracted/added back to the cycle count for each high degree vertex w adjacent to v.

Counting this type of cycles thus also requires $\mathcal{O}(\max(\frac{|E_G|}{K}, K^2))$ time.

The choice $K = |E_G|^{\frac{1}{3}}$ yields the desired runtime. □

4.5 Diamonds

Theorem 6. *Let G be a dynamic graph. We can maintain $c(\Diamond, G)$ in amortized time $\mathcal{O}(|E_G|^{\frac{2}{3}})$.*

Proof. We partition G into low and high degree vertices as in theorem 1 with threshold $K = |E_G|^{\alpha}$.

An updated edge $\{u, v\}$ could contribute to two types of diamonds: Diamonds in which this edge is the chord of the surrounding cycle and diamonds in which this edge is part of the cycle.

To count the diamonds whose chord is $\{u, v\}$, we distinguish two cases:

1. u or v is a low degree vertex (see figure 4.3a): Wlog assume u is a low degree vertex. For all unordered pairs of neighbours $x \neq y \in \mathcal{N}(u) - \{v\}$ we check whether we have $\{x, v\}, \{y, v\} \in E_G$, and if so, increment/decrement our count by 1. As u has low degree, $|\mathcal{N}(u)| \leq K$, so this can be done in $\mathcal{O}(K^2)$ time.

2. u and v are high degree vertices (see figure 4.3b): By corollary 2 we can count the paths of length 2 with low degree center node connecting u and v in $\mathcal{O}(K)$ time. By corollary 3 we can count paths of length 2 with high degree center node connecting u and v in $\mathcal{O}(\frac{|E_G|}{K})$ time. Therefore we can determine the number of paths of length two connecting u and v in time $\mathcal{O}(\max(\frac{|E_G|}{K}, K))$. Let this number be ν. Then we increment/decrement the diamond count by $\frac{(\nu-1)\nu}{2}$.

We conclude that we can count the diamonds whose chord is $\{u, v\}$ in time $\mathcal{O}(\max(\frac{|E_G|}{K}, K^2))$.

To count the diamonds where $\{u, v\}$ is part of the cycle (ie. out of $\{u, v\}$ one vertex is incident to the diamond's chord and one is not, and out of the other two vertices one vertex is incident to the diamond's chord and one is not), we distinguish four cases:

1. The other vertices in the diamond are low degree vertices: (see figure 4.3c) The number of this type of diamonds is given by the number of paws in which $\{u, v\}$ are mapped to the vertices with degree one and one of the vertices with degree two in the paw, which is maintained in $\mathcal{O}(K^2)$ time using corollary 8.

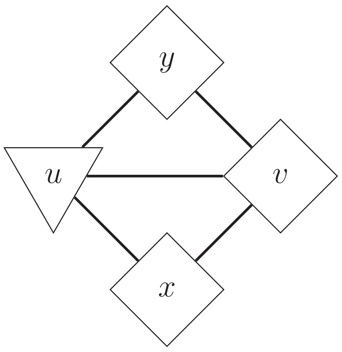
2. The other vertex incident to the diamond's chord is a low degree vertex, and the fourth vertex of the diamond is a high degree vertex: (see figure 4.3d) This type of diamonds can be counted by iterating through the set of high degree vertices in $\mathcal{O}(\frac{|E_G|}{K})$ time, and for each $w \in \mathcal{H} - \{u, v\}$ adding the number of claws with a low-degree center vertex and u, v, w as outer vertices. The number of such claws is counted in corollary 6 in time $O(K^2)$.

3. The other vertex incident to the diamond's chord is a high degree vertex, and the fourth vertex of the diamond is a low degree vertex: (see figure 4.3e) This type of diamonds can be counted by iterating through the set of high degree vertices in $\mathcal{O}(\frac{|E_G|}{K})$ time, and for each $w \in \mathcal{H} - \{u, v\}$ adjacent to both u and v adding/subtracting the number of paths of length 2 with a low degree center node between w and u and the number of paths of length 2 with a low degree center node between w and v. The number of these paths can be maintained in time $\mathcal{O}(K)$ using corollary 2. If u or v are low degree vertices, these paths contain paths with u or v being the center node of the path, so for each low degree vertex in $\{u, v\}$, one has to be subtracted/added back to the diamond count.

