FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of bachelor's thesis

Title:	Computational Complexity of Maximum Betweenness Centrality: A Multivariate Analysis
Student:	José Gaspar Smutný
Supervisor:	Ing. Šimon Schierreich
Study program:	Informatics
Branch / specialization:	Computer Science
Department:	Department of Theoretical Computer Science
Validity:	until the end of summer semester 2023/2024

Instructions

We study the NP-hard Maximum Betweenness Centrality problem (MBC), which can be defined as follows. We are given a simple undirected and connected graph G=(V,E), a function w assigning each vertex its cost, and a budget b. Our goal is to find a set C, where the sum of costs of all vertices in C is at most the budget such that the probability of detection of communication between two vertices s,t∈V is maximized. We assume that vertices communicate along the shortest (s,t)-paths and that the shortest path used for communication is chosen uniformly at random.

1. Make yourself familiar with the Maximum Betweenness Centrality problem [1,2] and the framework of parametrised complexity [3].
2. Find reasonable natural parameters for the MBC problem and try to derive the computational complexity with respect to these parameters.
3. Study the MBC problem with respect to selected structural parameters.
4. Combine parametrisation from 1. and 2. to obtain additional tractability and hardness results.

References:
[1] Fink, M., Spoerhase, J. (2011). Maximum Betweenness Centrality: Approximability and Tractable Cases. In: Katoh, N., Kumar, A. (eds) WALCOM: Algorithms and Computation. WALCOM 2011. Lecture Notes in Computer Science, vol 6552. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-19094-0_4

Electronically approved by doc. Ing. Jan Janoušek, Ph.D. on 21 September 2022 in Prague.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

[2] Puzis, R., Elovici, Y., and Dolev, S (2007). Finding the Most Prominent Group in Complex Networks. AI Communications 20(4):287–296. Available from: https://content.iospress.com/articles/ai-communications/aic411
[3] Cygan, M., Fomin, F.V., Kowalik, Ł., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S. (2015). Parameterized Algorithms. Springer, Cham. ISBN: 978-3-319-21274-6

Bachelor's thesis

Computational Complexity of Maximum Betweenness Centrality: A Multivariate Analysis

José Gaspar Smutný

Faculty of Information Technology
Department of Theoretical Computer Science
Supervisor: Ing. Šimon Schierreich
May 11, 2023

Citation of this thesis: Smutný José Gaspar. *Computational Complexity of Maximum Betweenness Centrality: A Multivariate Analysis.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2023.

Contents

List of Figures

List of Tables

Děkuji hlavně Šimonovi Schierreichovi, nejen kvůli cenné diskuze a podnětné rady v průběhu vytvoření bakalářské práce, nýbrž i za kvalitní a zajímavou výuku jež jsem měl příležitost se zůčastnit, a která mi otevřela brány do světa teoretické informatiky.

I am also grateful to Alan Howlett, who piqued my interest in computing, and whose friendship I am most fond of.

Bez nadmíru pečlivem a mílem přístupu všech mých účitelů a spolužáků ze Dvořákova Gymnázia nebyl bych býval schopen tuto práci napsat. I těm jsem vděčný.

Por último pero no por ello menos importante, le agradezco a mi familia el soporte y la paciencia que me han dedicado, no solo mientras trabajaba en mi tesis.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60 (1) of the Act.

In Prague on May 11, 2023 ....

Abstract

The MAXIMUM BETWEENNESS CENTRALITY problem consists in picking a group of b nodes from a social network so that it has the greatest likelihood of detecting communication, assuming that nodes communicate only along shortest paths, chosen uniformly at random. Finding such groups is relevant for the study of communication in complex networks. Although this problem has been studied within the framework of classical complexity, its parameterized complexity has remained an open question. We close this gap by studying MAXIMUM BETWEENNESS CENTRALITY from a parameterized perspective. We show that MAXIMUM BETWEENNESS CENTRALITY is W[1]-hard when parameterized by the budget b, and complement this result with a lower bound of $f(b) \cdot n^{o(b)}$ for any algorithm for MAXIMUM BETWEENNESS CENTRALITY parameterized by the budget b, under ETH. On a positive tone, we show that MAXIMUM BETWEENNESS CENTRALITY is in FPT when parameterized by the vertex cover number, by the distance to clique and the budget, or by the twin cover number and the budget.

Keywords Parameterized complexity, Maximum Betweenness Centrality, Fixed Parameter Tractable, Vertex cover number, Distance to clique, Twin cover number.

Abstrakt

Problém MAXIMUM BETWEENNESS CENTRALITY spočívá ve výběru skupiny úzlů uživateli zadané velikosti ze sociální sítě tak, abychom maximalizovali pravděpodobnost zachycení komunikace probíhající po nejkratších cestách v rámci sítě. Byť MAXIMUM BETWEENNESS CENTRALITY je poměrně probádáný v kontextu klasické složitosti, jeho parametrizovánáá složitost představovala dosavaď nezdupanou půdu. Tento nedostatek doplníme multivarietní analýzou. Ukázujeme, že MAXIMUM BETWEENNESS CENTRALITY je W[1]-těžký parametrizován velikostí skupiny. Tento výsledek doplňujeme dolní mezí $f(b) \cdot n^{o(b)}$ pro algoritmy pro MAXIMUM BETWEENNESS CENTRALITY parametrizován velikostí skupiny b, za předpokladu, že platí ETH. Oproti tomu ukazujeme že MAXIMUM BETWEENNESS CENTRALITY je v FPT když je parametrizován buďto parametrem vertex cover number, distance to clique a velikostí skupiny, nebo twin cover number a velikostí skupiny.

Klíčová slova Parametrizovaná složitost, Maximal Betweenness Centrality, FPT, Vertex cover number, Distance to clique, Twin cover number.

Abbreviations

BC	Betweenness Centrality
$\widetilde{\mathrm{BC}}$	Path Betweenness Centrality
ETH	Exponential Time Hypothesis
FPT	Fixed-Parameter Tractable
FPT	Fixed-Parameter Tractable
GBC	Group Betweenness Centrality
i. e.	That is
no.	Number
NP	Nondeterministic Polynomial time
P	Polynomial time
XP	Slice-wise Polynomial

Introduction

When studying networks that model communication a natural question arises: are some nodes in a network more important than others? The notion of *centrality* was conceived in the second half of the XXth century in an attempt to answer this question. Social scientists were searching for tools that would provide them with a better understanding of the structure of social networks. The observations of Bavelas [1] regarding patterns of centrality in graphs modeling communication were formalized by Freeman [2] and resulted in the introduction of multiple measures of centrality. Many new approaches to the measurement of centrality have since been described, each focusing on different aspects of network interactions. The Degree Centrality of a node is equal to its number of direct connections. Hence a more directly connected node is more central from a Degree Centrality point of view. Calculating the Closeness Centrality of a vertex is slightly more involved. From the perspective of Closeness Centrality, a node that is “close” to all others is considered to be more central. We focus on Betweenness Centrality, introduced by Freeman [2]. Betweenness Centrality gauges centrality based on shortest-path communication. The Betweenness Centrality of a node is equal to the sum of the fractions of shortest paths that contain said node for each pair of nodes in a network. Time has shown Betweenness Centrality to be an useful tool for the analysis of networks modeling communication. It has aided the study of wireless [3, 4, 5], transportation [6, 7, 8] and biological networks [9, 10, 11], among others.

Group Betweenness Centrality as a natural generalization of Betweenness Centrality was introduced by Everett and Borgatti [12] in 1999. The group composed of yellow vertices in Fig. 1 is the group of size 4 with maximum Betweenness Centrality. In other words, it is the group with the greatest likelihood of detecting communication in the network.

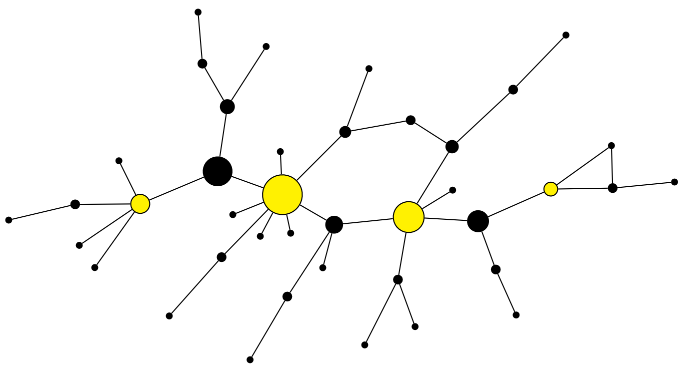

Figure 1 An example of a network. The size of each vertex is proportional to its Betweenness Centrality. The yellow vertices form the set of size 4 with maximum Betweenness Centrality.

Since its introduction, Group Betweenness Centrality has been used for determining ideal nodes for the monitoring of communication [13] and in general to gain an understanding of the combined influence of a particular group [12, 14, 15, 16]. Various non-equivalent definitions of Group Betweenness Centrality have appeared since its introduction, and an explanation of the differences between them is given in Section 3.2.

We concern ourselves with the Maximum Betweenness Centrality problem, that consists in finding a group of vertices of given size such that their Group Betweenness Centrality is at least equal to a particular threshold.

Although calculating the Betweenness Centrality of a group can be done in $\mathcal{O}(nm)$ time [12], by means of the $\mathcal{O}(nm)$ algorithm for calculating the Betweenness Centrality of every individual vertex [17, 18], finding effective exact algorithms for the Maximum Betweenness Centrality problem has proven to be a demanding task. As the number of possible groups of vertices of size b is in the worst case equal to n^b, a trivial algorithm that calculates the GBC of every group of size b runs in $\mathcal{O}(nm \cdot n^b)$ time. In Chapter 3 we introduce two new algorithms for Maximum Betweenness Centrality. One is based on the algorithm of Puzis et al. [19] for calculating the Group Betweenness Centrality of a group of size b in $\mathcal{O}(b^3)$ time, after $\mathcal{O}(nm)$ time preprocessing, and has a running time of $\mathcal{O}(nm + b^3 n^b)$. In a similar manner, we use the new algorithm described in Section 2.2 to calculate the GBC of all groups of size b in $\mathcal{O}(nm + n^{b+2})$.

As Maximum Betweenness Centrality is NP-complete[1], the latest research has been directed towards finding approximation algorithms, which are better suited for the extremely large instances that occur in natural settings. Mahmoody et al. [20] describes an approximation algorithm based on random path sampling. Very recently, Lagos et al. [21] presented an approximation based on integer programming with random path sampling. Although Angriman et al. [22] focuses on a slightly different problem, they also concern themselves with efficient approximations of group centrality. Although approximation schemes offer better tractability results, they are lacking in precision. We take a different approach, and study Maximum Betweenness Centrality from a parameterized perspective.

Parameterized algorithmics excel by being more granular in their analysis of complexity. As opposed to classical complexity, where the running time is expressed as a function of the total input size, the running time of a parameterized algorithm is a function of the input size and of a parameter (or combined parameters), which provides useful information about the structure of the instance. The addition of the parameter allows for a finer expression of complexity.

The theory of parameterized complexity has been very influential, and is still an active area of research. In the preface to their textbook Cygan et al. [23] stated that in 2014 Google scholar listed more than 4 000 works containing the term "fixed-parameter tractable." At the time of writing the number has more than doubled, and there are more than 10 000 such works. We attempt to complement these results by studying Maximum Betweenness Centrality from a parameterized point of view. The goal of this work is to obtain tractability and hardness results for the Maximum Betweenness Centrality problem parameterized by reasonable natural parameters. For a given parameter, we wish to show whether Maximum Betweenness Centrality is in FPT or in some level of the W hierarchy. If not in FPT, we wish to determine lower bounds for the computational complexity with respect to the parameter, and compare them to the results attained by unparameterized algorithms.

[1] Puzis et al. [19] hints at a proof that Maximum Betweenness Centrality is NP-hard. In Chapter 3, we show that Maximum Betweenness Centrality is not only NP-hard, but it is even NP-complete.

Chapter 1

Preliminaries

1.1 Basic notation

We refer to sets of elements using capital letters and to their elements using small letters. A set of sets is denoted by calligraphic capital letters.

The elements of ordered sets, like $A = (1, 2, 3)$, are listed inside of parenthesis, and the order of the elements indicates their ordering in the set. The elements of unordered sets, like $B = \{1, 2, 3\}$, are listed inside of braces, and any ordering results in an equivalent set.

