A family of centrality measures for graph data based on subgraphs

Cristian Riveros
Pontificia Universidad Católica de Chile & IMFD
cristian.riveros@uc.cl

Jorge Salas
Pontificia Universidad Católica de Chile & IMFD
jusalas@uc.cl

Abstract

We present the theoretical foundations of a new approach in centrality measures for graph data. The main principle of our approach is very simple: the more relevant subgraphs around a vertex, the more central it is in the network. We formalize the notion of "relevant subgraphs" by choosing a family of subgraphs that, give a graph G and a vertex v in G, it assigns a subset of connected subgraphs of G that contains v. Any of such families defines a measure of centrality by counting the number of subgraphs assigned to the vertex, i.e., a vertex will be more important for the network if it belongs to more subgraphs in the family. We show many examples of this approach and, in particular, we propose the all-subgraphs centrality, a centrality measure that takes every subgraph into account. We study fundamental properties over families of subgraphs that guarantee desirable properties over the corresponding centrality measure. Interestingly, all-subgraphs centrality satisfies all these properties, showing its robustness as a notion for centrality. Finally, we study the computational complexity of counting certain families of subgraphs and show a polynomial time algorithm to compute the all-subgraphs centrality for graphs with bounded tree width.

2012 ACM Subject Classification Mathematics of computing → Graph theory; Information systems → Graph-based database models

Keywords and phrases Graph data, graph centrality, centrality measures

Digital Object Identifier 10.4230/LIPIcs.ICDT.2020.25

Acknowledgements C. Riveros and J. Salas were partially funded by the Millennium Institute for Foundational Research on Data.

1 Introduction

Which are the most important or "central" nodes in a network? This is a crucial question that has been asked in several areas like social science [23], biology [20], computer science [10] and essentially every area where graph data is relevant [27]. Given the graph structure of data one expects that more central nodes are more important for the network and they will be relevant in understanding its underlying structure. Several centrality measures has been proposed like closeness [4], betweenness [17], Page Rank [10], Katz index [22], among others [27], trying to give an answer or explanation to our first question.

Which centrality measure is the most meaningful for network analysis? This has been behind all proposals of centrality measures and it is an old question that has been discussed from the beginning of network analysis [8, 16, 29]. Over the years, some axioms or properties have been risen as crucial for a centrality measure and several centrality measures has been axiomatized [30, 31, 33]. However, as it was shown in [7] many commonly used centrality measures do not satisfy even a simple set of "desirable" axioms (i.e. properties). The question above then remains unanswered: how to naturally and formally define a centrality measure that has reasonable properties?

To motivate our approach that aims both questions, consider the following setting from graph data management. Suppose a graph database G and a query language $\mathcal{L}$ for extracting patterns from G. Further, suppose Q is a query in $\mathcal{L}$ such that the evaluation of Q over G, denoted by $Q(G)$, retrieves a set of nodes in G. How should we rank $Q(G)$ in order to output the most meaningful outputs first? More specifically, suppose that G is a property graph and $\mathcal{L}$ is a language of basic graph patterns [1]. Given that queries dynamically change over time [11, 24], one would expect that, if $v \in Q(G)$ satisfies more patterns from $\mathcal{L}$, it will have more chances to appear latter as an extension of Q. More general, one would expect that the more queries from $\mathcal{L}$ where v is included, the more important is v on G with respect to $\mathcal{L}$. Furthermore, depending whether $\mathcal{L}$ is designed to look for paths, trees, or maybe triangles [1] on G, maybe the user would like its measure of centrality to focus more on these patterns than in all basic graph patterns.

In this paper, we tackle the first question following the simple idea motivated from the graph data management setting: the more relevant subpatterns around a node, the more central it is in the network. Several proposals in the literature (e.g degree, betweenness [17], cross-clique [15]) already have considered relevant subpatterns like edges, paths, or cliques to define meaningful centrality measures. We generalize this approach by defining centrality measures based on families of subgraphs. Specifically, we formalize the notion of "relevant subgraphs" by choosing any family of subgraphs that, given a graph G and a vertex v in G, it assigns a subset of connected subgraphs from G that contains v. Any of such families defines a measure of centrality by counting the number of subgraphs assigned to the vertex, i.e., a vertex will be more important for the network if it belongs to more subgraphs in the family. We show several examples that can be derived by following this approach. In particular, a natural family of subgraphs is to consider all connected subgraphs around a vertex, that we called *all-subgraphs centrality*, and we show that it defines a well-behaved notion of centrality.

With a family of centrality measures at hand we embark on answering the second question. Generally speaking, we can consider any property on the family of subgraphs and see what "axiom" it implies in the respectively centrality notion that it defines. With this strategy, we no longer depend on comparing centrality measures of different nature (e.g. Page Rank vs Betweenness). Instead, we can understand all centrality notions proposed in this paper by just understanding the properties that satisfy the families of subgraphs. We consider simple axioms that has been proposed in the literature (e.g. monotonicity [29] or isolated vertex [16]). Then, look for simple properties in the family of subgraph that imply them. Interestingly, we can show natural examples of families of subgraphs that do not satisfy these properties and whose corresponding centrality notions do not satisfy the axioms. This allows to have a more deep understanding of why a centrality measure does not behave as expected and, moreover, to look into ways on how to "fix" it. Finally, the all-subgraphs centrality proposed in this paper satisfies all these properties and axioms, showing its robustness as a measure for centrality.

The general definition of centrality based on subgraphs allows us to easily extend the idea from vertices to sets of vertices, also called group centrality. We propose an approach to extend every centrality measure to groups, and prove a natural way to reduce the computation of all-subgraph centrality from groups to vertices. We show that this extension over sets allows to answer simple questions on the dynamic of graphs, like how to maximize the centrality of a vertex when an edge is added.

Towards the end of the paper, we study the computational complexity of counting certain families of subgraphs. Unfortunately, we show that the centrality measures defined from families of subgraphs like all subgraphs or trees lead to intractability. In terms of good news, we show that these centralities can be efficiently computed in acyclic graphs (i.e. trees). Moreover, we show that this result can be extended to more classes of graphs, by showing a polynomial time algorithm to compute the all-subgraphs centrality over all classes of graphs with bounded tree width.

Related work. Centrality measures have been extensively studied since the 50's [4, 23] and the subject is spread in different research areas. Moreover, the literature contains several alternative proposals that, given space restrictions, it will be impossible to cover all of them here (see [27]). Instead, we review here the work that is more closed in spirit to our proposal by stressing the main differences.

Centrality measures based on some relevant subgraphs have been studied before (e.g. betweenness [17], cross-clique [15]). The difference with our approach is that we take a step further and studied families of subgraphs in a more general setting. In particular, to the best of our knowledge all-subgraphs centrality and trees centrality (see Section 3 and 4) are new measures and have not been studied before.

There are several papers that have studied centrality measures in terms of properties [8, 16, 29]. Furthermore, in the last years there are several proposals to axiomatize standard centrality measures [6, 7, 30, 31, 33]. In this paper, we study properties and axioms in terms of families of subgraphs, which is a different goal compared to previous approaches.

Finally, a centrality measure called subgraph centrality was proposed in [14]. Although the name resemble our approach and the paper also motivates the use of subgraphs, subgraph centrality sums the number of closed-walks weighted by its length and not all the connected subgraphs that contains a nodes, as in our case.

2 Preliminaries

For a finite set V, we denote by $\text{edges}(V) = \{\{u, v\} \subseteq V \mid u \neq v\}$ all subsets of V of size two. Sometimes, we consider a function f as a relation and write $f' \subseteq f$ when f' is a (partial) function resulting to take a subset of the order pairs from f. In the sequel, all logarithms are in base 2 unless it is stated differently.

Undirected graphs. We consider finite undirected graphs of the form $G = (V, E)$ where V is a finite non-empty set and $E \subseteq \text{edges}(V)$. Given a graph G, we will denote by $V(G)$ and $E(G)$ the set of vertices and edges, respectively. We will usually use u and v for denoting vertices and e and f for edges. Furthermore, we will use edges as sets and write $v \in e$ when e is an edge incident to v. We denote by $N(v, G) = \{u \mid \{u, v\} \in E(G)\}$ the neighborhood of v in G. We say that a graph G' is a subgraph of G, denoted $G' \subseteq G$, if $V(G') \subseteq V(G)$ and $E(G') \subseteq E(G)$. If two graphs G_1 and G_2 are isomorphic, we write $G_1 \cong G_2$. Furthermore, we write $G_1, v_1 \cong G_2, v_2$ for $v_1 \in V(G_1)$ and $v_2 \in V(G_2)$ if $G_1 \cong G_2$ and v_1 is equivalent to v_2 under the bijective function between G_1 and G_2.

Multigraphs. We also work with graphs with multiple edges between vertices, called multigraphs. A multigraph M is a triple $M = (V, E, r)$ such that V is a finite non-empty set, E is a finite set, and $r : E \to \text{edges}(V)$ (i.e. the edge-assignment function). Intuitively, E is a set of identifiers for edges and r assigns identifiers to edges (i.e. there could be multiple edges between two pair of vertices). Similar than for graphs, we denote by $V(M)$, $E(M)$, and $r(M)$ the corresponding set of vertices, edges, and edge-assignment of M, respectively. We say that a multigraph M' is a sub-multigraph of M, denoted by $M' \subseteq M$, if $V(M') \subseteq V(M)$, $E(M') \subseteq E(M)$, and $r(M') \subseteq r(M)$.

Note that a simple graph is a multigraph M where $r(M)$ is an injective function. For this reason, in the future we will not make distinction between graphs and multigraphs. Furthermore, we will usually work with graphs but all definitions and results also extend to multigraphs. When this is not the case, we will make the distinction explicitly.

Connected graphs. A path in a graph G is a sequence of nodes $\pi = v_0, \ldots, v_n$ such that $\{v_i, v_{i+1}\} \in E(G)$ for every $i < n$ and we say that the length of π is n. Note that v_0 is the trivial path from v_0 to itself of length 0. We say that G is connected if there exists a path between any pair of vertices. Furthermore, we say that $G' \subseteq G$ is a connected component of G if G' is connected and its maximal element over all subgraphs of G under $\subseteq$. We denote by $\text{ConnComp}(G)$ the set of all connected components of G. For $u, v \in V(G)$ we say that u is at distance d of v if there exists a path from u to v of length d and every path from u to v is of length at least d. We denote the distance d from u and v in G by $\text{dist}_G(u,v)$. Given this distance, the diameter of G is defined as $\max_{u,v \in V(G)} \text{dist}_G(u,v)$.

Families. We consider several families of graphs through the paper to give examples or show some properties of our centrality measures. Given a vertex v, we denote by G_v the graph with one vertex v (i.e. $V(G_v) = \{v\}$) and no-edges (i.e. $E(G_v) = \varnothing$). Given an edge $e = \{u, v\}$, we denote by G_e the graph only containing e (i.e. $V(G_e) = \{u, v\}$ and $E(G_e) = \{e\}$). For any $n \geq 1$, we write S_n for the star with $n+1$ vertices such that $V(S_n) = \{0, 1, \ldots, n\}$ and all nodes are connected to 0, namely, $E(S_n) = \{\{0, i\} \mid 0 < i \leq n\}$. Similarly, we write L_n for the line with n vertices where $V(L_n) = \{0, \ldots, n-1\}$ and $E(L_n) = \{\{i, i+1\} \mid 0 \leq i < n-1\}$. The circuit with n vertices is denoted by C_n with $V(C_n) = \{0, \ldots, n-1\}$ and $E(C_n) = \{\{i, (i+1) \bmod n\} \mid 0 \leq i \leq n-1\}$. Finally, the clique of size n is denoted by K_n where $V(K_n) = \{0, \ldots, n-1\}$ and $E(K_n) = \text{edges}(V(K_n))$.

Operations. Through the paper, we use several operations to create, modify, or combine graphs. Given $v \in V(G)$, we denote by $G - v$ the result of removing v from G and all its incident edges, namely, $V(G - v) = V(G) \setminus \{v\}$ and $E(G - v) = \{e \in E(G) \mid v \notin e\}$. Given $e = \{u, v\}$, we write $G + e$ for the result of adding e into G, formally, $V(G + e) = V(G) \cup e$ and $E(G + e) = E(G) \cup \{e\}$ (i.e. if u or v are not in G, then they are included as new vertices). Instead, we write $G - e$ for the result of removing all edges between u and v, namely, $V(G - e) = V(G)$ and $E(G - e) = E(G) \setminus \{e\}$. Note that if G is a (simple) graph, then at most one edge is removed, but if G is a multigraph then all edges between u and v are removed. For G_1 and G_2, we denote by $G_1 \cup G_2$ the union of the two graphs, namely, $V(G_1 \cup G_2) = V(G_1) \cup V(G_2)$ and $E(G_1 \cup G_2) = E(G_1) \cup E(G_2)$. In particular, $G + e = G \cup G_e$.

Let G be a graph and $U \subseteq V(G)$. We define the *set contraction* of U on G as the multigraph G/U by merging the vertices U to one vertex (called U) and keeping multi-edges into U. Formally, G/U is the multigraph M such that $V(M) = (V(G) \setminus U) \cup \{U\}$, $E(M) = \{e \in E(G) \mid e \nsubseteq U\}$ and for every $e \in E(M)$ either $r(M)(e) = e$ whenever $e \cap U = \varnothing$, or $r(M)(e) = \{v, U\}$ whenever $e = \{v, u\}$ with $v \notin U$ and $u \in U$. Note that we use U (i.e. the set) as the new vertex that represent the contraction in M/U. When G is a multigraph, the set contraction G/U easily follows from the above definition.

3 The all-subgraphs centrality

We start by introducing our first centrality measure based on all subgraphs, called the all-subgraphs centrality. In the next section, we generalize this idea to any family of subgraphs.