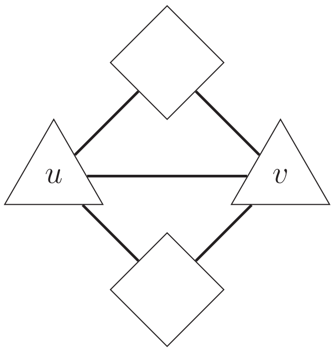
4. Both other vertices are high degree vertices: Wlog assume u is incident to the chord (ie. we do the same procedure with u and v swapped). We further distinguish two subcases: If u is a low degree vertex (see figure 4.3f), we iterate through all ordered pairs of high-degree neighbours $(x, y), x \neq y \in \mathcal{N}_{\mathcal{H}}(u) - \{v\}$ and for each pair we check whether we have $\{x, v\}, \{x, y\} \in E_G$ and if so, increment/decrement our count by 1. This requires $\mathcal{O}(K^2)$ time, as $|\mathcal{N}_{\mathcal{H}}(u)| \leq K$. If u is a high degree vertex (see figure 4.3g), we check for each high degree vertex $w \in \mathcal{H} - \{u, v\}$, whether we have $\{u, w\}, \{v, w\} \in E_G$ and if so, increase/decrease our diamond count by the number of paths of length 2 connecting u and w via a high degree node. If v is a high degree vertex, the diamond count has to be decremented/incremented by one, as the paths of length 2 also include the path $u - v - w$. The number of these paths can be maintained in $\mathcal{O}(\frac{E_G}{K})$ time according to corollary 3. The number of high degree vertices is bounded by $\mathcal{O}(\frac{E_G}{K})$, thus this type of diamonds can be counted in $\mathcal{O}(\frac{E_G}{K})$ time.

We conclude that we can count the diamonds where $\{u, v\}$ is part of the cycle in time $\mathcal{O}(\max(\frac{|E_G|}{K}, K^2))$.

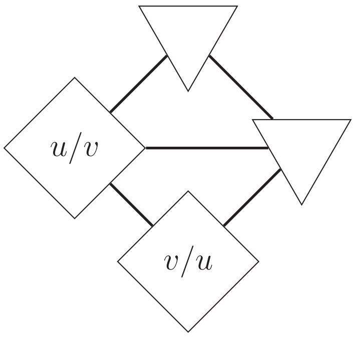
The choice $K = |E_G|^{\frac{1}{3}}$ thus yields the desired runtime. □