Let S be a set. A subset of S is any set S' such that all elements contained in S' are also contained in S. The set of all unordered subsets of S of size k is denoted by $\binom{S}{k}$.

Let A and B be two sets. The cartesian product $A \times B$ of A and B is the set of all ordered sets (a, b), such that $a \in A$ and $b \in B$. A^2 is used to denote the cartesian product of A with itself. By A^* we denote the cartesian product of A with itself an arbitrary number of times. Hence, for $A = \{1, 2, 3\}$, we have $(1, 1, 1) \in A^*$, $\emptyset \in A^*$ or $(1, 3, 2) \in A^*$.

1.2 Graph theory

We refer to Diestel et al. [24] for basic graph theory notation.

A *simple, undirected graph* $G = (V, E)$ is defined as a pair of sets, such that $E \subseteq \binom{V}{2}$ and V is non-empty. The elements of V are called *vertices*, and the elements of E are called *edges*. We use n to denote the number of vertices and m to denote the number of edges of G, respectively.

An u-v edge is written as $\{u, v\}$. Vertices $u, v \in V$ are *adjacent* when an $\{u, v\}$ edge exists. The u and v vertices are the *endpoints* of the $\{u, v\}$ edge, they are *incident* with the $\{u, v\}$ edge.

A *path* with k edges is a sequence of unique vertices $v_0, \ldots, v_k$, such that an $\{v_i, v_{i+1}\}$ edge exists for $i \in \{0, \ldots, k-1\}$. An u-v path denotes a path whose endpoints are the u and v vertices. The *distance* between two vertices u, v is defined as the length of the shortest u-v path, and is denoted by $\text{dist}_G(u, v)$. If no s-t path exists, then we have $\text{dist}_G(s, t) = \infty$. A graph is *connected* when there is a path between every pair of vertices.

The *degree* of a vertex v is defined as the number of edges which are incident to v, and is denoted by $\deg_G(v)$. If all the vertices of G have the same degree r, then G is *r-regular*, or simply *regular*. The *neighbourhood* of a vertex v is defined as the set of vertices that share an edge with v, and is denoted as $N(v)$.

Let $G = (V, E)$ be a graph. We say the graph $G' = (V', E')$ is a *subgraph* of G when $V' \subseteq V$ and $E' \subseteq E$, and write $G' \subseteq G$. If $G' \subseteq G$ and G' contains all the edges $\{s, t\}$ with $s, t \in V'$, then G' is an *induced subgraph* of G. We say that V' *induces* G' in G, and write $G' \equiv G[V']$.

A *clique* is a subgraph that is completely connected.

1.3 Centrality measures

Let $G = (V, E)$ be a simple, undirected and connected graph. Let s, t be two distinct vertices. The number of shortest s-t paths is denoted by $\sigma_{s,t}$. Let v be a vertex. The number of shortest s-t paths that contain v is denoted by $\sigma_{s,t}(v)$. By this definition, we have $\sigma_{s,t} = \sigma_{s,t}(s) = \sigma_{s,t}(t)$.

Let C be a set of vertices. The number of shortest s-t paths that contain no vertices from C is denoted by $\sigma^C_{s,t}$. Similarly, the number of shortest s-t paths that contain a vertex v and that share no vertices with C is denoted by $\sigma^C_{s,t}(v)$. Note that if every shortest s-t path shares a vertex with C (i. e. $\mathrm{dist}_{G[C]}(s,t) > \mathrm{dist}_G(s,t)$), then $\sigma^C_{s,t} = 0$.

The Betweenness Centrality of a vertex v is a positive real number proportional to the amount of shortest paths between pairs of distinct vertices which contain v.

Definition 1.1 (Dolev et al. [13]). *The Betweenness Centrality* (BC) *of vertex v is defined as*

$$\mathrm{BC}(v) \equiv \sum_{\substack{s,t \in V \\ s \neq t}} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}}$$

We assume that communication in G occurs between all pairs of vertices with the same probability. We also assume that when two vertices communicate, all shortest paths between them have the same probability of being used. Under these assumptions, the Betweenness Centrality of a vertex v is proportional to the probability that v detects communication between all pairs of distinct vertices in G.

The Group Betweenness Centrality measure extends the notion of Betweenness Centrality from a single vertex to groups of vertices. Let C be a set (or group) of vertices. The number of shortest s-t paths with at least one vertex in common with C is denoted by $\sigma_{s,t}(C)$.

Definition 1.2 (Dolev et al. [13]). *The* Group Betweenness Centrality (GBC) *of a set $C \subseteq V$ is defined as*

$$\mathrm{GBC}(C) \equiv \sum_{\substack{s,t \in V \\ s \neq t}} \frac{\sigma_{s,t}(C)}{\sigma_{s,t}}$$

Due to our assumptions regarding communication between vertices, GBC(C) is proportional to the probability that C detects communication between any pair of distinct vertices in G.

1.4 Computational complexity

We use the notation given by Arora and Barak [25].

A natural decision problem, such as "Is there a clique on at least k vertices in G?" can be represented by a boolean function $f\colon \{0,1\}^* \to \{0,1\}$ mapping an input string to a single bit of output. For this problem, the input string will encode the graph G, and function f will return 1 if a clique on at least k vertices exists in G or 0 otherwise. Thus a decision problem $L \subseteq \{0,1\}^*$ can be defined as $\{x\colon f(x) = 1\}$. The computational problem of deciding the language L is identical to computing f. In other words, the computational complexity of a problem L is the complexity of deciding whether an instance of the problem is a yes-instance.

The set of all decision problems that can be decided deterministically in bounded time is contained in the class DTIME, which is defined as follows

Definition 1.3 (The class DTIME, Definition 1.12 [25]). *Let $T\colon \mathbb{N} \to \mathbb{N}$ be some function. A language L is in DTIME($T(n)$) if and only if there is a Turing machine that runs in time $c \cdot T(n)$ for some constant $c > 0$ and decides L.*

To capture the notion of efficient computation the class P was introduced.

Definition 1.4 (The class P, Definition 1.13 [25])**.**

$$P \equiv \bigcup_{c \geq 1} DTIME(n^c)$$

To capture the notion of problems whose solution can be efficiently verified the complexity class NP was introduced[1].

Definition 1.5 (The class NP, Definition 2.1 [25])**.** *A language $L \subseteq \{0,1\}^*$ is in NP if there exists a polynomial $p \colon \mathbb{N} \to \mathbb{N}$ and a polynomial-time Turing machine M (called the verifier for L) such that for every $x \in \{0,1\}^*$,*

$$x \in L \iff \exists u \in 0, 1^{p(|x|)} \text{ such that } M(x, u) = 1$$

To be able to compare the complexity of two different problems, the concept of polynomial-time reductions was introduced.

Definition 1.6 (Reduction, Definition 2.7 [25])**.** *A language L is polynomial-time reducible to a language L' if there is a polynomial-time computable function $f \colon \{0,1\}^* \to \{0,1\}^*$ such that for every $x \in \{0,1\}^*, x \in L \iff f(x) \in L'$.*

We say that L' is NP-hard if every language L in NP is reducible to L'. We say that L' is NP-complete if L' is NP-hard and L' is in NP. No polynomial time algorithm for any NP-hard problem has been devised, and it is not known whether $\mathsf{P} \neq \mathsf{NP}$. Thus showing that a problem is NP-hard (usually by reducing from a problem known to be NP-hard) indicates that it is unlikely that a polynomial time algorithm for said problem exists.

A natural measurement of the complexity of an algorithm is the number of operations required to execute it. As we are usually not interested in the precise number of operations, it is useful to ignore low-level details and focus on the dominating sources of complexity. This is achieved using the *Big-Oh notation*

Definition 1.7 (Big-Oh notation, Definition 0.2 [25])**.** *Let f, g be two functions from $\mathbb{N}$ to $\mathbb{N}$. We say that*

- *$f = \mathcal{O}(g)$ if there is a constant c such that $f(n) \leq c \cdot g(n)$ for every sufficiently large n, and*
- *$f = o(g)$ if for every $\epsilon > 0$, $f(n) \leq \epsilon \cdot g(n)$ for every sufficiently large n.*

1.5 Parameterized complexity

We use the notation given by Cygan et al. [23].

The underlying idea behind a parameterized approach to a problem consists in expressing complexity as a function of not only the size of the input instance, but also of a parameter providing additional information. We call such a problem a *parameterized problem.* Formally, a parameterized problem is a language $L \subseteq \Sigma^* \times \Sigma^*$, where Σ is a fixed, finite alphabet. For an instance $(x, k) \in \Sigma^* \times \Sigma^*$, k is called the parameter. Note that as k can be either a single natural number or a more complex structure.

When solving an instance of a parameterized problem, it is often the case that the parameter k is small in comparison to the size of the instance. One way to make a problem more practically solvable (or *tractable*) is by finding an algorithm that solves an instance of problem L in time polynomial on the size of the instance for any fixed $|k|$. This notion is formalized in the concept of *fixed-parameter tractability.*

[1]The class of NP problems was originally defined as the class of problems which can be solved by a non-deterministic Turing machine in polynomial time. This is discussed in more detail by Arora and Barak [25] in section 2.1.2.

Definition 1.8 (FPT)**.** *A parameterized problem L is called* fixed-parameter tractable (FPT) *if there exists an algorithm $\mathcal{A}$, a computable function $f\colon \mathbb{N} \to \mathbb{N}$, and a constant c such that, given $(x, k) \in \Sigma^* \times \mathbb{N}$, $\mathcal{A}$ correctly decides whether $(x, k) \in L$ in time bounded by $f(k) \cdot |(x, k)|^c$.*

The algorithm $\mathcal{A}$ is then called *fixed-parameter tractable.* The complexity class containing all fixed-parameter tractable problems is called FPT.

Less efficient algorithms, yet also running in polynomial time for a fixed parameter, are formalized as *slice-wise polynomial* algorithms.

Definition 1.9 (Slice-wise polynomial)**.** *A parameterized problem L is called* slice-wise polynomial *if there exists an algorithm $\mathcal{A}$, two computable functions $f\colon \mathbb{N} \to \mathbb{N}$, $g\colon \mathbb{N} \to \mathbb{N}$, and a constant c such that, given $(x, k) \in \Sigma^* \times \mathbb{N}$, $\mathcal{A}$ correctly decides whether $(x, k) \in L$ in time bounded by $f(k) \cdot |(x, k)|^{g(k)}$.*

The complexity class containing all slice-wise polynomial problems is called XP.

As with polynomial reductions between decision problems, a parameterized problem can be reduced to another parameterized problem by means of a *parameterized reduction.*

Definition 1.10. *Let A, B be two parameterized problems. A* parameterized reduction *from A to B is an algorithm that, given an instance (x, k) of A, outputs an instance (x', k') of B such that*

1. *(x, k) is a yes-instance of A if and only if (x', k') is a yes instance of B,*
2. *$k' \leq g(k)$ for some computable function g, and*
3. *the running time is $f(k) \cdot |x|^{\mathcal{O}(1)}$.*

Cygan et al. [23] proves in Theorem 13.2 that if there exists a parameterized reduction from A to B and B is in FPT, then A is also in FPT.

Finding suitable parameters so that a parameterized problem is shown to be in FPT is the ideal outcome of a multivariate analysis. Although the running time of slice-wise polynomial algorithms is polynomial for a fixed parameter, in practice such algorithms are intractable for even small values of the parameter. Some parameterized problems, however, seem to be more complex than others. As an analogue to the NP class, the W hierarchy was introduced by Downey and Fellows [26] in an attempt to classify parameterized problems according to their hardness. Thus tractability results are complemented by an intractability theory. Because it is unlikely that FPT = W[1], showing that a parameterized problem is hard for any level of the W hierarchy can be understood as evidence that there is no FPT algorithm for said problem.

As with classical complexity, reductions between parameterized problems (which are not necessarily equivalent to polynomial reductions) can be used to show that a problem is at least as hard as some other parameterized problem. This is a useful tool for proofs of hardness and also for establishing lower bounds of computational complexity. Of particular interest is the so called *Exponential-Time Hypothesis* (ETH). It provides a lower bound for the complexity of the Satisfiability problem. Since it is believed that ETH holds, parameterized reductions can be used to give lower bounds on the complexity of parameterized problems, assuming ETH. This is discussed in detail in Chapter 14 of Cygan et al. [23].