Fix a graph G and a vertex $v \in V(G)$. We denote by $\mathcal{A}(v,G)$ the set of all connected subgraphs of G that contains v, formally, $\mathcal{A}(v) = \{G' \subseteq G \mid G' \text{ is connected} \wedge v \in V(G')\}$. The *all-subgraphs centrality* of v in G is defined as:

$$C_{\mathcal{A}}(v,G) \;:=\; \log\big(|\mathcal{A}(v,G)|\big)$$

namely, the logarithm of the number of connected subgraphs of G that contains v. Intuitively, the all-subgraphs centrality of a node only considers connected graphs since it captures the importance of the node in the neighborhood that it belongs. We add more importance to a node if its neighborhood is richer in substructures. Furthermore, we consider connected subgraphs since there is no argument to say that a node has more centrality by counting another component that is not directly connected to it.

The function $C_{\mathcal{A}}$ naturally induces a ranking between nodes: the higher the centrality $C_{\mathcal{A}}(v,G)$, the more important is v in G. We define the ranking $<_{\mathcal{A}}$ over $V(G)$ induced by $C_{\mathcal{A}}$ (or just $\mathcal{A}$-ranking for short) such that $u <_{\mathcal{A}} v$ if, and only if, $C_{\mathcal{A}}(u,G) > C_{\mathcal{A}}(v,G)$. Strictly speaking, $<_{\mathcal{A}}$ is not an order in $V(G)$, given that there could exist vertices u and v such that $C_{\mathcal{A}}(u,G) = C_{\mathcal{A}}(v,G)$ (e.g. u and v are isomorphic in G). In this case, we write $u =_{\mathcal{A}} v$.

▶ **Example 1.** Let v be a vertex. Recall that G_v is the trivial graph with one vertex v and no edges. Then one can easily check that $C_{\mathcal{A}}(v,G_v) = 0$ given that $\mathcal{A}(v,G_v) = \{G_v\}$ and then $\log(|\mathcal{A}(v,G_v)|) = \log(1) = 0$. Note that this is the only vertex and graph (up to isomorphism) where the centrality is equal to 0. This follows the intuition that an isolated vertex must have 0 centrality since no one is connected to him.

▶ **Example 2.** Recall that S_n denotes the star graph with $n+1$ vertices. Note that every connected subgraph of S_n corresponds to a subset of $E(S_n)$, and there are 2^n subsets of $E(S_n)$. Therefore, the centrality of the center of the star (i.e. the 0 vertex) is $C_{\mathcal{A}}(0,S_n) = n$. Interestingly, the all-subgraphs centrality of the center of a star coincides with its degree-centrality [27], following the intuition of what should be the centrality in this case. One can easily show that, for any $i \neq 0$, $C_{\mathcal{A}}(i,S_n) = n - 1 + \epsilon$ with $\epsilon \in o(1)$. Thus, in terms of ranking we have that $0 <_{\mathcal{A}} i$ and $i =_{\mathcal{A}} j$ for every $i,j > 0$.

The all-subgraphs centrality is measuring the worst-case entropy [13,26] of the set $\mathcal{A}(v,G)$, namely, the minimum number of bits that are required to represent the set $\mathcal{A}(v,G)$ with bit-codes. Of course, using the size of $|\mathcal{A}(v,G)|$ will give the same ranking of centrality over the vertex of G. Nevertheless, the log-function gives a better interpretation of the centrality in terms of information theory. Moreover, it normalizes the value $|\mathcal{A}(v,G)|$ in a scale that is in correspondence with the intuition of a centrality notion, e.g. Examples 1 and 2 above.

The next lemma is another result that validates the use of worst-case entropy and it will be useful for computing the all-subgraphs centrality over simple graphs. Recall that a vertex $v \in V(G)$ is a *cut vertex* of G if $|\mathrm{ConnComp}(G-v)| < |\mathrm{ConnComp}(G)|$, namely, whose removal increases the number of connected components of G.

▶ **Lemma 3.** *Let v be a cut-vertex of graph G and $G_1,\ldots,G_n$ are all the subgraphs that partition G and whose pairwise intersection is v, that is, $V(G) = \cup_{i=1}^{n} V(G_i)$, $E(G) = \cup_{i=1}^{n} E(G_i)$, and $V(G_i) \cap V(G_j) = \{v\}$ for $i \neq j$. Then*

$$C_{\mathcal{A}}(v,G) = \sum_{i=1}^{n} C_{\mathcal{A}}(v,G_i).$$

Namely, the centrality of v in G is the sum of its centrality in all the components G_i.

This property is usually known in the literature as cut-vertex additivity [31]. Since not every centrality measure satisfies it, this can be seen as the first distinction between all-subgraphs centrality and commonly used centrality measures (e.g. pagerank, betweenness).

▶ **Example 4.** Let G be any graph, $u \in V(G)$, and v be a new vertex not in G. For $e = \{u, v\}$, recall that G_e is the graph only containing e. Then one can easily see that $C_{\mathcal{A}}(u, G_e) = 1$. Since $G + e = G \cup G_e$ and u is a cut-vertex of $G + e$, by Lemma 3 we get:

$$C_{\mathcal{A}}(u, G + e) \;=\; C_{\mathcal{A}}(u, G) + C_{\mathcal{A}}(u, G_e) \;=\; C_{\mathcal{A}}(u, G) + 1$$

Thus, by connecting one new vertex directly to u its centrality grows exactly in one unit. This property is very appealing for a centrality measure and follows verbatim the intuition of the score-monotonicity axiom in [7] (see Section 5 for more discussion). On the other hand, one can check that the new vertex v in $G + e$ absorbs part of the centrality of u in G. Specifically, one can easily see that $|\mathcal{A}(v, G + e)| = |\mathcal{A}(u, G)| + 1$ and then $C_{\mathcal{A}}(v, G + e) = \log(|\mathcal{A}(u, G)| + 1) = C_{\mathcal{A}}(u, G) + \epsilon$, where ϵ is a negligible factor.

▶ **Example 5.** For $n \geq 1$, recall that L_n is the line with n nodes starting from 0 and ending in $n - 1$. For the 0-vertex in L_n there are n-different subgraphs, one for each vertex, and then $C_{\mathcal{A}}(0, L_n) = \log(n)$. The line graph is the most sparse graph with n vertices and 0 is the most extreme vertex in the graph. As one could expect, the centrality of 0 grows very slow, logarithmic in the number of vertex.

For the i-vertex in L_n, we can easily compute its centrality by using Lemma 3. Indeed, the centrality for i is the composition of two lines with $i + 1$ and $n - i$ vertices, respectively. Therefore, by Lemma 3:

$$C_{\mathcal{A}}(i, L_n) \;=\; C_{\mathcal{A}}(0, L_{i+1}) + C_{\mathcal{A}}(0, L_{n-i}) \;=\; \log(i + 1) + \log(n - i).$$

If n is odd, the vertex with maximum centrality is reached by the middle node $\frac{n-1}{2}$ and $C_{\mathcal{A}}(\frac{n-1}{2}, L_n) = 2(\log(n + 1) - 1)$. Thus, the middle point of a line doubles the centrality of the extreme vertices, nevertheless, the grow of its centrality is still logarithmic in n. Finally, note that the centrality is maximized in the middle node and the ranking decreases towards the extremes (i.e. $i <_{\mathcal{A}} i + 1$ for every $i < \frac{n-1}{2}$).

A natural question at this point is to think in lower and upper bounds of the centrality with respect to the number of edges of a graph. Indeed, the number of subgraphs $\mathcal{A}(v, G)$ could be exponential in G but its entropy is bounded by the number of edges as follows.

▶ **Proposition 6.** *For any connected graph G and $v \in V$, it holds that:*

$$\log(|E(G)| + 1) \;\leq\; C_{\mathcal{A}}(v, G) \;\leq\; |E(G)|.$$

From Example 2 above, we can infer that the upper bound is reached by the central vertex of a star. This follows the intuition that the central vertex of a star must be the most central vertex regarding the number of edges (i.e. all edges are pointing to him). Furthermore, in Example 5 we show that the extreme vertex of a line L_n has centrality $\log(n) = \log(|E| + 1)$. That is, the minimum centrality is reached in the extreme points of a line, agreeing with the intuition that the line graph is the most sparsest graph over all undirected graphs.

4 A family of centralities based on subgraphs

The idea of measuring the centrality of a vertex based on relevant substructures is not new [15, 17]. For example, the degree centrality counts how many edges are incident to a vertex and the betweenness centrality [17] counts how many geodesic paths passed through a vertex. In our case, all-subgraphs centrality measures all connected subgraphs including v, but maybe for an expert not all subgraphs are equally important and he will be interested in counting some of them. In this section we generalize the notion of all-subgraphs centrality to propose a framework of centrality notions based on measuring the worst-case entropy of relevant substructures surrounding a vertex.

A family of substructures is a function $\mathcal{F}$ that, given a graph G and a vertex $v \in V(G)$, it assigns a non-empty subset of connected subgraphs in G that contains v. Formally, $\mathcal{F}$ is a function such that $\mathcal{F}(v,G) \subseteq \mathcal{A}(v,G)$ and $\mathcal{F}(v,G) \neq \varnothing$. We also assume that $\mathcal{F}$ is closed under isomorphism, namely, if $G_1, v_1 \cong G_2, v_2$ then $\mathcal{F}(v_1, G_1)$ is isomorphic to $\mathcal{F}(v_2, G_2)$, by extending the isomorphism between G_1, v_1 and G_2, v_2 to subgraphs. For example, $\mathcal{A}$ is a family of substructures where $\mathcal{A}(v,G)$ contains all connected subgraphs in G containing v and is closed under isomorphism. Given a family of substructures we define the $\mathcal{F}$-subgraph centrality (denoted by $C_{\mathcal{F}}(v,G)$) as:

$$C_{\mathcal{F}}(v,G) \;\; := \;\; \log\left(|\mathcal{F}(v,G)|\right)$$

for any graph G and vertex $v \in V(G)$. In other words, following the idea of all-subgraphs centrality it measures the worst-case entropy of the substructures $\mathcal{F}(v,G)$. We could have left the framework open to any monotone positive function over $\mathcal{F}(v,G)$ instead of the logarithm, leading to the same ranking of centrality between vertices. Of course, this will derive in a more complex and enriched theory, however, for the purpose of this paper we will keep the simplicity of the logarithm as it still give place to novel results.

Note that $\mathcal{F}(v,G)$ is non-empty and, therefore, $C_{\mathcal{F}}(v,G)$ is always well-defined. Similar to all-subgraphs centrality, the centrality measures induced a ranking between nodes: we define the $\mathcal{F}$-ranking $<_{\mathcal{F}}$ over $V(G)$ such that $u <_{\mathcal{F}} v$ if, and only if, $C_{\mathcal{F}}(u,G) < C_{\mathcal{F}}(v,G)$.

▶ **Example 7.** Given a graph G and $v \in V(G)$, denote by $\mathcal{T}(v,G)$ all subgraphs $T \in \mathcal{A}(v,G)$ such that T is a tree. Note that an isolated vertex is defined as a trivial tree, so $\mathcal{T}(v,G)$ is always non-empty. Furthermore, the family $\mathcal{T}$ is closed under isomorphism. Then $C_{\mathcal{T}}$ measures the centrality of a vertex based on trees and we call it *the trees centrality*. For example, if L_n is a line graph with n vertices (see Example 5) then we have that $C_{\mathcal{A}}(v,G) = C_{\mathcal{T}}(v,G)$. Indeed, if T is a tree, then $C_{\mathcal{A}}(v,G) = C_{\mathcal{T}}(v,G)$ for every $v \in V(G)$. However, this is not always the case if G has cycles and one can find examples where the two measures give different values and ranking (see the appendix for some examples).

The motivation behind trees centrality is to considered substructures defined by acyclic graphs like trees or paths. For example, path queries [1] are at the core of graph queries languages and they are used to find path substructures between pair of nodes. Also, basic graph patterns that are acyclic (e.g. tree-shaped queries) forms a well-behaved core of graph query languages that can be evaluated efficiently [19]. Therefore, if the query languages mostly uses queries that are acyclic, maybe it makes sense to rank the results by a centrality notion based on trees.

The generalization of all-subgraphs centrality to any family of subgraphs opens the possibilities of defining any centrality notion based on a particular group of relevant subgraphs. In the next section, we use this framework to understand which properties in the family leads to desirable properties in the corresponding centrality measure. This will help to guide the design of a centrality notion based on subgraphs and, moreover, to have a better understanding of this framework and all-subgraphs centrality.

5 What families of subgraphs define good centrality measures?

Several attempts have been taken to define which properties a centrality measure should satisfy and how to axiomatize them [3, 30, 31]. In our framework, each family of subgraphs defines a new centrality measure, so it is not our purpose here to axiomatize them. In some sense, each family of subgraphs captures the know-how of an expert who knows what are the relevant subpatterns around a vertex. From this point of view, it does not make sense to prefer one notion of centrality over the other. Instead, we study here which properties over the family of subgraphs lead to desirable properties on the corresponding centrality notion. We hope that these properties will guide experts on the design of a centrality based on subgraphs and they will help to understand the benefits and problems of choosing one family over the other. Towards this goal, we consider several axioms of centrality that has been proposed in the literature and study which natural property on the family of subgraphs is enough to satisfy it. We also give several examples for showing what happens when a property is not satisfied.

In the sequel, a centrality measure is any function C that given a graph G and $v \in V(G)$, it outputs a non-negative value, i.e., $C(v, G) \geq 0$.