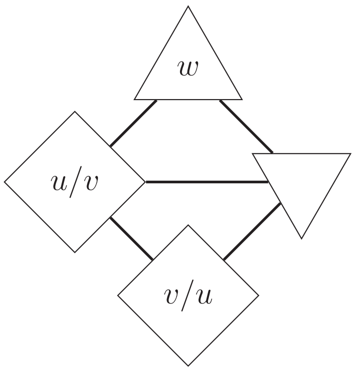
(a) First Diamond



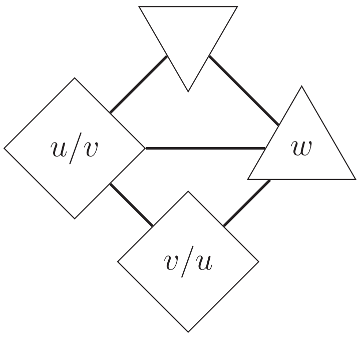
(b) Second Diamond



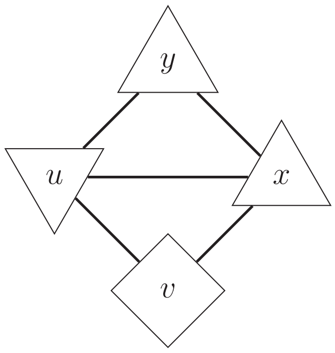
(c) Third Diamond



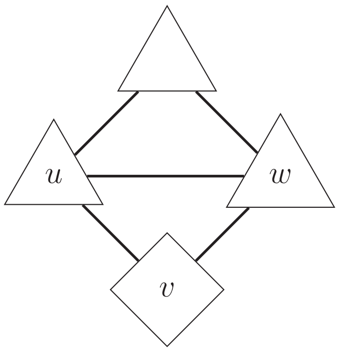
(d) Fourth Diamond



(e) Fifth Diamond



(f) Sixth Diamond



(g) Seventh Diamond

Figure 4.3: Diamonds in the proof of theorem 6. ▽ denotes low degree vertices, △ denotes high degree vertices and ◇ denotes arbitrary vertices.

Chapter 5

Induced Subgraph Isomorphism

As mentioned in chapter 2, we will show that computing the number of induced subgraph isomorphisms $c_I(H, G)$ is as hard as counting 4-cliques. Our hardness proof uses equations for these counts $c_I(H, G)$ in terms of the number of 4-cliques in G and the counts of different non-induced subgraph isomorphisms $c(H', G)$ which can be maintained in time $o(|E_G|)$ as shown in chapter 4. These equations for four-vertex subgraphs have been used by Eppstein et al. [EGST12] to obtain algorithms counting $c_I(H, G)$. We first show a derivation of these equations and how they are used to count $c_I(H, G)$ in the work of Eppstein et al., after which we will be able to show our hardness result.

The following well-known theorem (eg. theorem 2.3 in [Koc82]) is the main idea behind these equations:

Theorem 7. *Let G, H be any graph and let $\mathcal{S}$ be the set of all graphs on $|V_H|$ nodes. Then*

$$c(H, G) = \sum_{S \in \mathcal{S}} c(H, S) \cdot c_I(S, G).$$

We give a derivation in the case of connected four-vertex graphs:

Theorem 8. *Let G be any graph, H be a connected graph with four nodes and let $\mathcal{S}$ be the set of all connected graphs on four nodes. Then*

$$c(H, G) = \sum_{S \in \mathcal{S}} c(H, S) \cdot c_I(S, G).$$

Proof. For the proof of this theorem, we will write $A \sim B$ if the graphs A and B are isomorphic.

First note

$$c(H, G) = \sum_{\substack{a,b,c,d \in V_G, \\ a,b,c,d \text{ pairwise distinct}}} c(H, G[\{a, b, c, d\}]).$$

Because H is connected, $c(H, G[\{a, b, c, d\}]) \neq 0$ implies that a, b, c, d are connected, thus

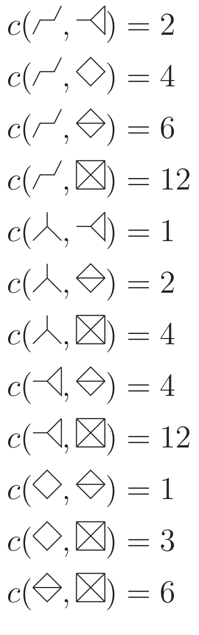
$G[\{a,b,c,d\}]$ is connected, therefore $G[\{a,b,c,d\}] \in \mathcal{S}$. We conclude