Cygan et al. [23] gives a definition of the W-hierarchy and explains the subject of fixed-parameter intractability in Chapter 13. Although of great theoretical importance, our concern with the W-hierarchy is mostly practical, as it is useful to know whether it is unlikely that Maximum Betweenness Centrality is in FPT when parameterized by a particular parameter.

Chapter 2

Algorithms for the computation of Group Betweenness Centrality

We discuss existing algorithms for the computation of Group Betweenness Centrality. In Section 2.2, we present a new algorithm for computing the Group Betweenness Centrality of a group of size k that runs in $\mathcal{O}(kn^2)$ time, after $\mathcal{O}(nm)$ time preprocessing.

2.1 Existing algorithms

2.1.1 Combinatorial shortest path counting

Everett and Borgatti [12] introduced the concept of Group Betweenness Centrality. Along with providing a formal definition, they describe an $\mathcal{O}(nm)$ time algorithm for computing the GBC of a set C of arbitrary size.

The idea of the algorithm is the following. Let C be a set whose GBC we wish to calculate. Instead of counting the number of shortest paths that have vertices in common with C, we count the number of shortest paths that share no vertices with C, and subtract it from the total number of shortest paths. Hence we can calculate GBC(C) as

$$\mathrm{GBC}(C) = \sum_{\substack{s,t\in V\\ s\neq t}} \frac{\sigma_{s,t} - \sigma^C_{s,t}}{\sigma_{s,t}} \tag{2.1}$$

Because Eq. (2.1) differs from the definition of GBC only by replacing $\sigma_{s,t}(C)$ with $\sigma_{s,t} - \sigma^C_{s,t}$, proving the correctness of this algorithm is equivalent to proving the following lemma.

Lemma 2.1. *Let G be a connected graph. For all sets $C \subseteq V$ we have $\sigma_{s,t}(C) = \sigma_{s,t} - \sigma^C_{s,t}$.*

Proof. By definition, $\sigma^C_{s,t}$ is equal to the number of shortest s-t paths not covered by C. The number of shortest s-t paths covered by C is equal to $\sigma_{s,t}(C)$. As every shortest path between any pair of s, t vertices is either covered or not covered by C, we have $\sigma_{s,t} = \sigma_{s,t}(C) + \sigma^C_{s,t}$, or equivalently $\sigma_{s,t}(C) = \sigma_{s,t} - \sigma^C_{s,t}$. □

Finally, the running time of the algorithm is dominated by the time needed for calculating $\sigma_{s,t}$ and $\sigma^C_{s,t}$. Brandes [18] describes an algorithm (Algorithm 1, Lemma 3) that calculates $\sigma_{s,t}$ for all $s, t \in V$ in $\mathcal{O}(nm)$ time. Because calculating $\sigma^C_{s,t}$ for all $s, t \in V$ can be done by using the same algorithm on a graph $G' = G[V \setminus C]$, and G' contains no more vertices and edges than G, determining the value of GBC(C) using Eq. (2.1) can be done in $\mathcal{O}(nm)$ time.

Algorithm 1 Group Betweenness Centrality

Input: graph G, set of vertices C

Output: GBC(C)

```
calculate σ_{s,t} for all s, t ∈ V
calculate σ^C_{s,t} for all s, t ∈ V
GBC(C) ← 0
for every distinct s, t in V do
    GBC(C) ← GBC(C) + (σ_{s,t} − σ^M_{s,t}) / σ_{s,t}
end for
output GBC(C)
```

On a practical note, calculating $\sigma^M_{s,t}$ can be done by running the algorithm described by Brandes [18] on G while disregarding vertices from M and edges adjacent to M. If this approach is taken, the distance between two s, t vertices may increase (when every shortest s-t path shares a vertex with M), and in that case $\sigma^M_{s,t}$ should be equal to 0.

Note that the running time of Algorithm 1 is not dependant on the size of the group whose GBC is to be determined. In the following section we describe the algorithm given by Puzis et al. [19], whose running time, after preprocessing, depends only on the size of the input group.

2.1.2 A fast algorithm

Puzis et al. [19] introduced an algorithm that calculates the GBC of a group of size k in $\mathcal{O}(k^3)$ time, after preprocessing done in $\mathcal{O}(nm)$ time.

In the rest of this section we use the definition of GBC used by Puzis et al. [19]

$$\mathrm{GBC}(C) \equiv \sum_{\substack{s,t \in V \setminus C \\ s \neq t}} \frac{\sigma_{s,t}(C)}{\sigma_{s,t}} \tag{2.2}$$

Note that if shortest paths whose endpoints are in C are considered, GBC(C) increases by exactly $|C|^2 - |C|$ and $2\,|C|\,(n - |C|)$ due to shortest paths with two endpoints and one endpoint in C, respectively. Hence to calculate GBC(C) as defined by Dolev et al. [13] it suffices to calculate $\mathrm{GBC}(C) + |C|\,(2n - |C| - 1)$ using the same definition for GBC as Puzis et al. [19].

Before proceeding with an explanation of the algorithm of Puzis et al. [19] we introduce additional notation.

2.1.2.1 Additional notation

The Path Betweenness Centrality of an ordered set C is proportional to the number of shortest paths that contain all vertices from C in order.

Definition 2.2 (Path Betweenness Centrality)**.** *Let $\widetilde{\sigma}_{s,t}(C)$ be the number of shortest s-t paths that contain the vertices from C in order. Path Betweenness Centrality is defined as*

$$\widetilde{\mathrm{BC}}(C) \equiv \sum_{\substack{s,t \in V \\ s \neq t}} \frac{\widetilde{\sigma}_{s,t}(C)}{\sigma_{s,t}}$$

$\widetilde{\mathrm{BC}}^M(C)$ denotes the Path Betweenness Centrality of a set C when paths that share vertices with M are disregarded.

We denote the Path Betweenness Centrality of a set $C = (v_1, \ldots, v_k)$ simply by $\widetilde{\mathrm{BC}}(v_1, \ldots, v_k)$. The following two definitions capture the notion of *dependency*.

Definition 2.3 (Pair-dependency [18]). *The pair dependency of vertices s and t on vertex v is defined as the ratio of shortest s-t paths that contain v, and is denoted by*

$$\delta_{st}(v) \equiv \frac{\sigma_{s,t}(v)}{\sigma_{s,t}}$$

The dependency of a vertex on another vertex indicates the degree to which the latter contributes to the centrality of the former.

Definition 2.4 (Dependency [18]). *The dependency of vertex s on vertex v is denoted by*

$$\delta_{s\bullet}(v) \equiv \sum_{t\in V} \delta_{st}(v)$$

2.1.2.2 Description of the algorithm

The idea of the algorithm given by Puzis et al. [19] is the following. Let C be a set whose GBC is known. For any $v \notin C$ it is easy to see that

$$\mathrm{GBC}(C \cup \{v\}) = \mathrm{GBC}(C) + (\mathrm{GBC}(C \cup \{v\}) - \mathrm{GBC}(C)) \tag{2.3}$$

The term $\mathrm{GBC}(C \cup \{v\}) - \mathrm{GBC}(C)$ corresponds to the contribution of v to $\mathrm{GBC}(C \cup \{v\})$[1]. The key observation is that this contribution is equal to the total fraction of shortest paths between all pairs of vertices that traverse v, when only the shortest paths that contain no vertices from C are considered. This is equivalent to $\widetilde{\mathrm{BC}}^C(v)$, the Path Betweenness Centrality of v when vertices from C are disregarded. Hence the Group Betweenness Centrality of a set C with k elements can be calculated using the following equation.

$$\mathrm{GBC}(C) = \widetilde{\mathrm{BC}}^{\emptyset}(C_1) + \widetilde{\mathrm{BC}}^{\{C_1\}}(C_2) + \ldots + \widetilde{\mathrm{BC}}^{\{C_1,\ldots,C_{k-1}\}}(C_k) \tag{2.4}$$

The algorithmic improvement comes from observing that updating the $\widetilde{\mathrm{BC}}$ of all unaccounted vertices from C after the contribution of a vertex has been considered can be done in $\mathcal{O}(k^2)$ time. Since calculating the first term of the sum in Eq. (2.4) can be done in $\mathcal{O}(nm)$ time, and determining the value of each following term can be done in $\mathcal{O}(k^2)$ time, this approach results in the total $\mathcal{O}(nm + k^3)$ running time.

Let C be a set with k vertices whose GBC we wish to determine. Let M be the set of vertices whose contribution has already been considered. We show that computing $\widetilde{\mathrm{BC}}^{M\cup\{v\}}(s)$ and $\sigma_{s,t}^{M\cup\{v\}}$ for all $s, t \in C \setminus M$ can be done in $\mathcal{O}(k^2)$ time, assuming that $\widetilde{\mathrm{BC}}^M(s,t)$ and $\sigma_{s,t}^M$ is known for all $s, t \in C \setminus M$ beforehand.

Let v be the vertex whose contribution to $\mathrm{GBC}(M \cup \{v\})$ we wish to calculate, and let $s, t \in C \setminus M$ be a pair of vertices whose contribution to $\mathrm{GBC}(C)$ is yet to be considered.

We first consider the case when $s = t$. The value of $\widetilde{\mathrm{BC}}^{M\cup\{v\}}(s)$ decreases exactly by the contribution of all shortest paths that contain both s and v. The contribution of these paths is equal to $\widetilde{\mathrm{BC}}^M(s,v) + \widetilde{\mathrm{BC}}^M(v,s)$, and hence the new $\widetilde{\mathrm{BC}}$ of s is equal to

$$\widetilde{\mathrm{BC}}^{M\cup\{v\}}(s) = \widetilde{\mathrm{BC}}^M(s) - \left(\widetilde{\mathrm{BC}}^M(v,s) + \widetilde{\mathrm{BC}}^M(s,v)\right) \tag{2.5}$$

[1]The term "contribution" is used loosely throughout this chapter, with the goal of aiding an intuitive understanding of the described algorithms. Although in a sense $\mathrm{GBC}(C \cup \{v\}) - \mathrm{GBC}(C)$ can indeed be seen as the contribution of v to $\mathrm{GBC}(C \cup \{v\})$, if such an approach were taken then summing the contribution of all vertices of a set M will result in a value smaller than $\mathrm{GBC}(M)$, since some shortest paths contain more than one vertex from M, assuming that M contains at least 2 vertices.

As we assume that both $\widetilde{\mathrm{BC}}^M(s,v)$ and $\widetilde{\mathrm{BC}}^M(v,s)$ were already calculated, this operation can be done in $\mathcal{O}(1)$ time for each vertex, and in $\mathcal{O}(k)$ time for all vertices from $C \setminus M$.

We now assume that s and t are distinct. As earlier, considering the contribution of v decreases the potential contribution of shortest paths containing s and then t by the contribution of shortest paths that contain the vertices s, t and v, where s is before t. If any such shortest path exists, then every path will contain the vertices s, t and v in one of three orders:

$$\widetilde{\mathrm{BC}}^{M\cup\{v\}}(s,t) = \widetilde{\mathrm{BC}}^M(s,t) - \begin{cases} \widetilde{\mathrm{BC}}^M(v,s,t) & \mathrm{dist}_G(v,t) = \mathrm{dist}_G(v,s) + \mathrm{dist}_G(s,t) \\ \widetilde{\mathrm{BC}}^M(s,v,t) & \mathrm{dist}_G(s,t) = \mathrm{dist}_G(s,v) + \mathrm{dist}_G(v,t) \\ \widetilde{\mathrm{BC}}^M(s,t,v) & \mathrm{dist}_G(s,v) = \mathrm{dist}_G(s,t) + \mathrm{dist}_G(t,v) \end{cases} \tag{2.6}$$

Puzis et al. [19] explains how the $\widetilde{\mathrm{BC}}$ of a group of three vertices can be calculated using only the $\widetilde{\mathrm{BC}}$ of pairs of vertices.

$$\widetilde{\mathrm{BC}}(s,t,v) = \frac{\sigma^M_{s,v}(t)}{\sigma^M_{s,v}} \cdot \widetilde{\mathrm{BC}}^M(s,v) \tag{2.7}$$

As we assume that $\sigma^M_{s,t}$ and $\widetilde{\mathrm{BC}}^M(s,t)$ are known for all $s,t \in C \setminus M$, in order to calculate $\widetilde{\mathrm{BC}}(s,t,v)$ we only need to determine the value of $\sigma^M_{s,v}(t)$. Puzis et al. [19] explains how this can be done in $\mathcal{O}(1)$ time.