Default axioms. We start our discussion by showing three natural axioms proposed in the literature that any $C_{\mathcal{F}}$ satisfies, for any family of substructures $\mathcal{F}$. We discuss these three axioms briefly and show that they are naturally satisfied by definition.

In [30] they present the so-called *locality axiom*, which says that the centrality of a vertex should only depend on the connected component it belongs. In other words, after removing components that are not connected to a vertex v the centrality of v should not change. This natural axiom is satisfied by any centrality measure based on subgraphs because we define a family of substructures as a subset of connected subgraphs. This might be seen as an irrelevant detail but it is an important design decision of our approach. In second place, an axiom called *anonymity* is introduced in [29]. This is the same as saying that a centrality measure is closed under isomorphism. In our definition we explicitly say that any feasible substructure must be closed under isomorphism, which means that the centrality measure as defined will satisfy this axiom. Finally, in [18] the authors propose a minimum value for any centrality. More specifically, the centrality of an isolated vertex is the minimum possible and it should be 0. These two properties are called isolated minimization and isolated zero, respectively. In our case, these axioms are satisfied by definition, because the set of substructures associated to a vertex must be always non-empty, which means that $|\mathcal{F}(v, G_v)| = 1$ for any family of substructures. Therefore, the minimum possible value for any centrality measure defined in this way is 0.

Monotonicity. The monotonicity axiom is probably the property that more people [7, 29, 30] agree that any centrality notion should satisfy. In [7], the definition of this axiom says that if an edge is added to the graph, then the centrality of the vertex that is incident with the new edge should not decrease. Clearly, a vertex is more central the more edges it has and, thus, a new edge should help to increase its relevance in the graph. A more general definition of this axiom was introduced in [29] where the effect of adding any new edge in the graph should not decrease the centrality of every vertex.

▶ **Axiom 1** (Monotonicity). *A centrality measure C satisfies the monotonicity axiom if for every graph G, $v \in V(G)$ and $e \notin E(G)$, it holds that $C(v, G) \leq C(v, G + e)$.*

Note that the axiom implies that if G_1 is a subgraph of G_2 and $v \in V(G_1)$, then $C(v, G_1) \leq C(v, G_2)$. This coincides with the intuition that v in G_2 has the same or more connections than in G_1 and, thus, its relevance in G_2 should be at least the one in G_1.

What property should a family of subgraphs $\mathcal{F}$ satisfy in order that $C_{\mathcal{F}}$ satisfy Axiom 1? Intuitively, when edge e is added to G we have $G \subseteq G + e$ and all subgraphs that are relevant for v in G should also be relevant for v in $G + e$. Moreover, if a subgraph S is relevant for v in $G + e$ but S is a subgraph of v in G, then it should also be a relevant subgraph of v in G. That is, all subgraphs of G that are relevant should also be relevant in $G + e$ and vice versa We call this the containment property.

▶ **Property 1** (Containment). *A family of subgraphs $\mathcal{F}$ satisfies the containment property if for every graphs G_1 and G_2 such that $G_1 \subseteq G_2$ and for every $v \in V(G_1)$ and $S \in \mathcal{A}(v, G_1)$, it holds that $S \in \mathcal{F}(v, G_1)$ if, and only if, $S \in \mathcal{F}(v, G_2)$.*

In particular, the containment property implies that $\mathcal{F}(v, G_1) \subseteq \mathcal{F}(v, G_2)$ whenever $G_1 \subseteq G_2$. As one could expect, the containment property is enough to satisfy the monotonicity axiom.

▶ **Theorem 8.** *If a family of subgraphs $\mathcal{F}$ satisfies the containment property, then the corresponding centrality measure $C_{\mathcal{F}}$ satisfies the monotonicity axiom.*

One can easily see that the family of all-subgraphs and trees satisfies the containment property and, therefore, the all-subgraphs centrality and trees centrality satisfy monotonicity as expected. Next, we show that this is not always the case.

▶ **Example 9.** Given a graph G and $v \in V(G)$, denote by $\mathcal{W}(v, G)$ all subgraphs $P \in \mathcal{A}(v, G)$ such that $P = v_0, \ldots, v_n$ is a geodesic path in G, namely, it is a path of minimal distance between v_0 and v_n. We assume here that the isolated vertex v is the only geodesic path from v to v. In [9], $|\mathcal{W}(v, G)|$ is defined as the stress centrality of vertex v. Then we define log-stress centrality of v in G as $C_{\mathcal{W}}(v, G)$. Of course, $C_{\mathcal{W}}(v, G)$ is not equivalent to Betweenness(v, G) as a value and in how we aggregate the number of geodesic paths. Nevertheless, it will be useful below to understand Betweenness in the context of counting subgraphs.

One can easily show that the family $\mathcal{W}$ does not satisfy the containment condition. Consider just a line L_3 = . Then if we connect the black vertices and make a triangle K_3 = , then the geodesic path is in $\mathcal{W}(1, L_3)$ but is not in $\mathcal{W}(1, K_3)$. Coincidentally, log-stress centrality (and betweenness centrality as well) do not satisfy the monotonicity axiom. Actually, one can show pathological examples where monotonicity does not hold [16]. For example, if one compares the circuit C_n with the clique K_n one can see that $C_n << K_n$ but $C_{\mathcal{W}}(0, C_n) > C_{\mathcal{W}}(0, K_n)$, and Betweenness$(0, C_n) >$ Betweenness$(0, K_n)$ as well.

It is important to note that, for some axiomatic approaches [30], it is desirable that the center of a star S_{n-1} is the most central node in a graph with n vertices, namely, a centrality measure C satisfies this axiom if, for any n and for any graph G with $|V(G)| = n$, it holds that $C(v, G) \leq C(0, S_{n-1})$ for every $v \in V(G)$. Unfortunately, this assumption contradicts the idea behind the monotonicity axiom, since we can add edges to S_{n-1} but the centrality of the center will never increase. Thus, given that this axiom contradicts the monotonicity axiom, we do not consider it in our analysis.

Rank monotonicity. Another axiom that has been remarked as important in the literature is rank monotonicity [6,7,12,29]. Similar than for monotonicity, this axiom says that if v is more central than u in G, then when we add a new edge e to v the ranking between u and v is preserved. In particular, if v is the most central vertex in G, then it will be the most central vertex in $G + e$ as well. We generalize this intuition as follows.

▶ **Axiom 2** (Rank monotonicity). *A centrality measure C satisfies the rank monotonicity axiom if for every graph G, $u, v \in V(G)$ and $e \notin E(G)$ with $v \in e$, then $C(u, G) \leq C(v, G)$ implies that $C(u, G + e) \leq C(v, G + e)$.*

Note that with $e = \{u, v\}$ it could happen that the increment in centrality for u is bigger than the increment on v, but the axiom says that the centrality of v will be still bigger than the centrality of u. In other words, if I meet Donald Trump, my centrality will rise more than his centrality, however, Donald Trump will still be the president of US.

It is important to say that in [7] an axiom called *density axiom* was proposed, which is a special case of rank monotonicity. Specifically, take a clique K_n, a circuit C_n, and vertices $u \in V(K_n)$ and $v \in V(C_n)$. Then the density axiom says that if we connect u and v with an edge $e = \{u, v\}$, then $G = (K_n \cup C_n) + e$ satisfies $C(u, G) > C(v, G)$ for a centrality measure C. Intuitively, given that the neighborhood of u is more dense that in v, then its centrality should be bigger. One can see that if C satisfies monotonicity (i.e. vertices in K_n has more centrality than in C_n), then rank monotonicity implies the density axiom [7]. Therefore, we can see rank monotonicity as a generalization of the density axiom in [7].

The containment property is useful to imply rank monotonicity but it is not enough. One can easily find centrality measures that satisfies Axiom 1 but it does not satisfy Axiom 2 (see Example 11 below). For this, one needs a notion of "fairness" in the family of subgraphs. Intuitively, if S is a relevant subgraph for v in G and S contains a vertex u, then S should also be relevant for u in S.

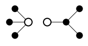
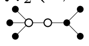
▶ **Property 2** (Fairness). *A family of subgraphs $\mathcal{F}$ satisfies the fairness property if for every graph G, $u, v \in V(G)$ and $S \subseteq G$ with $u, v \in V(S)$ it holds that $S \in \mathcal{F}(u, G)$ iff $S \in \mathcal{F}(v, G)$.*

As we show next, fairness is what you need if you want to preserve the ranking between vertices in a graph.

▶ **Theorem 10.** *If a family of subgraphs $\mathcal{F}$ satisfies the containment property and fairness, then the corresponding centrality measure $C_{\mathcal{F}}$ satisfies the rank monotonicity axiom.*

The family of all-subgraphs, trees and even betweenness (i.e. geodesic paths) satisfy fairness. Given that all-subgraphs and trees also satisfy the containment property, we conclude that both satisfy the rank monotonicity axiom. Next we show a natural family that satisfy the containment property but does not satisfy fairness.

▶ **Example 11.** A natural approach to define a family of subgraphs is to consider subpatterns on a neighborhood of bounded size around a vertex. Intuitively, an expert would not care if a vertex v can reach a far vertex u as long as there are many other substructures close to v. To formalize this, let $k \geq 1$. For a graph G, fix a vertex v and let N_k be the induced subgraph of all vertices at distance at most k of v, i.e., $V(N_k) = \{u \in V(G) \mid \text{dist}_G(u, v) \leq k\}$ and $E(N_k) = \{e \in E(G) \mid e \subseteq V(N_k)\}$. We define the family of subgraphs $\mathcal{N}_k$ such that $\mathcal{N}_k(v, G) = \mathcal{A}(v, N_k)$, that is, all subgraphs in the neighborhood of v with radius k. Then we define the k-neighborhood centrality of v on G as $C_{\mathcal{N}_k}(v, G)$. Note that if the diameter of the graph is less than k then $\mathcal{N}_k(v, G)$ and $\mathcal{A}(v, G)$ coincide.

The family of k-neighborhood satisfies monotonicity but it does not satisfy fairness. Moreover, it does not satisfy the rank monotonicity axiom. To see this, consider the family $\mathcal{N}_2$ and G_1 = . By counting, one can check that the left white vertex, called u, and the right white vertex, called v, satisfy $|\mathcal{N}_2(u, G_1)| = 8$ and $|\mathcal{N}_2(v, G_1)| = 5$, respectively. Then $C_{\mathcal{N}_2}(u, G_1) > C_{\mathcal{N}_2}(v, G_1)$. However, if we add an edge e between the two and create the graph $G_1 + e$ = , then one can check that $\mathcal{N}_2$ does not satisfy fairness. For instance, the whole graph $G_1 + e \in \mathcal{N}_2(v, G_1 + e)$, contains u and v, but $G_1 + e \notin \mathcal{N}_2(u, G_1 + e)$. One can also check by counting that $|\mathcal{N}_2(u, G_1 + e)| = 24$ and $|\mathcal{N}_2(v, G_1 + e)| = 45$. Thus, $C_{\mathcal{N}_2}(u, G_1 + e) < C_{\mathcal{N}_2}(v, G_1 + e)$ and 2-neighborhood does not satisfy the rank monotonicity axiom as well.

The previous example shows that, if we want to approximate all-subgraphs centrality by only counting subgraphs up to a certain radius, one will have to loose some natural properties, like rank monotonicity.

Line minimization. Everyone would agree that any reasonable notion for centrality should assign 0 centrality to an isolated vertex [3, 30]. Basically, there is nothing less central to a community than the vertex that is not connected to any other vertex. One can generalize this idea by considering, what is the most sparse connected graph with n vertices. Clearly, the line L_n should be this graph: it is the only graph with n vertices that maximizes the diameter. Then the vertices that minimize the centrality in the line L_n are its extreme points 0 and $n-1$ and one would expect that this should be the vertices that have less centrality over all connected graphs with n-vertices. We call this axiom line minimization

▶ **Axiom 3** (Line minimization). *A centrality measure C satisfies the line minimization axiom if for every n and every connected graph G with $|V(G)| = n$ it holds that $C(0, L_n) \leq C(v, G)$ for every $v \in V(G)$.*

All centralities that we consider in this paper satisfy the line minimization axiom. Of course, one can manage to find unnatural families of subgraphs that produce centrality measures not satisfying this axiom. Still, one would like to find under which circumstances a centrality measure defined from a family of subgraphs satisfies it. For this, we need to introduce the following property.

▶ **Property 3** (Inclusion). *A family of subgraphs $\mathcal{F}$ satisfies the inclusion property if for every graph G, $v \in V(G)$, and $S \in \mathcal{F}(v, G)$, if $S' \subseteq S$ and $v \in V(S')$, then $S' \in \mathcal{F}(v, G)$.*

Intuitively, this property is saying that every subgraph of a relevant subgraph should also be relevant for the family. Actually, this property is satisfied by all families of subgraphs proposed so far.

▶ **Theorem 12.** *If a family of subgraphs $\mathcal{F}$ satisfies the containment and inclusion properties, then the corresponding centrality measure $C_{\mathcal{F}}$ satisfies the line minimization axiom.*

Continuity. The inclusion property plus the containment property actually imply a natural property over centrality measures defined by family of subgraphs. Given that all subgraphs of a relevant subgraph are also included, it gives a sense of "continuity" in the centrality notion. Specifically, each time that we add a set of edges that rises the centrality of a vertex, there exists a way to add them, one at a time, in such a way that the centrality of the vertex always increases. We formalize this intuition as follows.

▶ **Axiom 4** (Continuity). *A centrality measure C satisfies the continuity axiom if for every graphs G and F, and $v \in V(G)$, if $C(v,G) < C(v,G \cup F)$, then there exists edges $e_1,\dots,e_k \in E(F)$ such that: $C(v,G) < C(v,G+e_1) < \dots < C(v,G+e_1+\dots+e_k) = C(v,G \cup F)$.*

To the best of our knowledge, the continuity axiom has not been proposed before in the literature. Furthermore, the inclusion and containment property implies the continuity axiom over the corresponding centrality measure.