$$\begin{aligned}
c(H,G) &= \sum_{\substack{a,b,c,d \in V_G,\\ a,b,c,d \text{ pairwise distinct},\\ G[\{a,b,c,d\}] \in \mathcal{S}}} c(H, G[\{a,b,c,d\}]) \\
&= \sum_{S \in \mathcal{S}} \sum_{\substack{a,b,c,d \in V_G,\\ a,b,c,d \text{ pairwise distinct},\\ G[\{a,b,c,d\}] \sim S}} c(H, G[\{a,b,c,d\}]) \\
&= \sum_{S \in \mathcal{S}} \sum_{\substack{a,b,c,d \in V_G,\\ a,b,c,d \text{ pairwise distinct},\\ G[\{a,b,c,d\}] \sim S}} c(H,S) \\
&= \sum_{S \in \mathcal{S}} c(H,S) \cdot \sum_{\substack{a,b,c,d \in V_G,\\ a,b,c,d \text{ pairwise distinct},\\ G[\{a,b,c,d\}] \sim S}} 1 \\
&= \sum_{S \in \mathcal{S}} c(H,S) \cdot c_I(S,G).
\end{aligned}$$

□

Since $H, S \in \mathcal{S}$ and $|\mathcal{S}| = 6$, the coefficients $c(H,S)$ in theorem 8 can be determined explicitly:

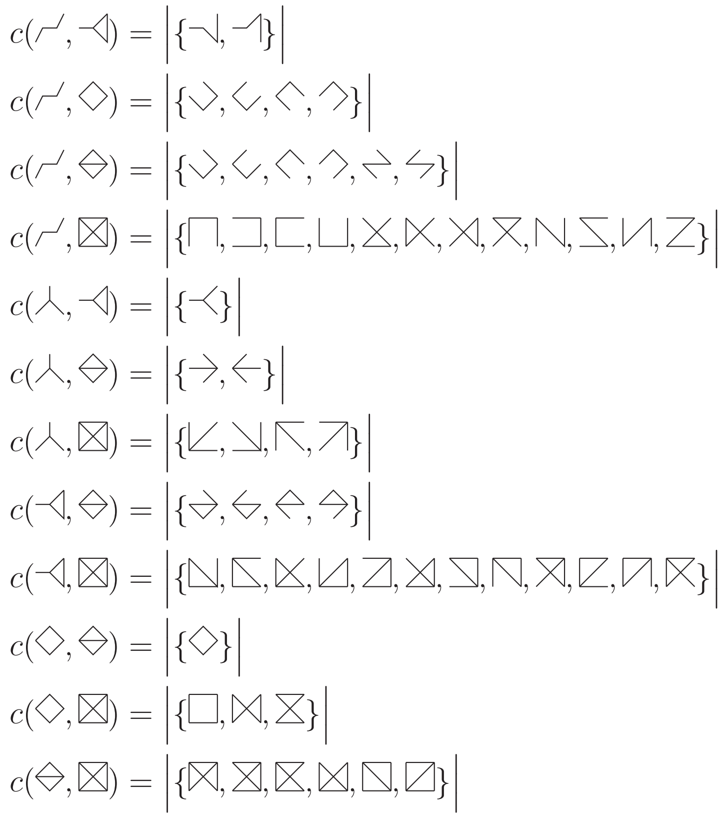
Lemma 18.



If G, H are connected graphs on four nodes and $c(H,G)$ is not mentioned in this list, then $c(H,G) = 1$ if G and H are isomorphic and $c(H,G) = 0$ otherwise.

Proof. The non-isomorphic pairs of graphs G, H not mentioned in this list have in common that $|E_H| \geq |E_G|$. If $|E_H| > |E_G|$, then obviously $c(H,G) = 0$, and if $|E_H| = |E_G|$, then any proper subgraph of G has fewer edges than H, so it cannot be isomorphic to H and G itself is also not isomorphic to H due to our assumption, therefore we also have $c(H,G) = 0$.