Lemma 2.5. *For all $s,t,v \in V$ and $M \subseteq V$ we have $\sigma^M_{s,t}(v) = \sigma^M_{s,v}\sigma^M_{v,t}$.*

Proof. Every shortest s-t path that contains v is composed of two shortest s-v and v-t paths that have only vertex v in common. If the two paths had any additional vertices in common, a shorter s-t path could be constructed, which would contradict the assumption that the original path was a shortest path. ☐

Thus $\widetilde{\mathrm{BC}}^{M\cup\{v\}}(s,t)$ can be calculated in $\mathcal{O}(1)$ time for each s,t pair, assuming that $\sigma^M_{s,t}$ is known for all $s,t \in C \setminus M$.

Finally, we can use Lemma 2.5 to show how $\sigma^{M\cup\{v\}}_{s,t}$ can be calculated in $\mathcal{O}(1)$ time for each $s,t \in C \setminus M$ pair.

$$\sigma^{M\cup\{v\}}_{s,t} = \sigma^M_{s,t} - \sigma^M_{s,t}(v) = \sigma^M_{s,t} - \sigma^M_{s,v}\sigma^M_{v,t} \tag{2.8}$$

To summarize, we have shown that $\widetilde{\mathrm{BC}}^{M\cup\{v\}}(s,t)$ and $\sigma^{M\cup\{v\}}_{s,t}$ can be calculated for each pair of vertices in $\mathcal{O}(1)$ time, assuming that $\widetilde{\mathrm{BC}}^M(s,t)$ and $\sigma^M_{s,t}$ are known beforehand for all $s,t \in C \setminus M$. Because there are k^2 pairs of vertices from C, updating the pair Path Betweenness Centrality of all pairs of vertices from C after considering the contribution of each vertex from C to GBC(C) can be done in $\mathcal{O}(k^2)$. As there are k such steps, this gives a running time of $\mathcal{O}(k^3)$ for calculating the GBC of a group of size k, assuming that $\widetilde{\mathrm{BC}}^\emptyset(s,t)$, $\mathrm{dist}_G(s,t)$ and $\sigma^\emptyset_{s,t}$ are known for every pair $s,t \in C$. Calculating $\mathrm{dist}_G(s,t)$ and $\sigma^\emptyset_{s,t}$ for all s,t can be done in $\mathcal{O}(nm)$ time using breadth first search. We can calculate the Path Betweenness Centrality of a pair of vertices using the following equation

$$\widetilde{\mathrm{BC}}^\emptyset(s,t) = \sum_{v\in V} \delta_{v\bullet}(t)\frac{\sigma^\emptyset_{v,t}(s)}{\sigma^\emptyset_{v,t}} \tag{2.9}$$

The value of $\delta_{v\bullet}(t)$ can be calculated for all $v,t \in V$ in $\mathcal{O}(nm)$ time [18]. This completes the required preprocessing, with a running time of $\mathcal{O}(nm)$.

A result of the previous observations is that the algorithm described by Puzis et al. [19] calculates the GBC of a group of size k in $\mathcal{O}(nm + k^3)$ time. Puzis et al. [19] provides pseudocode for their algorithm. They also compare the efficiency of their implementation to that of previous state of the art algorithms for calculating GBC, thus practically showing that—even for a small amount of groups whose GBC is to be calculated—the algorithm they describe outperforms previous algorithms.

2.2 A new algorithm

We present an algorithm for calculating the GBC of a group of size k in $\mathcal{O}(nm + kn^2)$ time, if the number of shortest paths between every pair of vertices can be determined in $\mathcal{O}(nm)$ time.

The intuition behind our approach is similar to that of Puzis et al. [19]. We calculate the GBC of a set C containing k vertices by gradually considering the contribution of each vertex. The correctness and the running time of the algorithm is given by the following lemma.

Lemma 2.6. *Let G be a connected graph and $C \subseteq V$. For any vertex $v \notin C$, if $\mathrm{GBC}(C)$ has already been calculated and $\sigma^C_{s,t}$ is known for all $s,t \in V$, then calculating $\mathrm{GBC}(C \cup \{v\})$ and $\sigma^{C\cup\{v\}}_{s,t}$ for all $s,t \in V$ can be done in $\mathcal{O}(n^2)$ time.*

Proof. Every shortest path covered by $C \cup \{v\}$ either has vertices in common with C (and potentially also contains v) or contains only v and shares no vertices with C. Formally, we have

$$\mathrm{GBC}(C \cup \{v\}) = \mathrm{GBC}(C) + \sum_{\substack{s,t\in V\\ s\neq t}} \frac{\sigma^C_{s,t}(v)}{\sigma_{s,t}} \tag{2.10}$$

Instead of calculating $\sigma^C_{s,t}(v)$ directly, we make use of the assumption that $\sigma^C_{s,t}$ is known for all $s,t \in V$, and calculate $\sigma^C_{s,t}(v)$ as $\sigma^C_{s,v}\sigma^C_{v,t}$ (Lemma 2.5). As there are $\mathcal{O}(n^2)$ pairs of vertices, calculating the contribution of a single vertex to the GBC of a group can be done in $\mathcal{O}(n^2)$ time, assuming that $\mathrm{GBC}(C)$ is known.

Finally, calculating $\sigma^{C\cup\{v\}}_{s,t}$ for all $s,t \in V$ can be done in $\mathcal{O}(n^2)$ time using Eq. (2.8). □

Theorem 2.7. *Let C be a set of size k. There is an algorithm that calculates $\mathrm{GBC}(C)$ in $\mathcal{O}(nm + kn^2)$ time.*

Proof. Let M be the set of vertices whose contribution to $\mathrm{GBC}(C)$ has already been considered.

At each step of the algorithm we add a vertex v from C to M, and calculate the increment to $\mathrm{GBC}(M)$ corresponding to paths that contain v but share no vertices with C. The algorithm terminates when all vertices from C have been added to M, and $\mathrm{GBC}(M)$ is equal to $\mathrm{GBC}(C)$.

As explained in Lemma 2.6 calculating $\mathrm{GBC}(M \cup \{v\})$ and $\sigma^{M\cup\{v\}}_{s,t}$ for all $s,t \in V$ can be done in $\mathcal{O}(n^2)$ time, assuming that $\mathrm{GBC}(M)$ has already been calculated and that $\sigma^M_{s,t}$ is known for all $s,t \in V$. Since k such steps are necessary for calculating the contribution of all vertices in C to $\mathrm{GBC}(C)$, the running time of the algorithm is $\mathcal{O}(kn^2)$.

Although at each step of the algorithm $\sigma^{M\cup\{v\}}_{s,t}$ is calculated for all $s,t \in V$, and hence can be used in the following step, $\sigma^{\emptyset}_{s,t}$ must be calculated directly. Brandes [18] describes how the number of shortest paths between all pairs of vertices can be calculated in $\mathcal{O}(nm)$ time. This results in the total running time of $\mathcal{O}(nm + kn^2)$. □

Note that for an actual implementation only three matrices are needed to store the number of shortest paths between pairs of vertices. One for the division on Line 8 and two for the number of paths when vertices from M and M' are disregarded, as $\sigma^{M'}_{\bullet,\bullet}$ becomes $\sigma^{M}_{\bullet,\bullet}$ after Line 11.

Algorithm 2 Group Betweenness Centrality

Input: connected graph G, set of vertices C
Output: GBC(C)

$M \leftarrow \emptyset$
$\text{GBC}(M) \leftarrow 0$
calculate $\sigma_{s,t}^{M}$ for all $s, t \in V$
for every v in C **do**
    $M' \leftarrow M \cup \{v\}$
    $\text{GBC}(M') \leftarrow \text{GBC}(M)$
    for every s, t in $V \setminus M$ **do**
        $\text{GBC}(M') \leftarrow \text{GBC}(M') + \frac{\sigma_{s,v}^{M}\sigma_{v,t}^{M}}{\sigma_{s,t}}$
        $\sigma_{s,t}^{M'} \leftarrow \sigma_{s,t}^{M} - \sigma_{s,v}^{M}\sigma_{v,t}^{M}$
    end for
    $M \leftarrow M'$
    $\text{GBC}(M) \leftarrow \text{GBC}(M')$
end for
output GBC(M)

Chapter 3

Maximum Betweenness Centrality

In this chapter, we define our problem of interest—Maximum Betweenness Centrality. Because Maximum Betweenness Centrality is closely related to Group Betweenness Centrality, some instances of Maximum Betweenness Centrality will be yes- or no-instances depending on the used definition of Group Betweenness Centrality. In Section 3.2, we discuss the existing definitions of Group Betweenness Centrality, and the effect they have in the results of algorithms for the Maximum Betweenness Centrality problem.

After showing that Maximum Betweenness Centrality is NP-complete in Section 3.3, we conclude this chapter by presenting two algorithms for Maximum Betweenness Centrality, with $\mathcal{O}(nm + b^3 n^b)$ and $\mathcal{O}(nm + bn^{b+2})$ running times, respectively, where b is the available budget.

3.1 Problem definition

The Maximum Betweenness Centrality problem consists in finding a group with maximal GBC, such that the size of the group is at most equal to a budget. We formulate it as a decision problem.

Maximum Betweenness Centrality

Input: A connected graph G, a budget b and a threshold t.

Task: Decide whether a set C of vertices exists such that its size does not exceed the budget b, and $\mathrm{GBC}(C)$ is at least equal to the threshold t.

A natural generalization of the Maximum Betweenness Centrality problem is the Weighted Maximum Betweenness Centrality problem, defined as follows

Weighted Maximum Betweenness Centrality [14]

Input: A connected graph G, a cost function $c\colon V \to \mathbb{N}$, a budget b and a threshold t

Task: Decide whether a set C of vertices exists such that the sum of the costs (or weights) of its vertices does not exceed the budget b, and $\mathrm{GBC}(C)$ is at least equal to the threshold t.

It is easy to verify that all achieved hardness results carry over directly from Maximum Betweenness Centrality to its weighted variant, as every instance of Maximum Betweenness Centrality can be seen as an instance of Weighted Maximum Betweenness Centrality where all vertices have unit weight.

3.2 Alternative definitions

The definitions of Group Betweenness Centrality found in the existing literature have subtle differences. Table 3.1 lists a selection of existing definitions.

Table 3.1 Comparison of existing definitions of Group Betweenness Centrality.

Source	GBC(C)	**Considered pairs of vertices**
Everett and Borgatti [12]	$\sum_{\substack{s,t \in V \\ s<t \\ s,t \notin C}} \frac{\sigma_{s,t}(C)}{\sigma_{s,t}}$	Unordered pairs of distinct vertices from $V \setminus C$
Puzis et al. [14]	$\sum_{\substack{s,t \in V \\ s,t \notin C}} \frac{\sigma_{s,t}(C)}{\sigma_{s,t}}$	Ordered pairs of distinct vertices that share at least one vertex with C [1]
Dolev et al. [13]	$\sum_{\substack{s,t \in V \\ s \neq t}} \frac{\sigma_{s,t}(C)}{\sigma_{s,t}}$	Ordered pairs of distinct vertices
Veremyev et al. [16]	$\sum_{s,t \in V} \frac{\sigma_{s,t}(C)}{\sigma_{s,t}}$	All ordered pairs

Because deciding an instance of Maximum Betweenness Centrality involves finding a set whose GBC is at least equal to a threshold, different definitions of GBC can potentially lead to non-equivalent definitions of Maximum Betweenness Centrality.