▶ **Theorem 13.** *If a family of subgraphs $\mathcal{F}$ satisfies the inclusion and containment properties, then the corresponding centrality measure $C_{\mathcal{F}}$ satisfies the continuity axiom.*

All families of subgraphs so far satisfy the inclusion property and, therefore, their centrality measure satisfy the continuity axiom as well. We give below a centrality measure based on cliques as a counter-example of this theorem.

▶ **Example 14.** Cliques are relevant substructure in network analysis and they are usually used to measure the importance of vertices [27]. In [15], this idea has been taken a step further by counting the number of cliques that a vertex belongs, which is called the cross-clique centrality. We can define this centrality with families of subgraphs as follows. Define the family $\mathcal{K}$ such that $\mathcal{K}(v,G)$ contains all subgraphs $K \in \mathcal{A}(v,G)$ such that K is a clique of size 1 (i.e. v) or size greater than 2 for every graph G and $v \in V(G)$. Then the clique centrality of v on G is defined as $C_{\mathcal{K}}(v,G)$. Note that $C_{\mathcal{K}}(v,G) = \log(\text{Cross-Clique}(v,G)+1)$ and, thus, we can use $C_{\mathcal{K}}$ as a proxy to understand cross-clique centrality.

Cliques $\mathcal{K}$ is a family that does not satisfy the inclusion property. Indeed, any subgraph of a clique is not necessarily a clique. One can also check that its centrality $C_{\mathcal{K}}$ also does not satisfy the continuity property. For example, consider a single edge $G =$ where the white vertex v has clique centrality $C_{\mathcal{K}}(v,G) = 0$. Then, if a triangle $F =$ is added to G, producing the graph $G+F =$ with $C_{\mathcal{K}}(v,G+F) = 1$, there is no way to rise the centrality of v from 0 to 1 by adding the edges of the triangle one-by-one.

Size. The last axiom that we study here is the one proposed in [7] about size. This was formalized as follows: for any $n > 0$ if we consider clique K_n and a circuit C_n, for a centrality measure C one would expect that $C(0,K_n) > C(0,C_n)$. Then no matter how big is $C(0,K_n)$, there should exists a value $m > n$ where the centrality of the cycle C_m passes the centrality of the clique K_n, namely, $C(0,K_n) < C(0,C_m)$. This argument is related to the size of graphs in the sense that no matter how slow the centrality of C_m grows, at some point it should beat the clique of size n. We propose a generalization of this axiom as follows.

▶ **Axiom 5** (Size). *A centrality measure C satisfies the size axiom if for every infinite sequence $\{G_n\}_{0\leq n}$ of connected graphs with $V(G_n) = \{0,\dots,n\}$ and for every value N there exists m such that $C(0,G_m) \geq N$.*

Here the sequence $\{G_n\}_{0\leq n}$ is playing the role of the circuits and N the role of the centrality in the clique. Thus, if a centrality measures satisfies Axiom 5 then it satisfies the size axiom of [7], but the converse of course is not true.

This axiom is clearly satisfied by all-subgraphs and trees centrality. Indeed, by Proposition 6 we know that $C_{\mathcal{A}}(v,G)$ is always bounded below by $\log(n)$ and thus the all-subgraphs satisfy the axiom (similar argument can be given for trees centrality). Typical centrality measures that do not satisfy the size axiom are "local measures" that only consider subgraphs of bounded size, i.e., degree or k-neighborhood centrality. However, there are families of subgraphs of unbounded size that also do not satisfy this axiom, i.e., clique centrality. In both cases, if we consider the sequence of lines $\{L_n\}_{0\leq n}$, we can see that the centrality on the vertex 0 is not growing and, thus, for a reasonable N the axiom does not hold. Actually, the next theorem shows that this counter-example is enough to show whether a centrality measure satisfy the size axiom or not.

▶ **Theorem 15.** *Let $\mathcal{F}$ be a family of subgraphs that satisfies the line minimization axiom. Then $C_{\mathcal{F}}$ satisfies the size axiom if, and only if,* $\lim_{n\to\infty} |\mathcal{F}(0, L_n)| = \infty$.

We remark that all centrality measures consider in this paper satisfies the line minimization axiom. Therefore, it is enough to check whether the family of subgraphs grows on the line to see whether the centrality notion is "local" or not.

We want to end this section by pointing out that in [7] it was shown that all standard notions of centrality in the literature (like closeness [4], betweenness [17], Page Rank [10], Katz index [22], etc) do not satisfy at least one of its axioms and, therefore, do not satisfy at least one of the general axioms stated above. This shows that all the standard notions for centrality studied in the literature are different with all-subgraphs centrality.

6 Extension to group centrality measures

Any natural centrality measure should come with a simple extension to measure the centrality of sets of vertices (also called *group centrality*). Although this is a desirable property, it is not always clear how to do it (i.e. not many centrality measures in the literature have a standard extension to group centrality). In this section, we embark on extending our families of centralities from vertices to sets and give a natural characterization for all-subgraphs group centrality. Towards the end, we show an application of this notion regarding the centrality maximization of a vertex.

Given an arbitrary family of subgraphs $\mathcal{F}$, what should be its extension to groups? A first approach is to consider all connected subgraphs in $\mathcal{F}$ that contains all elements in the group. Formally, given $U \subseteq V(G)$ one could consider the family of relevant subgraphs:

$$\mathcal{F}^*(U, G) \;=\; \{\, S \subseteq G \;\mid\; U \subseteq V(S) \wedge \exists v \in U.\; S \in \mathcal{F}(v, G) \,\}.$$

In other words, all relevant subgraphs of vertices in U that cover U. Although this is the direct extension for connected subgraphs, this definition rises two issues. First, some local families (e.g. k-neighborhood) could not keep the restriction of having all vertices in U inside a subgraph (i.e. $U \subseteq V(S)$). Moreover, if the size of U grows then there will be less subgraphs satisfying such restriction, making the definition impractical for some families of subgraphs. Second, sets that have more relevant subgraphs under this definition are likely to be closer in the graph. For example, if we look at the extension of all-subgraphs $\mathcal{A}^*(U, G)$, then in a circuit C_n a set U of k-vertices that has maximum centrality will be any set of k contiguous vertices. Clearly, if one looks for a central group of k-vertices in C_n, one would prefer a set of k-vertices that are equidistant in C_n because they cover more relevant structures of the graph as a group.

Given the previous discussion, we define the group extension of $\mathcal{F}$ to sets of vertices U on G, denoted as $\mathcal{F}(U, G)$, as follows:

$$\mathcal{F}(U, G) \;=\; \{\, S \subseteq G \;\mid\; U \subseteq V(S) \wedge \forall H \in \mathrm{ConnComp}(S).\; \exists v \in U.\; H \in \mathcal{F}(v, G) \,\}.$$

Note that this extension is similar to the one discussed above (i.e. $\mathcal{F}^*(U, G)$), but we asked that each connected component from S comes from a relevant subgraph of a vertex in U. This allows to use disconnected subgraphs to cover U and, at the same time, each connected component comes from connected subgraphs in $\mathcal{F}$. Unlike our first extension, this definition is not local anymore and gives meaningful results for any set U. In particular, when $U = \{v\}$ this definition generalizes the family of subgraphs for vertices given that $\mathcal{F}(U, G) = \mathcal{F}(v, G)$.

With a family of subgraphs for sets of vertices, it is natural to develop its corresponding group centrality. Similar than for vertices, given a set $U \subseteq V(G)$ from a graph G, we define the $\mathcal{F}$-group centrality measure of U in G as the worst-case entropy of $\mathcal{F}(U, G)$, namely:

$$C_{\mathcal{F}}(U, G) \;=\; \log\left(|\mathcal{F}(U, G)|\right).$$

All families introduced in previous sections have a corresponding group centrality measure. From now, we restrict our analysis to the all-subgraphs family and its centrality over groups, and leave the understanding of other families for future work.

▶ **Example 16.** Let C_n be a circuit of length $n \geq 3$ and consider all sets $U \subseteq V(C_n)$ of two vertices. Then one can check that the set U that maximizes $C_{\mathcal{A}}(U, C_n)$ is any pair of vertices that are at distance $\frac{n}{2}$ (assuming n even). Furthermore, if U are sets of k vertices with k a factor of n, then $C_{\mathcal{A}}(U, C_n)$ is maximized when all vertices in U are distributed in C_n with equal distance. Intuitively, this is the best way of covering a circuit C_n with k vertices.

Next we show that all-subgraphs group centrality over U can be reduced to computing the centrality of a vertex. Recall that we denote by G/U the set contraction of U on G, namely, to merge the vertices U to one vertex and keeping multi-edges into U (see Section 2). In particular, recall that U is a vertex in the multigraph G/U.

▶ **Theorem 17.** *Let G be a graph and $U \subseteq V(G)$. Then:*

$$C_{\mathcal{A}}(U, G) \;=\; C_{\mathcal{A}}(U, G/U) + |\{e \in E(G) \mid e \subseteq U\}|$$

The all-subgraphs group centrality of a set U in G is then reduced to the centrality of U (i.e. as a vertex) in the set-contraction of U on G plus the number of edges between vertices in U. Note that, in particular, this shows that if we look for k-sets of high centrality, then the all-subgraphs centrality is balancing between the number of edges of the set (i.e. how similar is the set to a clique) versus how central it is if we contract it into a vertex.

This connection between both definitions (i.e. vertices and sets) for all-subgraphs centrality is strictly related to the properties of the family. Given two subgraphs G_1 and G_2 of G with $V(G_1) \cap V(G_2) \neq \emptyset$, we can generate a new subgraph $G_1 \cup G_2$ by merging the nodes they share. Unfortunately, this is not possible for all families like the family of trees $\mathcal{T}$, that is, the union of two trees is not necessarily in $\mathcal{T}$. This means that Theorem 17 cannot be directly extended for families like trees, in particular, for trees centrality.

To end this section, we show an example how the all-subgraphs group centrality allows us to study simple questions regarding the maximization of the centrality of a vertex. Given a graph G and a vertex $v \in V(G)$, with whom should we connect v in G in order to maximize its centrality? In other words, if I am in a social network, with whom should I connect in order to maximize my centrality? A naive answer to this question is to connect v to the most central vertex in G. Actually, from the perspective of all-subgraphs centrality this is not the right answer: connecting to the most central node will rise its centrality but maybe the centrality of the most central vertex is highly dependent of v's centrality. Instead, all-subgraphs centrality says that v must be connected to the vertex u where $\{u, v\}$ (as a group) is more central in G.

▶ **Theorem 18.** *Given G and $v \in V(G)$ with $\{u \in V(G) \mid \{u, v\} \notin E(G)\} \neq \emptyset$, it holds that:*

$$\arg\max_{u \in V(G)} C_{\mathcal{A}}(v, G + \{v, u\}) \;=\; \arg\max_{\{u,v\} \notin E(G)} C_{\mathcal{A}}(\{v, u\}, G)$$

7 On computing centrality measures based on subgraphs

We study here the problem of computing centrality measures based on subgraphs. In particular, we study the problem of computing the all-subgraphs centrality. We state the problem as follows: given a family of subgraphs $\mathcal{F}$, consider the problem

Problem:	$\textsc{Count}(\mathcal{F})$
Input:	A graph G and a vertex $v \in V(G)$
Output:	$\lvert\mathcal{F}(v,G)\rvert$

Furthermore, given a class of graphs $\mathcal{G}$ we write $\textsc{Count}(\mathcal{F})[\mathcal{G}]$ for the parametrized version of $\textsc{Count}(\mathcal{F})$ when input graph G is restricted to $\mathcal{G}$. Of course, given a family $\mathcal{F}$ computing its centrality $C_{\mathcal{F}}$ requires also taking the logarithm to the output of $\textsc{Count}(\mathcal{F})$. Although these are not the same problems, the conclusions obtained here sheds light on the pitfalls of computing a centrality based on a family $\mathcal{F}$.

We start by giving an algorithm for computing $\textsc{Count}$ over all-subgraphs $\mathcal{A}$. Algorithm 1 shows a simple recursive algorithm for counting all connected subgraphs that contains a vertex $v \in V(G)$ in a (multi)graph G. The main idea is indeed very simple. Recall that $N(v,G)$ denotes the neighborhood of v in G (see Section 2). If $N(v,G) = \varnothing$, the vertex v is an isolated vertex and there is exactly one subgraph. Otherwise, v is connected to at least one vertex, called it $u \in N(v,G)$, and by some edge $e = \{u,v\}$. Then we can partition the set of connected subgraphs $\mathcal{A}(v,G)$ into those that u and v are directly connected by some edge, and those that are not. For the former, we can compute the exact number recursively as $\textsc{CountAll}(G-e,v)$ (recall here that $G-e$ contains no edges between u and v). For the latter, let $w(e)$ be the number of edges between u and v in G (recall that G could be a multigraph). Then all connected subgraphs where u and v are directly connected by some edge can be formed by choosing a non-empty set of edges between u and v (i.e. $2^{w(e)}-1$ many possibilities) plus a connected subgraph from $\mathcal{A}(e,G/e)$ where G/e is the set contraction of e on G (i.e. $\textsc{CountAll}(G/e,e)$ many possibilities). Therefore, we can compute $\textsc{CountAll}(G,v)$ by recursively computing $\textsc{CountAll}(G-e,v)$ and $\textsc{CountAll}(G/e,e)$. In both cases, the number of edges or the number of vertices is reduced, and $\textsc{CountAll}$ will eventually finish.