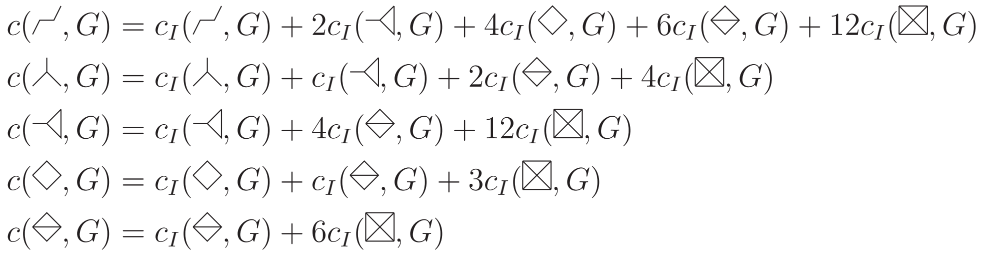
We have



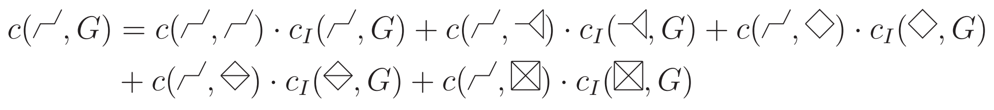
□

Now we can plug these coefficients into the equations of theorem 8.

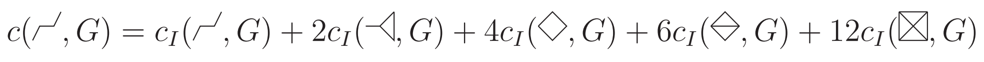
Corollary 9. *Let G be any graph. Then*



Proof. By theorem 8, we have



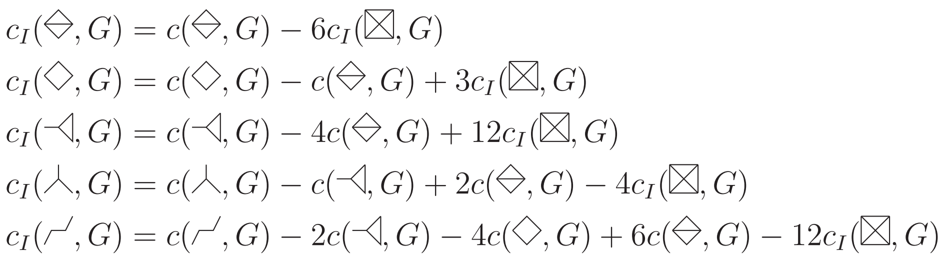
Plugging in the equations of lemma 18 yields



The other equations follow similarly. □

The equations in corollary 9 can be rearranged for $c_I(H, G)$.

Corollary 10. *Let G be any graph. Then*



Proof. Going from top to bottom (◇ to ⌐), each equation for $c_I(H, G)$ can be obtained by inserting the previous equations into the equation for $c(H, G)$ in corollary 9. □

The right hand sides of the equations in corollary 10 are linear combinations of $c(H, G)$. (Note that $c(\boxtimes, G) = c_I(\boxtimes, G)$) We have developed algorithms for maintaining $c(H, G)$ for any connected graph on four nodes, which is not a 4-clique, in chapter 4. For $c(\boxtimes, G)$, we have the following result in the literature:

Theorem 9. *([DLS20], [EGST12]) Let G be a dynamic graph. Then $c(\boxtimes, G) = c_I(\boxtimes, G)$ can be maintained in $\mathcal{O}(|E_G|)$ time.*

Corollary 11. *([EGST12]) Let G be a dynamic graph, let H be a connected graph on four nodes. Then $c_I(H, G)$ can be maintained in $\mathcal{O}(|E_G|)$ time.*

Proof. By chapter 4 we can maintain $c(S, G)$ for any connected graph on four nodes S, which is not a 4-clique, in time $o(|E_G|)$. By theorem 9 we can maintain $c_I(\boxtimes, G)$ in time $\mathcal{O}(|E_G|)$. Plugging these values into the equations of corollary 10 requires constant time and yields $c_I(H, G)$. □

In the remainder of this section we will show conditional optimality for this runtime. The optimality is based on the following conjecture (see eg. [BHG+21],[LWW18],[ABW15]):

Conjecture 1. *For any constant $\varepsilon > 0$, there is no $\mathcal{O}(n^{k-\varepsilon})$ time combinatorial algorithm for static k-clique detection with error probability at most $\frac{1}{3}$ in the word-RAM model with $\mathcal{O}(\log n)$ bit words.*