From the mentioned papers, both Puzis et al. [19] and Veremyev et al. [16] acknowledge the discrepancy between their definitions and previous ones. Veremyev et al. [16] show how their definition of Group Betweenness Centrality leads to an increase of GBC by $|C|$ when compared to a definition of GBC differing only by ignoring v-v paths (like the GBC given by Dolev et al. [13]). Puzis et al. [19] explains that the GBC of a group C calculated using the

[1] Originally, the definition given by Puzis et al. [14] uses the phrase "traverses at least one vertex of C," instead of "shares at least one vertex with C." Note that, technically speaking, a definition of GBC that considers only shortest paths that *traverse* at least one vertex of C would be more faithfully calculated as

$$\sum_{\substack{s,t \in V \\ s \neq t}} \frac{\sigma_{s,t}(C \setminus \{s,t\})}{\sigma_{s,t}}$$

If the Maximum Betweenness Centrality problem uses this definition, there is no way to modify instances of Maximum Betweenness Centrality that use a different definition, since for any given set C there is no way of knowing the amount of shortest paths that traverse it, where traverse is understood as "goes through." Additionally, based on the comments by Puzis et al. [19], it is clear that the definition from Puzis et al. [14] considers only shortest paths with no endpoints in C, as opposed to shortest paths that contain vertices from C not only as endpoints.

definition by Puzis et al. [14] as opposed to that of Dolev et al. [13] will be smaller by exactly $(|C|^2 - |C|) + 2|C|(n - |C|) = |C|(2n - |C| - 1)$, which corresponds to the contribution of shortest paths with at least one endpoint in C. Thus, an instance $\mathcal{I}$ of Maximum Betweenness Centrality with Group Betweenness Centrality as defined by Dolev et al. [13] can be obtained from any other instance of Maximum Betweenness Centrality with a different definition of GBC by modifying the threshold. We use the definition by Dolev et al. [13] as reference, and denote by t the base threshold and by t' the modified threshold. For the definition by Everett and Borgatti [12], we have $t = 2(t' + b(2n - b - 1))$. The factor of 2 is due to the definition considering only unordered pairs. Thus, for the definition of GBC given by Puzis et al. [14] we have $t = t' + b(2n - b - 1)$. If the definition by Veremyev et al. [16] is used, we have $t = t' - b$, since the only additionally considered pairs of vertices are the b s-s pairs.

With regards to Weighted Maximum Betweenness Centrality, none of the definitions are equivalent to that given by Dolev et al. [13], assuming arbitrary weights. The problem is that there is no way of knowing the size of a solution for a given instance, since it may contain many expensive vertices or few cheaper ones. Therefore establishing a threshold can only be done when all vertices have either a fixed cost, or a cost exceeding the budget. For the same reason, none of the definitions are equivalent to that given by Veremyev et al. [16]. Only the definition by Everett and Borgatti [12] is equivalent to the definition by Puzis et al. [14], since both differ only by considering ordered or unordered shortest paths.

Although the previous discussion highlights subtle differences between existing definitions of GBC, and therefore in the resulting Maximum Betweenness Centrality problem (and its weighted variant), all of the results in this paper remain valid if a different definition for GBC is used instead of that by Dolev et al. [13].

3.3 Complexity

A formal proof of the NP-completeness of Maximum Betweenness Centrality has not been published, although Puzis et al. [14] hints that a proof of the NP-hardness of Maximum Betweenness Centrality can be done by reducing from the Vertex Cover problem, defined as follows.

Vertex Cover [23]

Input:	A connected graph G and an integer k.
Task:	Decide whether a set C of at most k vertices exists such that at least one endpoint of every edge in G is in C.

We fulfill this gap by showing that Maximum Betweenness Centrality is not only NP-hard, but additionally is also in NP, and is therefore NP-complete.

Theorem 3.1. *Maximum Betweenness Centrality is NP-complete.*

Proof. To show that Maximum Betweenness Centrality is NP-hard, we observe that every vertex cover C detects all communication between every pair of adjacent vertices, as at least one endpoint of every edge is in C. The threshold t of a Maximum Betweenness Centrality instance can then be established so that only instances that contain a group that detects all communication between every pair of vertices are yes-instances. Finally, a budget equal to k ensures that only groups that contain at most k vertices are considered.

Formally, we describe a polynomial reduction from the Vertex Cover problem, which is known to be NP-complete [27]. The reduction, as outlined by Puzis et al. [14], is straightforward. Let $\mathcal{I} = (G, k)$ be an instance of Vertex Cover. We define a new instance $\mathcal{J} = (G, b = k, t)$ of Maximum Betweenness Centrality, where $t = n^2 - n$ is equal to the number of pairs of distinct vertices. The threshold t ensures that only instances of Maximum Betweenness

Centrality containing groups which detect all communication between every pair of vertices are yes-instances.

We now show that $\mathcal{I}$ is a yes-instance if and only if $\mathcal{J}$ is a yes-instance.

If $\mathcal{I}$ is a yes-instance, than there is a set $C \subseteq V$ such that $|C| \leq b$ and C is a vertex cover. As C contains at least one endpoint of every edge, every shortest path between a pair of distinct vertices must contain at least one vertex from C. Otherwise an uncovered edge would exist in G, contradicting the assumption that C is a vertex cover. Because all shortest paths between every pair of distinct vertices share at least one vertex with C, it follows that $\text{GBC}(C) = n^2 - n$, and thus $\mathcal{J}$ is also a yes-instance.

Now let $\mathcal{J}$ be a yes-instance. Then a set C exists such that $|C| \leq b$ and $\text{GBC}(C) \geq t$. Pursuing a contradiction, we assume that C is not a vertex cover. Then there is at least one $\{s,t\}$ edge such that neither s nor t are in C. Clearly, the only shortest s-t path is the $\{s,t\}$ edge not covered by C, and $\sigma_{s,t}(C)$ is therefore equal to 0. Hence $\text{GBC}(C)$ is at most equal to $n^2 - n - 2$. As $t = n^2 - n \neq n^2 - n - 2$, we arrive at a contradiction with our assumption that $\text{GBC}(C) \geq t$. It follows that $\mathcal{I}$ must also be a yes-instance.

This completes the reduction from Vertex Cover to Maximum Betweenness Centrality. To show that the reduction is indeed polynomial it suffices to see that an instance of Maximum Betweenness Centrality corresponding to an instance of Vertex Cover can be constructed in polynomial time, as no additional vertices nor edges are introduced with respect to the original input graph.

To prove that Maximum Betweenness Centrality is NP-complete we must additionally show that it is in NP. Because a solution for a yes-instance is a subset of n, it can be employed as a certificate of polynomial length. The $\mathcal{O}(nm)$ algorithm given by Puzis et al. [19] can be used to verify that the GBC of a given solution exceeds the threshold. As every yes-instance has a certificate of polynomial size that can be verified in polynomial time, it follows that Maximum Betweenness Centrality is in NP.

To summarize, we have shown that Maximum Betweenness Centrality is NP-hard, and that it is even in NP. It follows that Maximum Betweenness Centrality is NP-complete. ☐

Note that if s-s shortest paths are considered (the definition of GBC by Veremyev et al. [16]), the threshold of every instance of Maximum Betweenness Centrality can be modified to obtain an equivalent instance. For the modified threshold we have $t' = t + b$, where the addition of b corresponds to the b s-s paths covered by the vertices of a solution of size b. Similarly, if shortest paths with endpoints in C are ignored (the definition of GBC given by Puzis et al. [14]), for the modified threshold we have $t' = t - (b^2 - b) - (2b(n - b))$.

Corollary 3.2. *Weighted Maximum Betweenness Centrality is NP-complete.*

3.4 A trivial algorithm

In this section, we describe a trivial algorithm for Maximum Betweenness Centrality. The algorithm calculates the GBC of every group containing b vertices.

Theorem 3.3. *Let $\mathcal{I} = (G, b, t)$ be an instance of Maximum Betweenness Centrality with budget b and threshold t. There is an algorithm that decides $\mathcal{I}$ in $\mathcal{O}(nm \cdot n^b)$ time.*

Proof. The idea is straightforward. Using the $\mathcal{O}(nm)$ algorithm for calculating the GBC of a group, we calculate the GBC of all $\mathcal{O}(n^b)$ groups of size b. $\mathcal{I}$ is a yes-instance if and only if a group C exists such that $\text{GBC}(C) \geq t$. ☐

We can attain the previous result because we assume that only subsets of size exactly b need to be considered. This is not so with regards to Weighted Maximum Betweenness Centrality. Let C be a set whose cost is below the budget, such that $\text{GBC}(C)$ is greater than the established threshold t. Now consider the set $C \cup \{v\}$. Although $\mathrm{GBC}(C \cup \{v\})$ is certainly greater than $\mathrm{GBC}(C)$, we have no way to ensure that the cost of the set $C \cup \{v\}$ remains below the budget. Hence it is necessary to calculate the GBC of every set of size *at most* b. Using the $\mathcal{O}(nm)$ algorithm for the calculation of GBC for this task yields an $\mathcal{O}(nm \cdot e^b n^b)$ time [2] algorithm for Weighted Maximum Betweenness Centrality parameterized by the budget.

Corollary 3.4. *There is an $\mathcal{O}(nm \cdot e^b n^b)$ time algorithm for Weighted Maximum Betweenness Centrality.*

In the following section, we introduce two slightly more sophisticated algorithms with asymptotically better running times.

3.5 Two algorithms

In this section, we discuss the algorithm for Maximum Betweenness Centrality given by Puzis et al. [14], and present a simpler algorithm with the same asymptotic running time. Although both algorithms are intractable for large or even medium-sized instances, in Section 4.1.1 we show that both algorithms are asymptotically optimal with regards to n.

Theorem 3.5 (Puzis et al. [14])**.** *Let $\mathcal{I} = (G, b, t)$ be an instance of Maximum Betweenness Centrality parameterized by the budget b. There is an algorithm that decides $\mathcal{I}$ in $\mathcal{O}(nm + b^3 n^b)$ time.*

Proof. We can decide instance $\mathcal{I}$ by calculating the GBC of every subset of V of size b. As the number of said groups is bounded up by $\mathcal{O}(n^b)$, using the algorithm of Puzis et al. [19] to calculate the GBC of each group of size b in $\mathcal{O}(b^3)$, after $\mathcal{O}(nm)$ preprocessing, leads to an $\mathcal{O}(nm + b^3 n^b)$ time algorithm for Maximum Betweenness Centrality. ☐

Corollary 3.6 (Puzis et al. [14])**.** *There is an $\mathcal{O}(nm + b^3 e^b n^b)$ time algorithm for Weighted Maximum Betweenness Centrality parameterized by the budget b.*

3.5.1 A new algorithm

We present a new $\mathcal{O}(nm + bn^{b+2})$ time algorithm for Maximum Betweenness Centrality. It is similar to that given by Puzis et al. [14], and it works by calculating the GBC of all groups of size b.

Theorem 3.7. *Let $\mathcal{I} = (G, b, t)$ be an instance of Maximum Betweenness Centrality parameterized by the budget b. There is an algorithm that decides $\mathcal{I}$ in $\mathcal{O}(nm + bn^{b+2})$ time.*

Proof. As with the previous theorem, we use the algorithm described in Section 2.2 to calculate the GBC of each of the $\mathcal{O}(n^b)$ groups of size b in $\mathcal{O}(bn^2)$ time, after $\mathcal{O}(nm)$ preprocessing. This results in an $\mathcal{O}(nm + bn^{b+2})$ time algorithm for Maximum Betweenness Centrality. ☐

Corollary 3.8. *There is an $\mathcal{O}(nm + be^b n^{b+2})$ time algorithm for Weighted Maximum Betweenness Centrality parameterized by the budget b.*

[2] We use the following bound for partial sums of binomial coefficients:

$$\sum_{i=0}^{b} \binom{n}{b} \leq \sum_{i=0}^{b} \frac{n^i}{i!} \leq \frac{n^b}{b^b} \cdot \sum_{i=0}^{\infty} \frac{b^i}{i!} \leq e^b n^b \tag{3.1}$$

Chapter 4

Natural parameters

A good starting point for a multivariate analysis of MAXIMUM BETWEENNESS CENTRALITY is determining whether it is tractable when parameterized by its natural parameters. In the first section, we show that MAXIMUM BETWEENNESS CENTRALITY parameterized by the budget is W[1]-hard, which can be seen as evidence that it is unlikely that it is in FPT. We complement this result in Section 4.1.1 with a conditional lower bound matching the running time of our algorithm from Theorem 3.7 based on the well-known ETH [23]. We conclude by discussing parameterization by the threshold.

4.1 Budget

Our approach is inspired by the one of Fink and Spoerhase [28]. We show that MAXIMUM BETWEENNESS CENTRALITY is W[1]-hard when parameterized by the budget by reducing from PARTIAL VERTEX COVER, which is defined as follows.

PARTIAL VERTEX COVER [27]

Input:	A connected graph G, integers k and ℓ.
Task:	Decide whether there exists a set of vertices C of size at most k such that at least ℓ edges of G are incident to at least one vertex of C.