Although Algorithm 1 is easy to implement, it could take exponential time in the number of edges. Actually, this is the best that one can hope as we show in the next result. Recall that #P is the class of counting problems that can be defined as counting the number of accepting runs of a polynomial-time non-deterministic Turing machine. Further, a counting problem is #P-complete if it is in #P and all counting problems in #P can be reduced to it [32]. It is known that a polynomial-time algorithm for solving a #P-complete problem, if it existed, would imply that P = NP. For this reason, #P-complete is a class of counting problems considered as hard [2].

▶ **Theorem 19.** $\textsc{Count}(\mathcal{A})$ *and* $\textsc{Count}(\mathcal{T})$ *are* #P*-complete.*

This is a negative result for using all-subgraphs centrality or trees centrality in practice. Nevertheless, we believe that this should not overshadow the impact that both measures can have in defining good centrality notions. As we show in Section 5, both notions behaved well as centrality measures and, although they are difficult to compute, they can still be used, for example, to guide the definition of new centrality measures or to design new efficient algorithms for computing the most relevant vertices in a graph.

Given that computing all-subgraphs over any graph is a difficult problem, our next step is to consider classes of graphs $\mathcal{G}$ where $\textsc{Count}(\mathcal{F})[\mathcal{G}]$ can be solved efficiently. A natural class

Algorithm 1 All-subgraphs counting

```
1: Require: A graph $G$ and vertex $v \in V(G)$
2: procedure CountAll($G$, $v$)
3:     if $N(v,G) = \varnothing$ then
4:         return 1
5:     else
6:         let $u \in N(v,G)$
7:         $e \leftarrow \{u,v\}$
8:         return CountAll($G - e, v$) +
9:                 $(2^{w(e)} - 1) \cdot$ CountAll($G/e, e$)
```

Algorithm 2 All-subgraphs on trees

```
1: Require: A tree $T$ and vertex $v \in V(T)$
2: procedure CountTrees($T$, $v$)
3:     if $N(v,T) = \varnothing$ then
4:         return 1
5:     else
6:         let $u \in N(v,T)$
7:         $e \leftarrow \{u,v\}$
8:         return CountTrees($T - e, v$) $\cdot$
9:                 (CountTrees($T - e, u$) + 1)
```

to start here are trees. Indeed, when G is a tree every internal vertex is a cut-vertex and we can use the ideas of Lemma 3 for computing $|\mathcal{A}(v,G)|$ efficiently. More specific, from the proof of Lemma 3 (see the appendix) one can show that if v is a cut-vertex of a graph G and $G_1,\ldots,G_n$ are subgraphs that partitions G on v (i.e. $V(G) = \cup_{i=1}^{n} V(G_i)$, $E(G) = \cup_{i=1}^{n} E(G_i)$, and $V(G_i) \cap V(G_j) = \{v\}$ for $i \neq j$), then $\mathcal{A}(v,G) = \prod_{i=1}^{n} \mathcal{A}(v,G_i)$. We can exploit this in a tree by considering all subtrees $T_1,\ldots,T_n$ hanging from v and computing $\mathcal{A}(v,G)$ as the product of $\mathcal{A}(v,T_i)$.

In the procedure CountTrees of Algorithm 2 we use the previous idea for computing $|\mathcal{A}(v,T)|$ when T is a tree and $v \in V(T)$. It follows a similar approach to that in Algorithm 1. First, if v is an isolated vertex (i.e. $N(v,T) = \varnothing$), then it outputs 1. Otherwise, it takes a vertex $u \in N(v,T)$, defines the edge $e = \{u,v\}$, and decompose T in two subtrees by removing e from the graph. Notice that, if we remove e from T, we create two connected components T_v and T_u, where T_v and T_u contains v and u, respectively. One can easily check that T_v and $T_u + e$ partitions T on v and we have $|\mathcal{A}(v,T)| = |\mathcal{A}(v,T_v)| \cdot |\mathcal{A}(v,T_u + e)|$ by the previous discussion above. Furthermore, it is straightforward to check that $|\mathcal{A}(v,T_v)| = |\mathcal{A}(v,T-e)|$ and $|\mathcal{A}(v,T_u+e)| = |\mathcal{A}(u,T-e)+1|$. Thus, we can compute $|\mathcal{A}(v,T)|$ by recursively computing CountTrees$(T-e,v)$ multiplied by CountTrees$(T-e,u)+1$.

In contrast to Algorithm 1, the recursion in CountTrees separates the graph in two disjoint subtrees. This implies that the recursion eventually finishes and, moreover, it takes linear time in the size of the tree. Interestingly, we can extend this idea to any graph of bounded tree-width. To formalize the notion of bounded tree-width, we need to introduce some notation. Given a graph G, a tree decomposition T of G is a tree such that $V(T)$ are sets of $V(G)$ (i.e. $X \subseteq V(G)$ for every $X \in V(T)$) and satisfies the following three properties: (1) $V(G) = \bigcup_{X \in V(T)} X$, (2) if $v \in X \cap Y$ for $X, Y \in V(T)$, then $v \in Z$ for all $Z \in V(T)$ in the simple path from X to Y in T, and (3) for every $e \in E(G)$, there exists $X \in V(T)$ such that $e \subseteq X$. The width of a tree decomposition T is equal to $\max_{X \in V(T)} |X| - 1$ and the tree-width $\mathrm{tw}(G)$ of G is the minimum width among all possible tree decompositions of G [28]. A class $\mathcal{G}$ of graphs has bounded tree width if there exists a uniform bound k such that $\mathrm{tw}(G) \leq k$ for every $G \in \mathcal{G}$. For example, all trees is a class that has tree-width bounded by 1.

▶ **Theorem 20.** *If $\mathcal{G}$ has bounded tree-width, then* Count$(\mathcal{A})[\mathcal{G}]$ *can be solved in* PTime.

The previous result shows that the problem becomes tractable when graphs has bounded tree-width. Despite that graphs have high tree-width in practice [25], this result gives some clues on how to tackle the problem of computing the all-subgraphs centrality.

8 Future work

This work arises several research opportunities regarding centrality measures based on subgraphs, which are briefly discussed here. One of the most important question is whether all-subgraphs centrality can be approximated efficiently, or even if the rank order given by this measure can be approximated. Another interesting question is to consider when a family of graphs can approximate another family over some particular class of graphs (e.g. plain graphs). For the sake of simplification, we only considered undirected graphs but another relevant question is to study how to extend these results to directed graphs or to hypergraphs. Furthermore, the initial motivation of our approach came from centrality measures for graph query languages, but in order to incorporate this approach, several properties must be understood like, for example, how to mix the centrality measures to the output of a query. Finally, it would be interesting to consider a randomized version of our approach where not all subgraphs have the same chances to appear. Instead of considering the worst-case entropy, one could study the entropy of a family given a particular distribution and study their properties. We leave this and other questions for future work.

References

1 Renzo Angles, Marcelo Arenas, Pablo Barceló, Aidan Hogan, Juan L. Reutter, and Domagoj Vrgoc. Foundations of modern query languages for graph databases. *ACM Comput. Surv.*, 50(5):68:1–68:40, 2017.

2 Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

3 Sambaran Bandyopadhyay, Ramasuri Narayanam, and M Narasimha Murty. A generic axiomatic characterization for measuring influence in social networks. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 2606–2611. IEEE, 2018.

4 Alex Bavelas. Communication patterns in task-oriented groups. *The Journal of the Acoustical Society of America*, 22(6):725–730, 1950.

5 Hans L Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on computing*, 25(6):1305–1317, 1996.

6 Paolo Boldi, Alessandro Luongo, and Sebastiano Vigna. Rank monotonicity in centrality measures. *Network Science*, 5(4):529–550, 2017.

7 Paolo Boldi and Sebastiano Vigna. Axioms for centrality. *Internet Mathematics*, 10(3-4):222–262, 2014.

8 Stephen P Borgatti and Martin G Everett. A graph-theoretic perspective on centrality. *Social networks*, 28(4):466–484, 2006.

9 Ulrik Brandes. *Network analysis: methodological foundations*, volume 3418. Springer Science & Business Media, 2005.

10 Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117, 1998.

11 Carlos Buil-Aranda, Martın Ugarte, Marcelo Arenas, and Michel Dumontier. A preliminary investigation into sparql query complexity and federation in bio2rdf. In *Alberto Mendelzon International Workshop on Foundations of Data Management*, page 196, 2015.

12 Steve Chien, Cynthia Dwork, Ravi Kumar, Daniel R Simon, and D Sivakumar. Link evolution: Analysis and algorithms. *Internet mathematics*, 1(3):277–304, 2004.

13 Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

14 Ernesto Estrada and Juan A Rodriguez-Velazquez. Subgraph centrality in complex networks. *Physical Review E*, 71(5):056103, 2005.

15 Mohammad Reza Faghani and Uyen Trang Nguyen. A study of xss worm propagation and detection mechanisms in online social networks. *IEEE transactions on information forensics and security*, 8(11):1815–1826, 2013.

16 Linton C Freeman. Centrality in social networks conceptual clarification. *Social Networks*, 1(1978/79):215–239.

17 Linton C Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41, 1977.

18 Manuj Garg. Axiomatic foundations of centrality in networks. *Available at SSRN 1372441*, 2009.

19 Georg Gottlob, Gianluigi Greco, Nicola Leone, and Francesco Scarcello. Hypertree decompositions: Questions and answers. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pages 57–74, 2016.

20 Hawoong Jeong, Sean P Mason, A-L Barabási, and Zoltan N Oltvai. Lethality and centrality in protein networks. *Nature*, 411(6833):41, 2001.

21 Mark Jerrum. Counting trees in a graph is #p-complete. *Information Processing Letters*, 51:111–116, 1994.

22 Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.

23 Harold J Leavitt. Some effects of certain communication patterns on group performance. *The Journal of Abnormal and Social Psychology*, 46(1):38, 1951.

24 Johannes Lorey and Felix Naumann. Detecting sparql query templates for data prefetching. In *Extended Semantic Web Conference*, pages 124–139. Springer, 2013.

25 Silviu Maniu, Pierre Senellart, and Suraj Jog. An experimental study of the treewidth of real-world graph data. In *22nd International Conference on Database Theory, ICDT 2019, March 26-28, 2019, Lisbon, Portugal*, pages 12:1–12:18, 2019.

26 Gonzalo Navarro. *Compact data structures: A practical approach*. Cambridge University Press, 2016.

27 Mark Newman. *Networks: an introduction*. Oxford university press, 2010.

28 Neil Robertson and Paul D Seymour. Graph minors. iii. planar tree-width. *Journal of Combinatorial Theory, Series B*, 36(1):49–64, 1984.

29 Gert Sabidussi. The centrality index of a graph. *Psychometrika*, 31(4):581–603, 1966.

30 Oskar Skibski, Talal Rahwan, Tomasz P Michalak, and Makoto Yokoo. Attachment centrality: An axiomatic approach to connectivity in networks. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pages 168–176. International Foundation for Autonomous Agents and Multiagent Systems, 2016.

31 Oskar Skibski and Jadwiga Sosnowska. Axioms for distance-based centralities. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

32 Leslie G Valiant. The complexity of computing the permanent. *Theoretical computer science*, 8(2):189–201, 1979.

33 Tomasz Wąs and Oskar Skibski. An axiomatization of the eigenvector and katz centralities. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

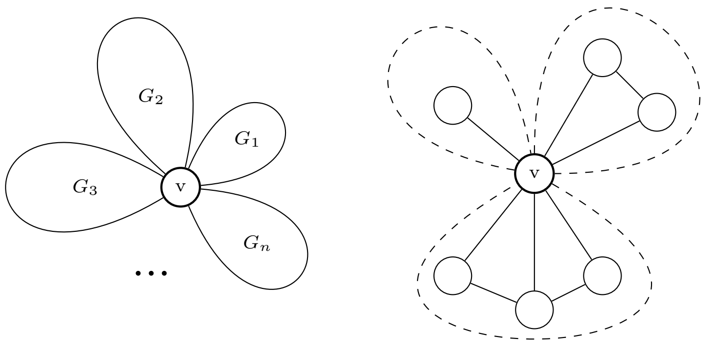


Figure 1 An example of Lemma 3 where v is the cut-vertex and $G_1, \ldots, G_n$ are all the subgraphs that partitions G whose pairwise intersection is v.

A Proofs Section 3

A.1 Proof Lemma 3

The argument is simply combinatorics (see Figure 1 for some intuition). For each subgraph G_i the subgraphs that contains v does not depend on the subgraphs of another G_k. This mean that all the possible subgraphs of G that contains v could be any combination of each independent sub graph. Thus, we have

$$|\mathcal{A}(v,G)| = \prod_{i=1}^{n} |\mathcal{A}(v,G_i)|.$$

Applying logarithm in both sides we obtain the result.