Lemma 19. *Let G, H be graphs, $t : \mathbb{N} \to \mathbb{R}_{\geq 0}$. If there exists a fully-dynamic algorithm to maintain $c(H, G)$ in amortized time $\mathcal{O}(t(|E_G|))$, where t is non-decreasing, then there exists a static $\mathcal{O}(|E_G| \cdot t(|E_G|))$ algorithm for computing $c(H, G)$ in the static setting. If there exists a fully-dynamic algorithm maintaining $c_I(H, G)$ in amortized time $\mathcal{O}(t(|E_G|))$, where t is non-decreasing, then there exists a static $\mathcal{O}(|E_G| \cdot t(|E_G|))$ algorithm for computing $c_I(H, G)$ in the static setting.*

Proof. We insert one edge of G after the other. The runtime is the sum of each update time:

$$\sum_{i=1}^{|E_G|} t(i) \leq \sum_{i=1}^{|E_G|} t(|E_G|) = |E_G| \cdot t(|E_G|)$$

□

Corollary 12. *Under conjecture 1, the runtime in corollary 11 is optimal in the sense that for any constant $\varepsilon > 0$, there is no $\mathcal{O}(|E_G|^{1-\varepsilon})$ combinatorial algorithm for maintaining $c_I(H, G)$ for any connected four node subgraph H.*

Proof. If $c_I(H, G)$ could be maintained in $\mathcal{O}(|E_G|^{1-\varepsilon})$ time, then the number of cliques could also be maintained in $\mathcal{O}(|E_G|^{1-\varepsilon})$ time, because we can maintain $c(S, G)$ for any connected graph on four nodes S, which is not a 4-clique, in time $O(|E_G|^{\frac{2}{3}})$ by the theorems in chapter 4, plug $c(S, G)$ and $c_I(H, G)$ into the equation for $c_I(H, G)$ in corollary 10 and solve the equation for $c_I(\boxtimes, G)$. Lemma 19 would then yield a $\mathcal{O}(|E_G|^{2-\varepsilon}) = O(|V_G|^{4-2\varepsilon})$ algorithm for counting four-cliques contradicting conjecture 1. □

Chapter 6

Conclusion

Our algorithms allow us to maintain counts of non-induced claws in constant time, paths of length three in time $\mathcal{O}(\sqrt{m})$ and paws, cycles of length four and diamonds in time $\mathcal{O}(m^{\frac{2}{3}})$. Whether our algorithms for non-induced connected four-vertex subgraphs may be further improved is an interesting question for future work. Because our algorithms maintain non-induced subgraphs in sublinear time, we have seen that counting the number of induced connected four-vertex subgraphs is equally hard, regardless of which subgraph being counted. In particular, an algorithm counting any induced connected four-vertex subgraph in $\mathcal{O}(n^{4-\varepsilon})$ time would already allow us to count every other induced connected four-vertex subgraph in $\mathcal{O}(n^{4-\varepsilon})$ time and falsify the conjecture that 4-cliques cannot be detected in $\mathcal{O}(n^{4-\varepsilon})$ time.

Another question for future work is whether the same results extend to the case of general k-vertex subgraphs: Can their non-induced counts also be maintained in $\mathcal{O}(m^{\frac{k}{2}-1})$ time and if so, can the counts of every k-vertex subgraph, which is not the k-clique, also be maintained in time $\mathcal{O}(m^{\frac{k}{2}-1-\varepsilon})$? If this is true, then all induced subgraph counting problems for subgraphs with the same number of vertices would be equally hard and any breakthrough would falsify the clique conjecture 1.

Bibliography

[ABW15] Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. If the current clique algorithms are optimal, so is valiant's parser. In *Proceedings of the 2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*, FOCS '15, page 98–117, USA, 2015. IEEE Computer Society.