PARTIAL VERTEX COVER is known to be W[1]-complete when parameterized by the size of the cover k [29].

Theorem 4.1. *MAXIMUM BETWEENNESS CENTRALITY is W[1]-hard when parameterized by the budget b.*

Proof. We are given an instance (G, k, ℓ) of PARTIAL VERTEX COVER, and construct an instance $(G', b = k, t = 2\ell d^2)$ of MAXIMUM BETWEENNESS CENTRALITY. The number d can be picked arbitrarily, as long as it is large enough in comparison to n. In Eq. (4.3) we show that d can be bounded up by $n^{\mathcal{O}(1)}$. The goal of the construction is to increase the centrality of vertices adjacent to edges from E, so that a proportion between the amount of covered edges by a set C and $\mathrm{GBC}(C)$ can be more easily established. We construct G' as follows.

1. Copy all vertices and edges from G into G'.

2. For every vertex $s \in V$ add d additional vertices $D_s = s_1, \ldots, s_d$ to V'.

3. Add edges to E' so that the set $D_s \cup \{s\}$ forms a clique.

4. For every edge $st \notin E$ add a vertex z_{st} to V'.

5. Connect every z_{st} vertex to every vertex of $D_s \cup D_t$.

The running time of the construction is given by the size of G'. The number of vertices $|V|$ is bounded up by $n + dn + m$, and the number of edges $|E|$ is bounded up by $m + d^2 n + 2md$. As adding vertices or edges can be done in $\mathcal{O}(1)$, and d can be bounded by $n^{\mathcal{O}(1)}$ (inequality (4.3)), the running time of the construction is $n^{\mathcal{O}(1)}$.

This completes the construction. An illustration is shown in Fig. 4.1.

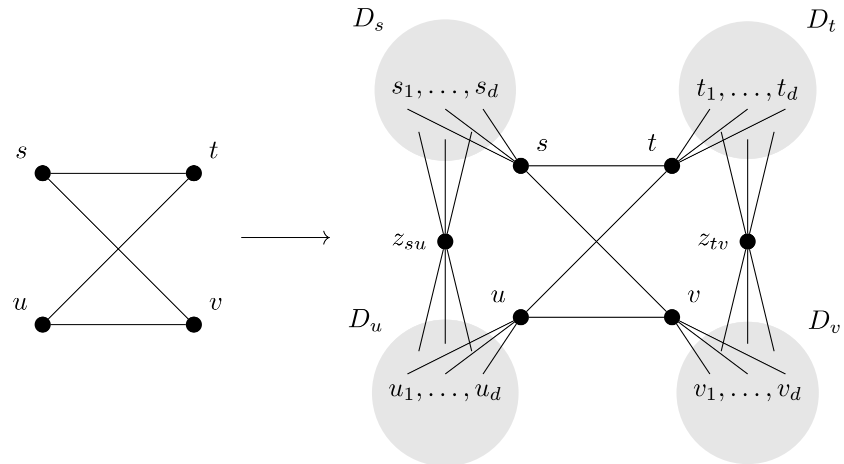

Figure 4.1 Construction used in the parameterized reduction from Partial Vertex Cover to Maximum Betweenness Centrality.

We have described a construction that runs in $n^{\mathcal{O}(1)}$ time and that preserves the size of the parameter—the size of the cover for Partial Vertex Cover and the budget for Maximum Betweenness Centrality. To prove that the construction constitutes a parameterized reduction we must additionally show that an instance of Partial Vertex Cover is a yes-instance if and only if the transformed instance of Maximum Betweenness Centrality is also a yes-instance.

Let $\mathcal{I} = (G, k, \ell)$ be an instance of Partial Vertex Cover, and $\mathcal{J} = (G', b = k, t = 2d^2\ell)$ be the instance of Maximum Betweenness Centrality corresponding to $\mathcal{I}$.

The threshold t of $\mathcal{J}$ is equal to the minimal GBC of a set covering ℓ edges. Let C be a set containing b vertices, such that it covers at least ℓ edges from E'. Every $\{s, t\}$ edge covered by C corresponds to $2d^2$ unique shortest paths between vertices added as copies of the s and t vertices, respectively. Hence GBC(C) is at least equal to $2d^2\ell$. It follows that if $\mathcal{I}$ is a yes-instance, then $\mathcal{J}$ is also a yes-instance.

We now show that if $\mathcal{J}$ is a yes-instance, then $\mathcal{I}$ is also a yes-instance. It is easy to verify that this is equivalent to showing that if $\mathcal{I}$ is a no-instance, then $\mathcal{J}$ is also a no-instance. Let $C \subseteq V'$ be a set containing b vertices. C covers at most ℓ edges, as otherwise $\mathcal{I}$ would be a yes-instance. We wish to show that the GBC of C is always strictly smaller than the threshold t. As considering the combined contribution of all vertices leads to a weak bound, we partition V' into three groups, and give upper bounds for the contribution of each of the 6 resulting pairs of groups. V denotes the original set of vertices, Z denotes the vertices that correspond to two unadjacent vertices, and D denotes the set of vertices added as copies of vertices from V. It is easy to verify that every vertex in V' is a member of exactly one of the previous groups. Table 4.1 lists upper bounds for the number of pairs from each combination of two groups of vertices.

Table 4.1 Upper bounds for number of pairs between vertices from V, Z and D.

Pairs	Count
V-V	$n^2 - n$
V-Z	$2nm$
V-D	$2n(nd)$
Z-Z	$m^2 - m$
Z-D	$2m(nd)$
D-D	$d^2n^2 - dn$

Without any further assumptions, it is clear that the upper bound for GBC(C) obtained by adding the bounds given in Table 4.1 is still not strong enough, as the amount of D-D pairs easily exceeds $2d^2\ell$. We strengthen this bound by observing that the contribution of D-D pairs to GBC(C) is at most equal to $2d^2(\ell - 1)$.

For any vertex $v \in C$ that is not in V, the contribution of D-D pairs to GBC(C) is maximized when v is picked from Z, and is at most equal to $2d^2$. Because picking the endpoint of an uncovered edge contributes with at least $2d^2$ to GBC(C), vertices from Z can be replaced with vertices from V without decrementing GBC(C). Note that there are always enough uncovered edges to allow for this modification. If this were not the case, C would cover all edges, and as $l - 1 < m$, this would contradict the assumption that C covers at most $l - 1$ edges. Hence we can assume that C is a subset of V. As C covers at most $\ell - 1$ edges, the contribution of D-D pairs to GBC(C) is at most equal to $2d^2(\ell - 1)$.

With this stronger bound for the contribution of D-D pairs we can establish an upper bound for GBC(C).

$$\begin{aligned} \mathrm{GBC}(C) &\leq (n^2 - n) + (2n^3) + (2n^2d) + (n^4 - n^2) + (2n^3d) + \big(2d^2(\ell - 1 - p) + 2d^2p\big) \\ &\leq n^4 + 2n^3(d+1) + 2n^2d - n + 2d^2(\ell - 1) \end{aligned} \tag{4.1}$$

Having determined a bound for GBC(C), we can prove that the GBC of a group that covers less than ℓ edges smaller than the threshold t for large enough d.

$$\mathrm{GBC}(C) < 2d^2\ell \tag{4.2}$$

$$\Updownarrow$$

$$n^4 + 2n^3(d+1) + 2n^2d - n + 2d^2(\ell - 1) < 2d^2\ell$$

$$\Updownarrow$$

$$\frac{n^3 + n^2}{2} + \frac{1}{2}\sqrt{n^6 + 2n^5 + 3n^4 + 4n^3 - 2n} < d \tag{4.3}$$

It is easy to verify that a large d bounded by $n^{\mathcal{O}(1)}$ can be picked so that the last inequality is fulfilled. From the equivalence of inequalities (4.2) and (4.3), it follows that for large enough d inequality (4.2) holds. This implies that a value, namely $2d^2\ell$, can be chosen so that all sets C that cover less than ℓ edges in E have a GBC(C) strictly smaller than $2d^2\ell$. Thus we have shown that if $\mathcal{I}$ is a no-instance, then $\mathcal{J}$ must also be a no-instance. Equivalently, if $\mathcal{J}$ is a yes-instance, then $\mathcal{I}$ is also a yes-instance.

This completes the parameterized reduction. To summarize, we showed how an instance $\mathcal{I} = (G, k, \ell)$ of Partial Vertex Cover can be used to construct an instance $\mathcal{J} = (G', k, 2d^2\ell)$ of Maximum Betweenness Centrality. We showed that the construction can be done in polynomial time, and that it preserves the size of the parameter of both instances. Finally, by proving that instance $\mathcal{I}$ is a yes-instance if and only if $\mathcal{J}$ is also a yes-instance, we showed that the construction constitutes a parameterized reduction.

As Partial Vertex Cover is known to be W[1]-complete when parameterized by the size of the cover [23], the existence of the reduction implies that Maximum Betweenness Centrality is W[1]-hard when parameterized by the budget. ☐

Corollary 4.2. *Weighted Maximum Betweenness Centrality is* W[1]*-hard when parameterized by the budget.*

This result can be understood as evidence against the existence of a FPT algorithm for Maximum Betweenness Centrality parameterized by the budget. We complement this result by giving lower bounds for this problem, and showing that existing state of the art algorithms are indeed asymptotically optimal.

4.1.1 Lower bounds

Lower bounds on computational complexity are useful for knowing whether an optimal algorithm has been found. They also prevent the expenditure of many unfruitful hours which would otherwise be devoted to finding algorithms that are provably nonexistent. We show that there is no $f(b) \cdot n^{o(b)}$ time algorithm for Maximum Betweenness Centrality parameterized by the budget b.

Theorem 4.3. *There is no algorithm running in $f(b) \cdot n^{o(b)}$ time for Maximum Betweenness Centrality parameterized by the budget b, assuming* ETH.

Proof. Cygan et al. [23] give in Theorem 13.6 a parameterized reduction from Independent Set on regular graphs to Partial Vertex Cover that preserves the size of the parameter. From [23, Observation 14.22, Corollary 14.23] it follows that there is no $f(b) \cdot n^{o(b)}$ time algorithm for Partial Vertex Cover parameterized by the size k of the cover. As the reduction given in Theorem 4.1 also preserves the size of the parameter, it follows from [23, Observation 14.22] and from the previous theorem that there is no algorithm running in $f(b) \cdot n^{o(b)}$ time for Maximum Betweenness Centrality parameterized by the budget b. ☐

Corollary 4.4. *There is no algorithm running in $f(b) \cdot n^{o(b)}$ time for Weighted Maximum Betweenness Centrality parameterized by the budget b, assuming* ETH.

The $f(b) \cdot n^{o(b)}$ lower bound indicates that the n^b factor in the running time of the algorithms described in Section 3.5 cannot be improved, unless ETH fails.

4.2 Threshold

In this section, we discuss the parameterization of Maximum Betweenness Centrality by the threshold. We begin by giving bounds for non-trivial thresholds.

Lemma 4.5. *Let $\mathcal{I} = (G, b, t)$ be an instance of Maximum Betweenness Centrality parameterized by the threshold t. If either $t < n$ or $t > n^2 - n$, then for any budget b instance $\mathcal{I}$ can be trivially decided.*

Proof. Clearly, picking any single vertex v will cover at least all shortest paths between v and every vertex from $V \setminus \{v\}$, hence GBC($\{v\}$) will be at least equal to $n - 1$. On the other hand, as there are at most $n^2 - n$ pairs of vertices in G, and every pair can contribute with at most 1 to the GBC of any group, it follows that there is no group C such that GBC(C) is greater than $n^2 - n$. Hence any instance with a threshold greater than $n^2 - n$ is a no-instance. ☐

Because nontrivial instances will have large thresholds in comparison to n, even if an FPT algorithm for Maximum Betweenness Centrality parameterized by the threshold were to be found, it would not necessarily lead to an improvement in computational complexity. Assume that an $f(t) \cdot n^{\mathcal{O}(1)}$ time FPT algorithm for Maximum Betweenness Centrality parameterized by the threshold t was found. Although for fixed thresholds the algorithm would be polynomial in n, for nontrivial instances the running time of the trivial algorithm would be asymptotically better, unless $f = o(n^b)$. To overcome this limitation, we define a variant of

MAXIMUM BETWEENNESS CENTRALITY whereby the threshold indicates what fraction of the total number of shortest paths is covered.