A.2 Proof of Proposition 6

For the lower bound, let G be a connected graph. Then, if x is a fixed node from G, every edge can be reached from x. Therefore we can define an order relation between edges based on the distance from x. This is, given two edges e_1 and e_2,

$$e_1 \leq e_2 \iff d_x(e_1) \leq d_x(e_2).$$

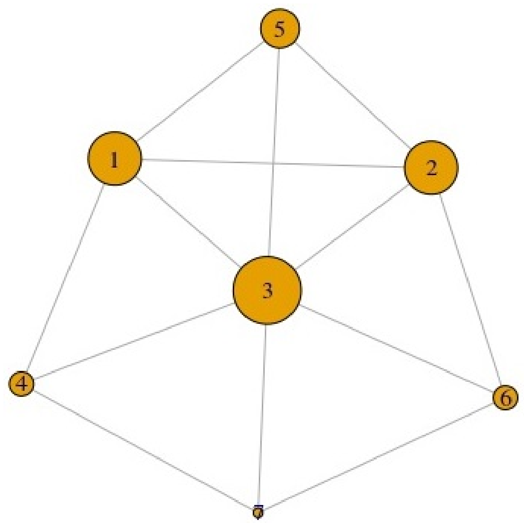
Using this order, we can start removing edges from the furthest nodes from x. The resulting graph is still connected and contains x. Every time an edge is removed we create a new connected subgraph of G that contains x. We have to add the final subgraph with no edges on it. In other words,

$$|E(G)| + 1 \leq |\mathcal{A}(x,G)|.$$

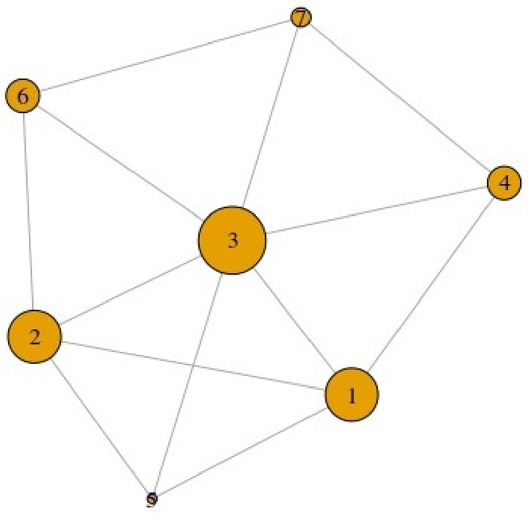
To prove the upper bound, we use the following argument. Whenever an edge is added or removed from the graph G, a new graph is obtained. In this new graph there is always a connected component that contains x. But every time we make this process this component could or could not change. In consequence, this connected component represents the connected subgraph that we need to count for the computation of the centrality of x. Then,

$$|\mathcal{A}(x,G)| \leq 2^{|E(G)|}$$

applying logarithm we conclude the proof.



(a) All-Subgraph centrality ranking.



(b) Trees centrality ranking.

B Proofs Section 4

B.1 Example of all-subgraphs centrality vs trees centrality

It is not trivial to show that exists a graph G and nodes $u, v \in V(G)$ such that $u <_{\mathcal{A}} v$ and $v <_{\mathcal{T}} u$. In Figure 2a, relative sizes of nodes give an intuition for the all-subgraphs centrality rank of every node in G. On the other hand, Figure 2b shows the trees centrality rank for the same graph G. In particular, we can notice that $6 <_{\mathcal{A}} 5$ but $5 <_{\mathcal{T}} 6$.

C Proofs Section 5

C.1 Proof Theorem 8

Let $\mathcal{F}$ be any family satisfying the containment property. It is clear that the $G \subseteq G + e$. Thus, by the containment property $\mathcal{F}(v, G) \subseteq \mathcal{F}(v, G + e)$. Therefore, $|\mathcal{F}(v, G)| \leq |\mathcal{F}(v, G + e)|$ and by monotonicity of logarithm it holds that

$$C_{\mathcal{F}}(v, G) \leq C_{\mathcal{F}}(v, G + e).$$

C.2 Proof Theorem 10

Let G be a graph and $u, v \in V(G)$ and $e \notin E(G)$. If $C_{\mathcal{F}}(u, G) \leq C_{\mathcal{F}}(v, G)$ then $|\mathcal{F}(u, G)| \leq |\mathcal{F}(v, G)|$. Let suppose that $C_{\mathcal{F}}(u, G + e) > C_{\mathcal{F}}(v, G + e)$. We know that $\mathcal{F}$ satisfies the containment property. Then, since $G \subseteq G + e$, for every subgraph $S \subseteq G$, $S \in \mathcal{F}(u, G)$ if and only if $S \in \mathcal{F}(u, G + e)$. The same occurs for v. Thus, the only way this could be true is that exists $S \subseteq G + e$ such that $e \in E(S)$ and $S \in \mathcal{F}(u, G + e)$ but $S \notin \mathcal{F}(v, G + e)$ which directly contradicts the Fairness property.

C.3 Proof Theorem 12

Let $\mathcal{F}$ be a family of substructures satisfying the containment and inclusion property. Let fix $k = \max_{S \in \mathcal{F}(0,n)} |V(S)|$, i.e the size of the biggest substructure in the family for L_n that contains

0. Because of the inclusion property, for every $i \leq k$ we have that $L_i \in \mathcal{F}(0, L_n)$. Therefore, $|\mathcal{F}(0, L_n)| = k$. Now, let G be any graph with n vertices and $v \in V(G)$. We define the radius from v as $d = \max_{u \in V(G)} d_G(v, u)$. If $d \geq k$ it means that exists a simple path π from v to some $u \in V(G)$ where $|\pi| \geq k$. Using the containment property, if $i \leq k$ we know that $L_i \in \mathcal{F}(v, G)$ because $L_k \subseteq G$ and $L_i \in \mathcal{F}(0, L_k)$. [1] On the other hand, if $d < k$, then for every $v \in V(G)$, since G is connected, there exists a simple path from v to u let say π_v. Every path π_v is a connected subgraph starting in v and finishing with u. Then, if $v \neq w$ we have that $\pi_v \neq \pi_w$. We also know that $\pi_v \in \mathcal{F}(u, G)$ because $|\pi_u| \leq d < k$ and, using the same argument as before, there is an isomorphism from π_u to a subgraph $L_{|\pi_u|} \in \mathcal{F}(0, L_n)$. This implies that $|\mathcal{F}(v, G)| \geq n \geq |\mathcal{F}(0, L_n)|$. In both cases we conclude that $|\mathcal{F}(0, L_n)| \leq |\mathcal{F}(v, G)|$. In consequence $C_{\mathcal{F}}$ satisfies the isolated vertex axiom.

C.4 Proof Theorem 13

For a node $v \in V(G)$, let define M as the minimal connected subgraph in the following sense. $\mathcal{F}(v, G \cup M) = \mathcal{F}(v, G \cup F)$ and for any $S \subseteq F$, if $\mathcal{F}(v, G \cup S) = \mathcal{F}(v, G \cup F)$ then $M \subseteq S$. This means, because M is minimal and containment property of $\mathcal{F}$, that if $K \subset M$ then $\mathcal{F}(v, G \cup K) \subset \mathcal{F}(v, G \cup M)$. Therefore, $C_{\mathcal{F}}(v, G \cup K) < C_{\mathcal{F}}(v, G \cup M)$.

Since M is a connected graph, we can add edges to G in an order such that every time we add an edge there is only one connected component. Let define the ordered set as $X = \{e_1, e_2, ..., e_k\} = E(M)$. From this point we are going to prove two facts. First, let fix $i \in \{1, 2, ..., k-1\}$. Thus, for every subgraph $S \in \mathcal{F}(v, G + e_1 + e_2 + ... + e_i)$ then $S \in \mathcal{F}(v, G + e_1 + ... + e_{i+1})$. Second, there exists $S \in \mathcal{F}(v, G + e_1 + ... + e_{i+1})$ such that $S \notin \mathcal{F}(v, G + e_1 + e_2 + ... + e_i)$. The first result is a direct consequence of the containment property since $G + e_1 + e_2 + ... + e_i \subseteq G + e_1 + e_2 + ... + e_{i+1}$. For the second fact let suppose that every subgraph $S \in \mathcal{F}(v, G + e_1 + ... + e_{i+1})$ is in $\mathcal{F}(v, G + e_1 + ... + e_i)$. Then $e_{i+1} \notin E(S)$. If there exists $j > i + 1$ such that for some $S' \in \mathcal{F}(v, G + e_1 + ... + e_j)$ it is true that $e_{i+1} \in S'$. For the inclusion property and because the order of X ensure the connectivity of $K = S' - \{e_{i+2}, ..., e_j\} \subseteq S'$, we have that $K \in \mathcal{F}(v, G + e_1 + ... + e_j)$. Then, using containment property, we have that $K \in \mathcal{F}(v, G + e_1 + ... + e_{i+1})$. This contradicts the fact that every subgraph in $\mathcal{F}(v, G + e_1 + ... + e_{i+1})$ do not use the edge e_{i+1}. Therefore, j does not exists. In consequence, $\mathcal{F}(v, G + e_1 + ... + e_i) = \mathcal{F}(v, G + e_1 + ... + e_j)$ for every $j > i$ which contradicts the definition of M. Concluding that $C_{\mathcal{F}}$ satisfies the continuity axiom.

C.5 Proof Theorem 15

Let $\mathcal{F}$ be a family of substructures satisfying the isolated vertex axiom. If $C_{\mathcal{F}}$ satisfies the size axiom let choose the infinite sequence $\{L_n\}_{0 \leq n}$. We know that for every $N \in \mathbb{N}$ exists $m > 0$ such that $C_{\mathcal{F}}(0, L_m) \geq N$. This means that $\lim_{n \to \infty} 2^{C_{\mathcal{F}}(0, L_n)} = \infty$. On the other hand, if we know that $\lim_{n \to \infty} |\mathcal{F}(0, L_n)| = \infty$ then for every $N \in \mathbb{N}$ exits $m > 0$ such that $\log(|\mathcal{F}(0, L_m)|) \geq N$. Now, if $\{G_n\}_{n \geq 0}$ is any sequence of connected graphs such that $|V(G_n)| = n$, using the isolated vertex axiom we know that for every $m > 0$ it holds that $N \leq C_{\mathcal{F}}(0, L_m) \leq C_{\mathcal{F}}(0, G_m)$. Which means $C_{\mathcal{F}}$ satisfies the size axiom.

[1] Here is important to notice that families are close under graph isomorphism.

D Proofs Section 6

D.1 Proof of Theorem 17

For any set $U \subseteq V(G)$, the amount $\mathcal{A}(U, G(U))$ represent every subgraph not necessarily connected in $G(U)$ that contains every node in U. This means that every edge in $E(U)$ could be added or removed to make a new subgraph. Thus,

$$|\mathcal{A}(U, G(U))| = 2^{|E(G(U))|}.$$

On the other hand, let G' be a subgraph of G where every strongly connected component has at least one node from U. Then, G' can be divided into two parts. The one that is completely contained in $G(U)$ and the one that only has edges outside of $G(U)$. Then we have the following inequality

$$|\mathcal{A}(U, G)| \leq \sum_{G' \in \mathcal{A}(U, G(U))} |\mathcal{A}(U, G/U)| = |\mathcal{A}(U, G/U)||\mathcal{A}(U, G(U))|.$$

In the other direction, for every element in $|\mathcal{A}(U, G/U)||\mathcal{A}(U, G(U))|$ there is a corresponding subgraph of G where every strongly connected component has non empty intersection with S. Therefore,

$$|\mathcal{A}(U, G)| \geq |\mathcal{A}(U, G/U)||\mathcal{A}(U, G(U))|.$$

Finally, applying logarithm we conclude the proof.

D.2 Proof of Theorem 18

Let u be any node not directly connected with v. Then for $e = \{v, u\}$ we need an expression for $|\mathcal{A}(v, G + e)|$. Any subgraph in $G + e$ that contains v has the option to use e or not. For $S \in \mathcal{A}(v, G + e)$ such that $e \notin E(S)$ then $S \in \mathcal{A}(v, G)$. On the other hand, if $e \in E(S)$ then there is a corresponding subgraph $H \in \mathcal{A}(\{v, u\}, G/\{v, u\})$ that satisfies two properties. First, for all $e \in E(H)$ if $e = \{\{v, u\}, w\}$ then exists an edge $e' \in E(S)$ where $e' = \{w, v\}$ or $e' = \{w, u\}$. In second place, for every $e \in E(H)$ such that $\{v, u\} \notin e$ then $e \in E(S)$. In other words

$$|\mathcal{A}(v, G + e)| = |\mathcal{A}(v, G)| + |\mathcal{A}(\{v, u\}, G/\{v, u\})|.$$

From here we can notice that maximizing the variation is equivalent to maximize the amount of subgraphs in $\mathcal{A}(\{v, u\}, G/\{v, u\})$. The result follows from theorem 17 and monotonicity of logarithm.

E Proofs Section 7

E.1 Proof of theorem 19

Here we show that, unfortunately, some family of substructures make the computation of centrality an intractable task in the size of the graph. This proof is mainly based on the one made in [21] where Jerrum proved that counting labelled trees is complete for the class #P. In order to prove that $\textsc{Count}(\mathcal{A})$ and $\textsc{Count}(\mathcal{T})$ are #P-complete we need some preliminar problems.

1.

Problem:	ϕ
Input:	A graph G, $S \subseteq V(G)$ and $k > 0$
Output:	Amount of connected subgraphs covering S with exactly k nodes from G

2.

Problem:	ψ
Input:	A graph G and $S \subseteq V(G)$
Output:	Amount of connected subgraphs from G covering S.

3.

Problem:	#3-HAM
Input:	A cubic graph G
Output:	Amount of Hamiltonian paths in G.

4.

Problem:	CountAll($\mathcal{F}$)
Input:	A graph G
Output:	$\lvert \bigcup_{v \in V(G)} \mathcal{F}(v, G) \rvert$.

▶ **Theorem 21.** *$\#3 - HAM \leq \phi$ then ϕ is #P-hard.*

Proof. As seen in [21], the problem of counting the amount of labelled trees with k edges covering a set S of nodes is $\#P - Complete$. This is proven by a *weak parsimonius* reduction from this problem to $\#3 - HAM$. In this procedure the author use the fact that a minimum graph with $k = 4n - 3$ nodes covering a set S must be a tree corresponding to a Hamiltonian path in the original graph. We can use this exactly reduction to prove that ϕ is complete, we just use the exact same graph, $k = 4n - 3$ and the set S of nodes inside the structure defined in that reduction. This is correct since the only sub graphs we can count are trees. ◀

▶ **Theorem 22.** *$\phi \leq \psi$ then ψ is #P-hard.*

Proof. Given a graph G, a set $S \subseteq V(G)$ and $k > 0$, we want to count every connected sub graph of G covering S with exactly k nodes. We define the following elements:

1. The set of nodes $V(G_i) = V(G) \cup V(G) \times \{0, 1, ..., i - 1\}$.
2. The set of edges $E(G_i) = E(G) \cup \{(v, (v, j)) \mid v \in V(G) \text{ and } 0 \leq j \leq i - 1\}$.