[AW21] Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In *Proceedings of the Thirty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '21, page 522–539, USA, 2021. Society for Industrial and Applied Mathematics.

[AYZ94] Noga Alon, Raphael Yuster, and Uri Zwick. Finding and counting given length cycles (extended abstract). In *Proceedings of the Second Annual European Symposium on Algorithms*, ESA '94, page 354–364, Berlin, Heidelberg, 1994. Springer-Verlag.

[BFL$^+$06] Luciana S. Buriol, Gereon Frahling, Stefano Leonardi, Alberto Marchetti-Spaccamela, and Christian Sohler. Counting triangles in data streams. In *Proceedings of the Twenty-Fifth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '06, page 253–262, New York, NY, USA, 2006. Association for Computing Machinery.

[BHG$^+$21] Thiago Bergamaschi, Monika Henzinger, Maximilian Probst Gutenberg, Virginia Vassilevska Williams, and Nicole Wein. New techniques and fine-grained hardness for dynamic near-additive spanners. In *Proceedings of the Thirty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '21, page 1836–1855, USA, 2021. Society for Industrial and Applied Mathematics.

[BYKS02] Ziv Bar-Yossef, Ravi Kumar, and D. Sivakumar. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '02, page 623–632, USA, 2002. Society for Industrial and Applied Mathematics.

[DLS20] Laxman Dhulipala, Quanquan C. Liu, and Julian Shun. Parallel batch-dynamic k-clique counting, 2020.

[DT13] Zdeněk Dvořák and Vojtěch Tůma. A dynamic data structure for counting subgraphs in sparse graphs. In *Proceedings of the 13th International Conference on Algorithms and Data Structures*, WADS'13, page 304–315, Berlin, Heidelberg, 2013. Springer-Verlag.

[EGST12] David Eppstein, Michael T. Goodrich, Darren Strash, and Lowell Trott. Extended dynamic subgraph statistics using h-index parameterized data structures. *Theoretical Computer Science*, 447:44–52, 2012. Combinational Algorithms and Applications (COCOA 2010).

[ES09] David Eppstein and Emma S. Spiro. The h-index of a graph and its application to dynamic subgraph statistics. In *Proceedings of the 11th International Symposium* *on Algorithms and Data Structures*, WADS '09, page 278–289, Berlin, Heidelberg, 2009. Springer-Verlag.

[KKM95] Ton Kloks, Dieter Kratsch, and Haiko Müller. Finding and counting small induced subgraphs efficiently. In *Proceedings of the 21st International Workshop on Graph-Theoretic Concepts in Computer Science*, WG '95, page 14–23, Berlin, Heidelberg, 1995. Springer-Verlag.

[KNN+19] Ahmet Kara, Hung Q. Ngo, Milos Nikolic, Dan Olteanu, and Haozhe Zhang. Counting Triangles under Updates in Worst-Case Optimal Time. In Pablo Barcelo and Marco Calautti, editors, *22nd International Conference on Database Theory (ICDT 2019)*, volume 127 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 4:1–4:18, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[Koc82] William L Kocay. Some new methods in reconstruction theory. In *Combinatorial Mathematics IX*, pages 89–114. Springer, 1982.

[KPP+14] Tamara G. Kolda, Ali Pinar, Todd Plantenga, C. Seshadhri, and Christine Task. Counting triangles in massive graphs with mapreduce. *SIAM Journal on Scientific Computing*, 36(5):S48–S77, 2014.

[LWW18] Andrea Lincoln, Virginia Vassilevska Williams, and Ryan Williams. Tight hardness for shortest cycles and paths in sparse graphs. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '18, page 1236–1252, USA, 2018. Society for Industrial and Applied Mathematics.

[MSOI+02] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: Simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

[ZSSH14] Xin Zhang, Shuai Shao, H. Eugene Stanley, and Shlomo Havlin. Dynamic motifs in socio-economic networks. *EPL (Europhysics Letters)*, 108(5):58001, dec 2014.