MAXIMUM BETWEENNESS CENTRALITY with normalized threshold

Input:	A graph G, a budget b, and a threshold t'.
Task:	Decide whether a set C of vertices exists such that its size does not exceed the budget b, and $\frac{\mathrm{GBC}(C)}{n^2-n}$ is at least equal to the threshold t'.

Similarly, the threshold can be understood as the excess of GBC over n.

MAXIMUM BETWEENNESS CENTRALITY with shifted threshold

Input:	A graph G, a budget b, and a threshold t''.
Task:	Decide whether a set C of vertices exists such that its size does not exceed the budget b, and $\mathrm{GBC}(C) - n$ is at least equal to the threshold t''.

PARTIAL VERTEX COVER parameterized by the minimal number of edges to cover is in FPT either when vertices are unweighted (Kneis et al. [30]) or weighted (Bläser [31]). Thus even though the reduction used in the proof of Theorem 4.1 does constitute a parameterized reduction from PARTIAL VERTEX COVER parameterized by the number of edges to cover to MAXIMUM BETWEENNESS CENTRALITY with normalized threshold or with shifted threshold parameterized by the threshold, we cannot make any inferences regarding the parameterized complexity of either variant of MAXIMUM BETWEENNESS CENTRALITY (nor its weighted versions).

Chapter 5
Vertex cover number

The notion that vertices in a vertex cover are more central is somewhat intuitive. The proof of the NP-hardness of Maximum Betweenness Centrality is based on the fact that no group can have greater GBC than a vertex cover, as a vertex cover detects all communication between every pair of vertices.

We study Maximum Betweenness Centrality parameterized by the vertex cover number; the size of a minimal vertex cover. We show that Maximum Betweenness Centrality is in FPT when parameterized by the vertex cover number. More precisely, we describe an algorithm that decides an instance of Maximum Betweenness Centrality in $\mathcal{O}(nm + (2^k k)^k k^3)$ time, when the GBC of a set of size b can be determined in $\mathcal{O}(b^3)$ time, after $\mathcal{O}(nm)$ time preprocessing.

5.1 Algorithmic results

In this section, we show that the amount of vertices that need to be considered in order to find a group with maximal GBC can be expressed as a function of the vertex cover number. Note that although the Maximum Betweenness Centrality problem consists only in deciding whether a group with GBC at least equal to a certain threshold exists, if we can show that a group with maximal GBC can be chosen from a subset of vertices, the remaining vertices do not need to be considered. If the GBC of the group with maximal GBC is below the threshold, then the GBC of any group will also be below the threshold. Although this approach does not constitute a kernelization scheme, since the remaining vertices cannot be removed, it reduces the size of the search space considerably.

Let $\mathcal{I} = (G, b, t, k)$ be an instance of Maximum Betweenness Centrality parameterized by the vertex cover number k. We assume that b is smaller than k, as picking the vertex cover is otherwise a trivial solution.

We begin by observing that all vertices not in a vertex cover that are connected to the same vertices from the vertex cover are interchangeable.

Remark 5.1. Let T be a set of vertices such that $\mathrm{N}(t)$ is equal for all t in T, and let C be a group containing d vertices from T. If a group C' differs from C only in the particular vertices from T it contains, then $\mathrm{GBC}(C) = \mathrm{GBC}(C')$.

This stems from the fact that said vertices all have the same neighbourhood, and that they are pairwise nonadjacent.

We employ this observation to justify the selection of a subset P of vertices from $V \setminus M$, so that a group with maximal GBC can be picked from $P \cup M$. The algorithmic improvement comes from observing that the size of P can be bounded up with a function of the vertex cover number. Consider the graph in Fig. 5.1.

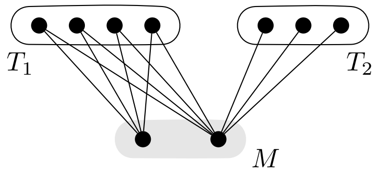

Figure 5.1 A graph with vertex cover M.

For a budget equal to 2, it is easy to see that at most 2 vertices will be picked either from T_1 or T_2. Since the particular selection of vertices from T_1 and T_2 is not important, as observed in Remark 5.1, we can mark only 2 vertices from each set as *include* and disregard the rest[1], thus reducing the search space for groups with maximal GBC.

Although the unmarked vertices from T_1 and T_2 do not need to be considered, they may not be removed from G, since the contribution of the shortest paths that contain these vertices to the centrality of the unmarked vertices is not constant. Consider the following graph.

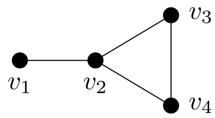

Figure 5.2 A small graph.

If the budget b equals 1, then picking v_1 is never optimal, as v_2 covers at least all shortest paths that contain v_1. Although in this sense v_1 can be disregarded when searching for groups with maximal GBC, it may not be removed from G. If v_1 were to be removed, all the remaining vertices would have the same BC (4). Clearly, only the v_2 vertex has the greatest BC.

We can use the previous observations to bound the number of vertices that need to be considered in order to find a group with maximal GBC.

Lemma 5.2. *Let M be a vertex cover of size k. No more than $2^k k$ vertices from $V \setminus M$ need to be considered in order to find a group with maximal* GBC*.*

Proof. Because M is a vertex cover, the neighbourhood of every vertex not in M is a subset of M. As M contains k vertices, there are at most 2^k vertices not in M with unique neighbourhoods.

Let T be the set of vertices whose neighbourhood is $S \subseteq M$. Consider a group C, such that it contains p vertices from $S \cup T$ and there is no group with greater GBC. As described in Remark 5.1, if C contains p vertices from T then any p vertices can be picked without changing GBC(C). Combining this with the observation that no more than b vertices can be picked from T, we can mark b vertices from T as *include*, and disregard the rest.

Because there are 2^k subsets of M, and processing each subset can lead to the inclusion of at most b vertices, it follows that there are at most $2^k b$ vertices not in M that need to be considered in order to find a group with maximal GBC. As we assume that the budget is smaller than the vertex cover number, we can relax this bound to $2^k k$. ☐

Corollary 5.3. *There is an $\mathcal{O}(nm + (2^k k)^k k^3)$ time* FPT *algorithm for* Maximum Betweenness Centrality *parameterized by the vertex cover number k.*

[1] In practice, picking more than $|N(T_1)|$ vertices from $T_1 \cup N(T_1)$ leads to no increase in GBC, since every shortest path that contains a vertex from $T_1 \cup N(T_1)$ can be covered simply by picking all vertices from $N(T_1)$. Thus only 2 vertices from T_1 and 1 vertex from T_2 need to be considered in order to find a group with maximal GBC.

Proof. Using the algorithm of Puzis et al. [19] to check each of the $\binom{2^k k}{b}$ groups of size b results in an $\mathcal{O}(nm + (2^k k)^b)$ time algorithm. We can replace b with k as we assume the vertex cover number is larger than the budget. ☐

Corollary 5.4. *There is an* $\mathcal{O}(nm + (2^k k)^k e^k k^3)$ *time* FPT *algorithm for* Weighted Maximum Betweenness Centrality *parameterized by the vertex cover number* k*.*

The only difference with regards to Weighted Maximum Betweenness Centrality, apart from the need to check all sets of size *at most* b, is that only the cheapest b vertices from any T should be considered when searching for groups with maximal GBC.

As a practical note, although we cannot exclude all vertices that share a neighbourhood when the size of the common neighbourhood is arbitrary, this is not the case when the degree of all vertices is equal to 1. In other words, assuming vertices of degree at least 2 are available, vertices with degree 1 can be safely ignored when looking for a group with maximal GBC. Although these vertices can be disregarded, they may not be removed, as they affect the centrality of the remaining vertices in a graph to varying degrees.

5.2 An open question

In the previous section we gave an upper bound for the number of vertices not in a vertex cover that need to be considered in order to find a group with maximal GBC. In this section we try to gain a deeper understanding of this bound, and conjecture that a stronger bound may exist.

A good starting point is asking whether a group with maximal GBC can be picked from a minimal vertex cover. After all, a vertex cover detects all communication between every pair of vertices, and the notion that vertices in a vertex cover are more central is fairly reasonable. This is not the case, however, as the examples in Fig. 5.3 show.

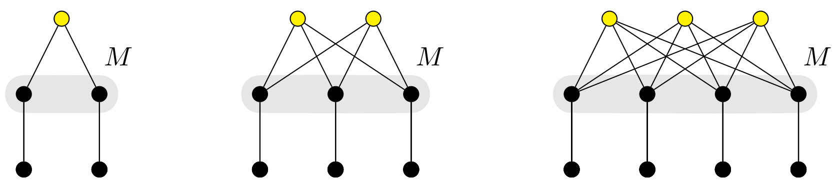

Figure 5.3 Graphs for which a group of size $|M| - 1$ with maximal GBC is not a subset of a minimal vertex cover M. The groups of yellow vertices are the groups with maximum GBC.

Note that for each of the graphs from Fig. 5.3, adding a vertex such that it is connected to every vertex in M leads to every group with maximal GBC becoming a subset of M. An example is shown in Fig. 5.4.

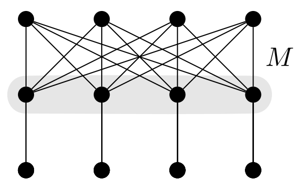

Figure 5.4 Graph for which every group of size at most $|M|$ with maximal GBC is a subset of the minimal vertex cover M.

This has to do with Remark 5.1. The more vertices that are added to a group of vertices that shares the same neighbourhood, the more their centrality becomes “diluted,” and the less convenient it is to pick vertices from said group. Formally, we have the following conjecture.

Conjecture 5.5. *Let S be a subset of a minimal vertex cover M and T the set of vertices whose neighbourhood is equal to S. Let C be a group containing p vertices from $S \cup T$, p_s from S and p_t from T. If $|S| \leq |T|$, then each of the p vertices chosen from the group with maximal* GBC *belongs to S and $p_t = 0$.*

In other words, we believe that if a group T of vertices not in M whose neighbourhood is S contains more elements than S, then a group with maximal GBC exists such that it does not contain any vertex from T. Note that this is a stronger result than that of the previous section. Instead of marking at most $|S|$ vertices as *include* of every T, we mark only those vertices of the sets T that have less elements than S.

We were neither able to prove nor disprove this conjecture.

Chapter 6

Distance to clique

We show that Maximum Betweenness Centrality is in FPT when parameterized by the distance to clique and the budget. The distance to clique is defined as follows:

Definition 6.1 (Distance to clique)**.** *Let G be a graph. A* clique modulator *of M is a subset of V such that $G[V \setminus M]$ is a clique. The* distance to clique *of graph G is equal to the size of a minimal clique modulator. Equivalently, the distance to clique of G is equal to the least number of vertices that need to be removed from G in order to obtain a clique.*

In Remark 5.1 we observed that vertices with equal neighbourhoods are interchangeable, in the sense that the particular selection of a fixed amount of said vertices has no effect on the GBC of a group containing them. The structure of a graph indicated by the distance to clique parameter is slightly different, but we use a similar approach. For any given clique modulator M, it is clear that the remaining vertices induce a clique (and hence all induced subgraphs of $V \setminus M$ are also cliques). Thus, as in Remark 5.1, we can group vertices from $V \setminus M$ by their neighbourhood in M. We show that vertices from such groups are also interchangeable.

Lemma 6.2. *Let G be a graph. Let C be a set containing d vertices from $K \subseteq V$, such that $G[K]$ is a clique and all vertices from K are connected to the same vertices from $V \setminus K$. Then the particular selection of a fixed amount of vertices from K has no effect on* GBC(C)*.*

Proof. We begin by observing that every shortest path either contains at most one vertex from K, or is a path of length 1 containing two vertices from K.

It is easy to see that no shortest path contains more than two vertices from K, as $G[K]$ is a clique. For the same reason, any shortest path containing two vertices from K must have length 1. With regards to these K-K paths, the particular selection of vertices from K will always result in the same contribution to GBC, as the number of covered K-K paths is only a function of the number of picked vertices from K.