For every $i, k \in \{1, ..., n\}$ define a_i as the amount of subgraph covering S in the graph G_i and N_k the amount of subgraphs from G covering S with k nodes. It is easy to notice that

$$a_i = \sum_{k=1}^{n} 2^{ik} N_k.$$

Thus, we can compute every a_i and recover the vector of N_k. This is the commonly used technique of interpolation. Then, the amount of subgraphs from G covering S with k nodes will be the component N_k from the vector. Finally, this reduction is clearly polynomial time to conclude the proof.

◀

▶ **Theorem 23.** *$\psi \leq$ CountAll($\mathcal{A}$) then CountAll($\mathcal{A}$) is #P-hard.*

Proof. For the final proof of this section, given a graph G and a set $S \subset$, we want to count every connected subgraph of G covering S. To prove this reduction we define the following elements similar to the previous proof:

1. The set of nodes $V(G_i) = V(G) \cup S \times \{0, 1, ..., i - 1\}$.
2. The set of edges $E(G_i) = E(G) \cup \{(v, (v, j)) \mid v \in S \text{ and } 0 \leq j \leq i - 1\}$.

Therefore, if N_k is the amount of connected subgraphs covering exactly k nodes from S, then the number of connected subgraphs from G_i is

$$\textsc{CountAll}(\mathcal{A})(G_i) = \sum_{k=0}^{|S|} 2^{ik} N_k.$$

From here we can use the interpolation technique and recover $N_{|S|}$ which is the value we are looking for. ◀

▶ **Theorem 24.** $\textsc{CountAll}(\mathcal{A}) \leq \textsc{Count}(\mathcal{A})$ *then* $\textsc{Count}(\mathcal{A})$ *is* #P*-hard.*

Proof. According to theorem 23 we know that $\textsc{CountAll}(\mathcal{A})$ is hard. Given an arbitrary graph G we can enumerate its nodes as $V(G) = \{v_1, v_2, ..., v_n\}$ then we can count the amount of sub graphs in G as

$$\begin{aligned}\textsc{CountAll}(\mathcal{A})(G) &= \textsc{Count}(\mathcal{A})(v_1, G) + \textsc{Count}(\mathcal{A})(v_2, G - \{v_1\}) + \\ &\quad \textsc{Count}(\mathcal{A})(v_3, G - \{v_1, v_2\}) + ... + \textsc{Count}(\mathcal{A})(v_n, G - \{v_1, v_2, ..., v_{n-1}\}) \\ &= \sum_{i=1}^{n} \textsc{Count}(\mathcal{A})(v_i, G - (\cup_{j<i}\{v_j\})).\end{aligned}$$

To show this, let S be any connected sub graph of G. Let $V(S)$ the nodes in S and let i be the minimum integer such that $v_i \in V(S)$. For every node v_j such that $j < i$ then $v_j \notin V(S)$. In other words we can not count S in any $\textsc{Count}(\mathcal{A})G - \{v_1, ..., v_{j-1}\}(v_j)$. On the other hand, since $v_i \in V(S)$ then S can not be counted in $\textsc{Count}(\mathcal{A})(v_k, G - \{v_1, v_2, ..., v_i, ..., v_{k-1}\})$ for every $k > i$. ◀

▶ **Corollary 25.** *If $\mathcal{F}$ is a family of substructures such that* $\textsc{CountAll}(\mathcal{F})$ *is #P-hard, then* $\textsc{Count}$*($\mathcal{F}$) is #P-hard.*

Proof. Using the same technique to prove theorem 23. ◀

In [21], Jerrum proves that $\textsc{CountAll}(\mathcal{T})$ is $\#P$-hard. Corollary 25 shows that $\textsc{Count}(\mathcal{T})$ is also $\#P$-hard.

To conclude the proof of theorem 19 we need to show that $\textsc{Count}(\mathcal{A})$ and $\textsc{Count}(\mathcal{T})$ are actually in the class $\#P$. It is clear that exists a non deterministic Turing machine that enumerate edges in $E(G)$. Then, non deterministically chooses edges following the enumeration. Finally, the machine corroborates if the subgraph defined by those edges is connected and contains v which can be done in polynomial time. In the case of $\textsc{Count}(\mathcal{T})$, determining if a subgraph is a connected tree is also possible in polynomial time.

E.2 Proof of theorem 20

We introduce some definitions needed for this proof.

▶ **Definition 26.** *Given a graph G, we call the structure $\mathcal{T} = (T, \mathcal{B})$ a tree decomposition of G if it satisfies the following conditions:*

1. *$T = (V(T), E(T))$ is a tree graph.*
2. *The union of all the sets(bags) in $\mathcal{B}$ is equal to $V(G)$.*
3. *For every edge $(u, v) \in E(G)$ there is a bag $B \in \mathcal{B}$ such that $v, u \in B$.*
4. *For a vertex $x \in V(T)$ We denote B_x as the bag associated with x. Let u, v be two nodes of $V(T)$ and w is a vertex that belongs to the unique path connecting u, v in T. B_v and B_u. then $B_u \cap B_v \subseteq B_w$.*

The definition above is equivalent to the one defined in the section but this notation is more useful for this proof.

▶ **Definition 27.** *Given a tree decomposition $\mathcal{T} = (T, \mathcal{B})$, the width of $\mathcal{T}$ is*

$$W(\mathcal{T}) = \max_{B \in \mathcal{B}} |B| - 1.$$

▶ **Definition 28.** *At the same time we can define the tree width of a graph G as*

$$TW(G) = \inf_{\mathcal{T} \in TD(G)} W(\mathcal{T}).$$

In other words, the minimum possible width for a tree decomposition of G.

▶ **Definition 29.** *Given a tree decomposition $\mathcal{T} = (T, \mathcal{B})$ and $B \in \mathcal{B}$. We denote the node associated with B as $\epsilon(B) \in V(T)$.*

▶ **Definition 30.** *Given two bags B_v, B_u from a tree decomposition $\mathcal{T} = (T, \mathcal{B})$ of a graph G. If v is a leaf in T, we define the operator* $\textsc{Fusion} : \mathcal{B}^2 \times TD(G) \to TD(G)$ *such that* $\textsc{Fusion}(B_1, B_2, \mathcal{T}) = (T', \mathcal{B}')$ *if and only if $T' = (V(T) - \{v\}, E(T) - \{v, u\})$ and $\mathcal{B}' = (\mathcal{B} - \{B_v, B_u\}) \cup (B'_u)$ where $B'_u = B_u \cup B_v$. In other words, we merge a leaf with its father and generate a new tree decomposition where the bag associated with the father contains the nodes in the leaf.*

The following definition allows us to formalize a way to recover a graph G from a tree decomposition $\mathcal{T}$. This construction is necessary to compute all-subgraphs centrality recursively.

▶ **Definition 31.** *For a tree decomposition $\mathcal{T} = (T, \mathcal{B})$ of a graph G, we call a binary labeled tree graph $H = (V, E, L)$ a recomposing tree graph of G if satisfies the following conditions:*

1. *For every $v \in \textsc{Leaves}(H)$, $L(v) \in \mathcal{B}$.*
2. *For every $u \notin \textsc{Leaves}(H)$, $L(u) = \textsc{Fusion}$.*
3. *If $\{v, u\} \subseteq \textsc{Leaves}(H)$, such that $\textsc{Parent}(v) = \textsc{Parent}(u)$ then there exists $x, y \in V(T)$ such that $\{x, y\} \in E(T)$, $L(v) = B_x$, $L(u) = B_y$ and $\{x, y\} \cap \textsc{Leaves}(T) \neq \varnothing$.*
4. *$\bigcup_{v \in \textsc{Leaves(H)}} L(v) = V(G)$.*
5. *For $u, v \in \textsc{Leaves}(H)$, such that $u \neq v$ then $L(u) \neq L(v)$.*

The most important property of a recomposing tree graph is stated in the following lemma.

▶ **Lemma 32.** *Let $H = (V, E, L)$ be a recomposing tree graph of a tree decomposition $\mathcal{T} = (T, \mathcal{B})$ for G. If $\{v, u\} \in \textsc{Leaves}(H)$ such that for a vertex $w \in V(H), w = parent(v) = parent(u)$. Then the graph $H' = (V - \{u, v\}, E - \{\{u, w\}, \{v, w\}\}, L')$ where $L'(z) = L(z)$ if $z \notin \{u, v, z\}$ and $L'(w) = L(u) \cup L(v)$ is a recomposing tree graph of the tree decomposition $\mathcal{T}' = \textsc{Fusion}(B_{\epsilon(L(u))}, B_{\epsilon(L(u))}, \mathcal{T})$ for G. We call this operation $\textsc{Merge}(v, u, H) = H'$.*

Proof. We just have to show that H' satisfies the conditions for a recomposing tree graph.

1. Let $v \in \textsc{Leaves}(H')$. Notice that the node z is now in the leaves of H'. Then, If $v \neq z$ then $L'(v) = L(v) \in \mathcal{B}$. If $v = z$ then by the definition of the $\textsc{Fusion}$ operator we have that $L'(z) = B_{\epsilon(L(u))} \cup B_{\epsilon(L(v))} \in \mathcal{B}'$.
2. Since we did not change any node $v \notin \textsc{Leaves}(H)$ except for z, this property is still satisfied.

3. If exists $x \in \textsc{Leaves}(H')$ such that $\textsc{Parent}(x) = \textsc{Parent}(z)$ then there is a node $w = \epsilon(L'(x)) \in \textsc{Leaves}(T)$ such that $L'(x) = B_w$ because H' satisfies the first property of a recomposing tree graph. On the other hand, we know that H satisfies the property 3 of a recomposing tree graph. Which means that the node w was connected to $s \in V(T)$. Without loss of generality let assume $s = \epsilon(L(u))$. Therefore, y will be connected to s in $\mathcal{T}'$ which implies the property since $B'_s = L'(z)$. In any other case if the vertex z is not involved, the property holds because H was a recomposing tree graph.
4. From the definition of $\textsc{Fusion}(B_{\epsilon(L(u))}, B_{\epsilon(L(u))}, \mathcal{T})$ we know that $\mathcal{T}'$ is a tree decomposition for G. On the other hand, for any other node different than z we are maintaining the labels in L. Then, H' will satisfy that $\cup_{v \in \textsc{Leaves}(H')} L'(v) = V(G)$.
5. This holds from the fact that H is a recomposing tree graph and $L'(z) \neq L'(v)$ for any $v \neq z$.

◀

This property shows the way to re use a recomposing tree graph after merging two bags of a tree decomposition. Moreover, it gives a recursive method to use the structures defined before. Now we show that recomposing trees are easy to compute. Which is necessary in order to use this tool.

▶ **Lemma 33.** *Given a tree decomposition $\mathcal{T} = (T, \mathcal{B})$ of a graph G, there exist a polynomial time algorithm that generate a recomposing tree graph for G from $\mathcal{T}$.*

Proof. Intuitively we can assign every leaf v in $V(T)$ to a leaf x in H. Then, $L(x) = B_v$. Now for a node $u \in V(T)$ such that $\{u, v\} \in E(T)$ then we add two nodes to $V(H)$. First w which is associated with u and $L(w) = B_u$. Second is an auxiliary node containing the fusion operator, let say z and $L(z) = \textsc{Fusion}$. Then we add $\{w, z\}$ and $\{v, z\}$ to $E(H)$. It is clear that this can be done recursively following an order in the tree T from leaves to the root. ◀

Now that we know how to recompose the graph from a tree decomposition, we need an especial definition for tree decompositions that satisfy desirable properties when computing All-Subgraphs Centrality.

▶ **Definition 34.** *Given a graph $G = (V, E)$, we define a subgraph tree decomposition as $\mathcal{T} = (T, (\mathcal{B}, \tau))$. Where $(T, \mathcal{B})$ is an usual tree decomposition of G and $\tau : \mathcal{B} \to 2^E$ is a function from bags of $\mathcal{T}$ to a set of edges from G. In other words τ maps every bag of nodes to a subset of edges which defines completely a subgraph of G. In order to $\mathcal{T}$ defines a correct subgraph tree decomposition, τ must satisfy that for every pair of bags A, B in $\mathcal{B}$, $\tau(A) \cap \tau(B) = \varnothing$.*

This kind of tree decomposition let us to not count twice the same edge when recomposing the graph. The definition of the $\textsc{Fusion}$ operator is equivalent, we just have to maintain the edges after the fusion. We are not proving the fact that we can efficiently construct a subgraph tree decomposition from a conventional tree decomposition. Intuitively, we just have to assign every edge $e = \{u, v\} \in E(G)$ to one bag B_w such that $e \subseteq B_w$. From now on we consider the set of shared nodes from two adjacent bags as $D = B_v \cap B_u$. Since we want to count every possible connected subgraph we can partition them by the nodes they have. Anyways, it is important to explicitly exclude the nodes that we do not want to appear at the counting. In order to simplify our notation, we write $\mathcal{N}(v, G) = |\mathcal{A}(v, G)|$. For a set of sets of nodes $P \subseteq 2^{V(G)}$, we also define

$$\mathcal{N}(P, G) = |\{S \subseteq G \mid \text{For every } A \in P, \text{ exists } H \in \text{ConnComp}(S) \text{ such that } A \subseteq V(H)\}|$$

. In other words, $\mathcal{N}(P,G)$ is the amount of subgraphs where every connected component cover some set in P. The main idea of the algorithm is based on the following result.

▶ **Lemma 35.** *Given a graph G and a set of nodes D from G. The size of the All-Subgraph family of a node $v \in D$ is*

$$\mathcal{N}(v,G) = \sum_{A \subseteq D, v \in A} \mathcal{N}(\{A\}, G - (D - A)).$$

Proof. For every subgraph S of G containing v, we have just one maximal subset of D that has x and is a subset of node from S, lets say A. Since A is maximal, every node in $D - A$ is not in S. Thus, S will be counted in $\mathcal{N}(\{A\}, G - (D - A))$. Then, we have that

$$\mathcal{N}(v,G) \leq \sum_{A \subset D, v \in A} \mathcal{N}(\{A\}, G - (D - A)).$$

On the other hand, the family of the set $\{A\}$ will count subgraphs that have the nodes in A and only one connected component. Since $v \in A$, we are counting connected subgraph of G containing v. Then we have that

$$\mathcal{N}(v,G) \geq \sum_{A \subset D, x \in A} \mathcal{N}(\{A\}, G - (D - A)).$$

◀

Since we can count subgraphs containing a vertex $v \in D$ in this way, our goal is to know the value of $\mathcal{N}(\{A\}, G - (D - A))$ for every set $A \subseteq D$ such that $v \in A$. Now we need a tool to compute those values by taking advantage from the structure of a tree decomposition. Here we introduce partitions to accomplish this.

▶ **Definition 36.** *For every $A \subseteq D$, we define a partition of A as a collection $\{P_i\}_{i=1}^{k}$ satisfying the following conditions:*

1. *For all i, j in $\{0, 1, \ldots, k\}$, if $j \neq i$ then $P_i \cap P_j = \varnothing$.*
2. *The union of every P_i is equal to A.*

▶ **Definition 37.** *Given a subset A of D the set of all partitions of A is denoted as $Part(A)$.*

It is important to notice that given a set $A \subseteq D$ when we fuse two bags B_1, B_2 from a subgraph tree decomposition, there are some connected subgraphs in the merged graph that will have every node in A. Thus, we can look to this subgraph as two separated parts. The one that belongs to $G(B_1)$ and the part that was in $G(B_2)$. At the same time, this two parts can look like unconnected subgraphs inside the corresponding graph. Nevertheless, in the merged graph they generate a connected subgraph. This situation is captured by an order in partitions.

▶ **Definition 38.** *Let P_1 and P_2 two partitions from a set $A \subseteq D$. We say that $P_1 \leq P_2$ if for every set B in P_1 exists a set C in P_2 such that $B \subseteq C$. In other words, P_1 refine P_2.*

It is a well known result that the set of partitions of a set A, $Part(A)$ is a partially order set with the relation defined above. Even more this conform a lattice, which means that every subset from $Part(A)$ has an infimum and supremum. The supremum under this order is exactly what we needed to generate a connected subgraph based in partitions.

▶ **Definition 39.** *Given two partitions P_1, P_2 from a set $A \subseteq D$, we say that the partition P is the supremum of P_1 and P_2 if for all set $C \in P_i, i = 1, 2,$ exists a set D in P such that $C \subseteq D$. We denote this as*

$$P = \sup\{P_1, P_2\}.$$

Now, let assume we have G in two separated parts G_1, G_2 such that $V(G_1) \cap V(G_2) = D$, $E(G_1) \cap E(G_2) = \varnothing$ and $G_1 \cup G_2 = G$. In other words, we can generate G by merging this two separated parts. The following result shows a way to compute the values $\mathcal{N}(\{A\}, G-(D-A))$ from those two parts.

▶ **Lemma 40.** *Let $\mathcal{T} = (T, (\mathcal{B}, \tau))$ be a subgraph tree decomposition of a graph G. Let G_1 and G_2 two graphs defined by two adjacent bags of $\mathcal{T}$. Let D be the set of nodes that both bags share, i.e $D = B_1 \cap B_2$. If $P \in Part(D - R)$ we denote the substructures of P without R in the graph G_i as $\mathcal{N}(P, G_i - R)$ where $i = 1, 2$. Thus, the substructures of P without R in the graph G is*

$$\mathcal{N}(P, (G_1 \cup G_2) - R) = \sum_{\substack{P_1, P_2 \in Part(D-R) \\ P = \sup(P_1, P_2)}} \mathcal{N}(P_1, G_1 - R)\mathcal{N}(P_2, G_2 - R).$$

Proof. The main idea is to count subgraphs containing P using the information we already know before the fusion. Let S be a subgraph of G where every set A in P is contained exclusively in one connected component of S and every node from R is not in $V(S)$. We denote $CC(A)$ as the connected component associated with the set A. At the same time, we can separate every $CC(A)$ in the section corresponding to each bag B_i for $i \in \{1, 2\}$. Each of this sections will have $m_i(A) \geq 1$ connected components in the corresponding graph $G(B_i)$ and A appears in both sections. From here is easy to notice that for every k in $\{1, 2, ..., m_i(A)\}$ there is a set $L_k^i(A) \subseteq A$ such that $\bigcup_{k \in \{1,2,...,m_i\}} L_k^i(A) = A$. We can do this for every set A. Finally, we define

$$P_i = \bigcup_{A \in P} \bigcup_{k=1}^{m_i(A)} \{L_k^i(A)\}.$$

It is clear that every P_i do not contain R because it is not in S. Moreover, P_i is a partition of $D - R$ and $P = \sup\{P_1, P_2\}$. Concluding that

$$\mathcal{N}(P, (G_1 \cup G_2) - R) \geq \sum_{\substack{P_1, P_2 \in Part(D-R) \\ P = \sup(P_1, P_2)}} \mathcal{N}(P_1, G_1 - R)\mathcal{N}(P_2, G_2 - R).$$

In the other direction, we know that if two partitions P_1, P_2 of $D - R$ have P as its supremum. Then for every set A from P there are two collection of sets $L_1(A) \subseteq P_1$ and $L_2(A) \subseteq P_2$ such that

$$\bigcup_{L \in L_i(A)} L = A.$$

This means that for every set H in $L_1(A)$ exists a set J in $L_2(A)$ such that $H \cap J = K \neq \varnothing$. The same can be verified for $L_2(A)$. In other words, every connected component having H in the bag B_1 will be connected in B_2 through the nodes in K. Since this is true for every set in $L_i(A)$ we can create a complete connected component in S by merging a subgraph counted in $\mathcal{N}(P_1, G_1 - R)$ and other subgraph counted in $\mathcal{N}(P_2, G_2 - R)$. Concluding that

$$\mathcal{N}(P, (G_1 \cup G_2) - R) \leq \sum_{\substack{P_1, P_2 \in Part(D-R) \\ P = \sup(P_1, P_2)}} \mathcal{N}(P_1, G_1 - R)\mathcal{N}(P_2, G_2 - R).$$

◀

This result is assuming we have stored the values of $\mathcal{N}(P, G_i - R)$ for both graphs G_i and for every possible partition P which depends on the set D. To make a recursive procedure, we have to use information from previous iteration in order maintain it as a polynomial time algorithm. The main issue left arises after merging two bags from a tree decomposition. After one iteration we have all the information for every sub partition as in lemma 40. Then, for the next iteration of the algorithm we would like to merge the graph generated during last iteration with one bag from the original tree decomposition. The problem is that the actual set D' of vertices that both bags share is different from the one in the previous iteration.

▶ **Lemma 41.** *Let $\mathcal{T} = (T, (\mathcal{B}, \tau))$ be a subgraph tree decomposition of a graph G, B be a bag from $\mathcal{B}$ and $D \subseteq B$. For a vertex $v \notin B$ define $D' = (D \cup v)$. If H is the subgraph induced by B and S the subgraph induced by $B \cup \{v\}$ then for any $P \in Part(D' - R)$*

1. *If $\{\{v\}\} \cap P = \varnothing$ and $v \notin R$. Then $\mathcal{N}(P, S - R) = 0$.*
2. *If $\{\{v\}\} \cap P = \varnothing$ and $v \in R$. Then $\mathcal{N}(P, S - R) = \mathcal{N}(P, H - (R - \{v\}))$.*
3. *If $\{\{v\}\} \cap P \neq \varnothing$ then $\mathcal{N}(P, S - R) = \mathcal{N}(P - \{\{v\}\}, H - R)$.*

For a bag $B \in \mathcal{B}$ we call this operation $\textsc{Add}(v, B)$.

Proof. The idea behind this procedure is to use previous information to compute values associated with v. Since v did not belong to the bag then none of the previous nodes is connected with v.

In the first case, we have that $v \notin R$, then must be a set A in P such that $v \in A \neq \{k\}$. Then, we are counting subgraphs where the node v must be connected with others. Which is impossible since we say that v will have no connections.

In the second case, we want subgraphs that do not contain v. Which is the same as counting the subgraphs before adding v to the bag.

Finally we would want to count the amount of subgraphs that has v as a separated connected component. This clearly is the same amount of subgraphs of the partition $P - \{\{v\}\}$ in the graph H because v is disconnected in S. ◀

Lema 41 shows the way to add a new vertex to the bag that is not connected to any vertex inside and update the information about the subgraphs of every partition. At the same time, there may be some vertex u in the previous iteration belonging to D, that the algorithm will not use anymore for the computation. Here we would like to forget the information related to u and use it to include a new vertex needed for a future iteration. The following result show how to update the values of every sub partition in a new set $D' = D - \{u\}$.

▶ **Lemma 42.** *Let $\mathcal{T} = (T, (\mathcal{B}, \tau))$ be a subgraph tree decomposition of a graph G, B be a bag from $\mathcal{B}$ and $D \subseteq B$. For a vertex $u \in B$ define $D' = (D - \{u\})$. If H is the subgraph induced by B then for any $P \in Part(D' - R)$*

$$\mathcal{N}(P, H - R) = \sum_{A \in P} \mathcal{N}([P - A] \cup \{A \cup \{u\}\}, H - R) + \mathcal{N}(P, H - (R \cup \{u\})).$$

For a bag B, we call this operation $\textsc{Forget}(u, B)$.

Proof. We want to forget the information associated with u and pass it to other partitions. In other words, we will count every possible subgraph containing the partition P which is not computed when u is not present in R nor P. For this we sum over different sub partitions, $P' = [P - A] \cup [A \cup \{u\}]$ which cover every possibility for a sugraph containing u. Finally, it is possible that u did not belong to any connected component of a subgraph we want to count. This is why we add the final term $\mathcal{N}(P, H - R \cup \{u\})$. ◀

Finally, the following algorithm compute All-subgraph centrality in polynomial time for bounded tree width.

1. Given a graph G of bounded tree width and$v \in V(G)$, compute a subgraph tree decomoposition $\mathcal{T} = (T, (\mathcal{B}, \tau))$.
2. Compute a recomposing tree graph $H = (V, E, L)$ for G such that exists $u \in \textsc{Leaves}(H)$ where $v \in L(u)$ and $\textsc{Parent}(u) = \textsc{Root}(H)$.
3. For every bag $B \in \mathcal{B}$ define $D_B = B$. Then, for every $R \subseteq D_B$ and every $P \in Part(D_B - R)$ compute $\mathcal{N}(P, G(B))$ using algorithm 1.
4. Choose $\{w, z\} \subseteq \textsc{Leaves}(H) - \{u\}$. Such that $\textsc{Parent}(w) = \textsc{Parent}(z)$. If $\epsilon(L(w)) \in \textsc{Leaves}(T)$ define $D = D_{L(z)} \cap D_{L(w)}$. Then $y_1 \in D_{L(z)} - D$ and $y_2 \in D_{L(w)} - D$. Then apply $\textsc{Add}(y_1, L(w))$.
5. Redefine $\mathcal{T} = \textsc{Fusion}(L(w), L(z), \mathcal{T})$ and $H = \textsc{Merge}(w, z, H)$. Define $D_{L(z)} = D$.
6. For every $R \subseteq D_{L(z)}$ and for every $P \in Part(D_{L_z} - R)$, using lemma 40 compute $\mathcal{N}(P, L(z))$.
7. Apply $\textsc{Forget}(y_2, L(z))$.
8. If $|\mathcal{B}| > 1$ go to 4.
9. Output $C_{\mathcal{A}}(v, G)$ using lemma 35 and the information for $B \in \mathcal{B}$.

The correctness of this algorithm follows from the proof of every result in the section. Now, we need to show that this is a polynomial time algorithm in the size of G. In first place, as stated in [5], if $TW(G) = k$, there is an algorithm that takes time $O(2^k n)$ to get a tree decomposition $\mathcal{T}$ of width k. Therefore, we know a subgraph tree decomposition can be obtained in time $O(n)$. Second, by lemma 33 we can compute a recomposing tree graph in time $O(n)$. In third place, we know that for every bag $B \in \mathcal{T}$ then $|B| \leq k$. Therefore, the size of every partition P is bounded by k. This means that at most we are computing 2^k values $\mathcal{N}(P, G(B))$ in every iteration. It is also known that $|V(T)| \leq n$. Then, in step 3 we are running algorithm 1 at most $n2^k$ times, which still lineal in n since $|G(B)| \leq k$. Then, every iteration after step 3 is merging two bags and updating the values $\mathcal{N}(P, G)$. This update will take $O(2^k)$ by the same argument as before. Step 7 is necessary to do not store unnecessary values, which also takes time $O(2^k)$. Since $|\mathcal{B}| \leq n$ we are not repeating this process for more than n iterations. Finally we apply lemma 35 which only use the information of a few partitions. This concludes that the algorithm takes time $O(f(k)n)$.