We now consider only shortest paths containing a single vertex from K. Every such path corresponds to $|K|-1$ shortest paths differing only in the particular vertex from K they contain. Hence the GBC of a group containing a fixed amount of vertices from K is independent of the particular selection of vertices from K. □

As with Lemma 5.2, we use the previous result to reduce the size of the search space necessary for finding a group with maximal GBC. Consider the graph in Fig. 6.1. For a budget $b = 2$, only 2 vertices from K_1 and 2 from K_2 need to be considered in order to find a group with maximal GBC. Note that, as was the case with the vertex cover number, considering only 1 vertex from K_2 is not sufficient, since picking all vertices from $N(K_2) \cap M$ does not detect all communication that would be detected if all vertices from K_2 were to be picked.

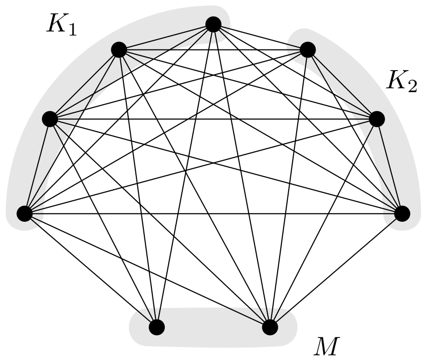

Figure 6.1 Example of a graph G with distance to clique 2. The clique modulator of G is the set M. The sets K_1 and K_2 contain vertices that are connected to the same vertices from M, respectively.

Lemma 6.3. *Let M be a clique modulator of size k. There are at most $2^k b$ vertices not in M that can form part of a group with maximal* GBC*.*

Proof. As M is a clique modulator, we can partition $V \setminus M$ into at most $2^{|M|}$ groups according to their connections with M. As shown in Lemma 6.2, the particular selection of vertices from any such group is irrelevant. Hence at most b vertices from each such group need to be considered, and at most $2^k b$ vertices not in a clique modulator need to be considered in order to find a group with maximal GBC. ☐

Theorem 6.4. *There is an $\mathcal{O}\Big(nm + (2^k b)^b b^3\Big)$ time* FPT *algorithm for* Maximum Betweenness Centrality *parameterized by the combined parameter distance to clique k and budget b.*

Proof. The running time is attained by using the algorithm of Puzis et al. [19] to calculate the GBC of each of the $2^k b$ groups of size b in $\mathcal{O}(b^3)$ time, after $\mathcal{O}(nm)$ preprocessing. ☐

Corollary 6.5. *There is an $\mathcal{O}\Big(nm + (2^k b)^b e^b b^3\Big)$ time* FPT *algorithm for* Weighted Maximum Betweenness Centrality *parameterized by the combined parameter distance to clique k and budget b.*

Note that we cannot assume that the budget is smaller than the distance to clique, as can be done regarding the vertex cover number. A budget greater than the vertex cover number allowed picking the vertex cover as a solution, hence detecting all communication. Because we were not able to prove that picking the clique modulator is always optimal, it is possible that non-trivial instances of Maximum Betweenness Centrality exist such that the budget is greater than the distance to clique.

Chapter 7
Twin cover number

In the previous chapters, we have shown that Maximum Betweenness Centrality is in FPT when parameterized by either the vertex cover number or the distance to clique and the budget. In this chapter, we study a less restrictive parameter called twin cover number, a generalization of both the vertex cover number and the distance to clique. The twin cover number is defined as follows.

Definition 7.1 (Twin cover number [32])**.** *$M \subseteq V(G)$ is a* twin-cover *of G if for every edge $\{u, v\} \in E$ either*

1. *$u \in M$ or $v \in M$, or*

2. *u and v are twins, i. e. , all other vertices are either adjacent to both u and v, or none.*

We then say that G has twin-cover number k if k is the least possible size of a twin-cover of G.

From this definition, it follows that $G - M$ is the disjoint union of potentially multiple cliques.

We show that Maximum Betweenness Centrality is in FPT when parameterized by the twin cover number k and the budget b.

Consider the graph in Fig. 7.1.

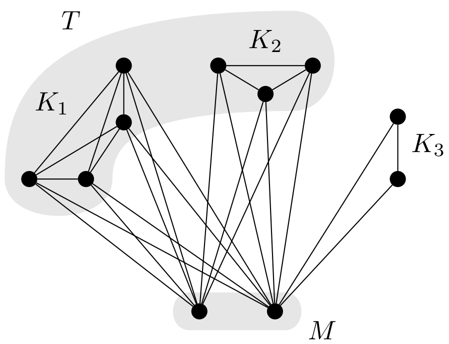

Figure 7.1 A graph with twin cover M.

As with the vertex cover number and the distance to clique, we can group vertices from $V \setminus M$ into at most 2^k groups, such that all vertices from each group are connected to the same vertices from M. In this case, the vertices from $V \setminus M$ can be divided into two groups, $T = K_1 \cup K_2$ and K_3. We now consider the set T, and denote by S the vertices from M that T is connected to (in this particular case $S = M$). The set $R = V \setminus (S \cup T)$ corresponds to the remaining vertices from V (in this case K_3).

We wish to show that only a small amount of vertices from T need to be considered in order to find a group with maximal GBC. In order to do this, we observe that, with respect to certain shortest paths, the particular selection of vertices from T has no effect on the GBC of a group that contains them.

Lemma 7.2. *Let M be a twin cover, and $S \subseteq M$ its subset. Let $T \subseteq V \setminus M$ be a set such that for all $v \in T$, $N(v) = S$. Then every shortest path containing vertices from $S \cup T$ is either a subset of $S \cup T$, or it contains at most one vertex from T.*

Proof. We show by contradiction that no shortest path containing vertices from $S \cup T$ and also from $V \setminus (S \cup T)$ can contain more than one vertex from T.

Let P be a shortest path containing a vertex v from $R = V \setminus (S \cup T)$ and two vertices $t_1, t_2 \in T$. It is easy to verify that P contains at least one vertex from S, as vertices from T communicate with those from R exclusively through S. If t_1 and t_2 belong to the same clique from T, then P cannot be a shortest path, as all vertices from T are connected to all vertices from S. We now assume that t_1 and t_2 belong to different cliques from T. In this case, P must contain two vertices from S, since the cliques in T communicate exclusively through S. Clearly, such a path may be shortened, since the first vertex from S in P is already connected to all vertices from T, and thus P cannot be a shortest path. ◻

This indicates that, with regards to R, vertices from T are interchangeable. In other words, the contribution of shortest paths containing vertices from R and from T to the GBC of a group C containing a fixed amount of vertices from T is independent of the particular selection of vertices from T.

We extend this property to all but T-T and T-S-T shortest paths, by calculating the exact contribution of shortest paths that are subsets of $S \cup T$ to the GBC of any group. The contribution of a shortest s-t path to the GBC of any group is equal to $\frac{1}{\sigma_{s,t}}$. Thus the contribution of a set X to the GBC of a set C can be understood as the sum of the contribution of all shortest paths that are covered exclusively by X, or equivalently $\mathrm{GBC}(C) - \mathrm{GBC}(C \setminus X)$.

Let C be a set containing p_s vertices from S and p_t from T. The number of cliques in T is denoted by ℓ, and the size of the i-th clique is denoted by c_i. Finally, the number of picked vertices from the i-th clique is denoted by p_i. Then the contribution of shortest paths that are subsets of $S \cup T$ to $\mathrm{GBC}(C)$ is equal to

$$
\begin{aligned}
&\sum_{\substack{u,v \in S \cup T \\ u \neq v}} \frac{\sigma^R_{u,v}(C)}{\sigma_{u,v}} = \\
&\qquad (|S| + |T|)^2 - (|S| + |T|) && \text{Paths that are subsets of } S \cup T \\
&\qquad - 2(|S| - p_s)(|T| - p_t) && \text{Uncovered } S\text{-}T \text{ paths} \\
&\qquad - \left(\left((|S| - p_s)^2 - (|S| - p_s) \right) \frac{|T| - p_t}{|T|} \right) && \text{Uncovered } S\text{-}T\text{-}S \text{ paths} \\
&\qquad - \sum_{i=1}^{\ell} \left((c_i - p_i)^2 - (c_i - p_i) \right) && \text{Uncovered } T\text{-}T \text{ paths (in a single clique)} \\
&\qquad - 2 \sum_{i=1}^{\ell-1} \sum_{j=i+1}^{\ell} \left((c_i - p_i)(c_j - p_j) \frac{|S| - p_s}{|S|} \right) && \text{Uncovered } T\text{-}S\text{-}T \text{ paths}
\end{aligned}
$$

It is easy to see that, for any fixed p_s and p_t, the contribution of S-T and S-T-S paths is independent of the particular selection of vertices from $S \cup T$. Note that we disregard shortest paths containing vertices from R. The contribution of these paths is given either by the particular selection of vertices from S, or by the total amount of vertices picked from T. Thus none of this paths is dependent on the particular selection of vertices from T. On the contrary, the contribution of T-T and T-S-T paths depends on the amounts of picked vertices from each individual clique in T. If we wish to pick only b vertices from T for consideration when searching for groups with maximal GBC, we need to ensure that the selection of said vertices minimizes

$$\sum_{i=1}^{\ell}\Big((c_i - p_i)^2 - (c_i - p_i)\Big) + 2\sum_{i=1}^{\ell-1}\sum_{j=i+1}^{\ell}\left((c_i - p_i)(c_j - p_j)\frac{|S| - p_s}{|S|}\right) \tag{7.1}$$

Before proceeding, we show how this expression can be simplified. For brevity, we denote the expression in Eq. (7.1) by $g(p)$, where p_i is the number of vertices picked from the i-th clique of T. Then for $g(p)$ we have

$$\begin{aligned}
g(p) &= \sum_{i=1}^{\ell}\Big((c_i - p_i)^2\Big) - (|T| - p_t) + \frac{|S| - p_s}{|S|}\left(\sum_{i=1}^{\ell}\sum_{j=1}^{\ell}((c_i - p_i)(c_j - p_j)) - \sum_{i=1}^{\ell}(c_i - p_i)^2\right)\\
&= \left(1 - \frac{|S| - p_s}{|S|}\right)\sum_{i=1}^{\ell}\Big((c_i - p_i)^2\Big) - (|T| - p_t) + \frac{|S| - p_s}{|S|}\sum_{i=1}^{\ell}\left((c_i - p_i)\sum_{j=1}^{\ell}(c_j - p_j)\right)\\
&= \left(1 - \frac{|S| - p_s}{|S|}\right)\sum_{i=1}^{\ell}\Big((c_i - p_i)^2\Big) - (|T| - p_t) + \frac{(|S| - p_s)(|T| - p_t)}{|S|}\sum_{i=1}^{\ell}(c_i - p_i)\\
&= \left(1 - \frac{|S| - p_s}{|S|}\right)\sum_{i=1}^{\ell}\Big((c_i - p_i)^2\Big) - (|T| - p_t) + \frac{(|S| - p_s)(|T| - p_t)^2}{|S|}
\end{aligned} \tag{7.2}$$

Thus, for fixed p_t and p_s, minimizing $g(p)$ can be done by minimizing

$$\sum_{i=1}^{\ell}(c_i - p_i)^2 \tag{7.3}$$

It is easy to see that this can be done greedily. We begin by setting $p_i = 0$ for all i. Let j be the index of a term of the sum in Eq. (7.3), such that there is no other term with greater value. Clearly, the sum decreases the most when p_j is increased by one. We can repeat this step until p_t vertices have been picked, ensuring that the sum in Eq. (7.3) is minimal, and thus that the selected vertices from T cover shortest paths with the greatest contribution.

Note that the set of ideal vertices for $p_t = d$ can be picked so that it is a subset of the set of ideal vertices for $p_t = d + 1$. If this were not so, picking the b ideal vertices would not be of much use. After selecting the set of vertices from $V \setminus M$, we still need to find a group with maximal GBC, and check whether its centrality exceeds the threshold. A potential group might not necessarily contain exactly b vertices from T, in which case a different set of vertices from T would need to be considered.

We use the previous observation to justify the selection of only a small amount of vertices for consideration when searching for groups with maximal GBC. Since the amount of included vertices can be bounded by a function of the twin cover number, the described algorithm is FPT.

Theorem 7.3. *There is an* $\mathcal{O}\Big(nm + 2^k n\log(n) + (2^k b)^b b^3\Big)$ *time algorithm for* Maximum Betweenness Centrality *parameterized by the twin cover number k and the budget b.*

Proof. Let G be a graph, containing a twin cover M of size k. We can partition vertices not in M into at most 2^k groups, such that the vertices from each group are all connected to the same vertices from M. Let T be one such group, and S its neighbourhood in M. We denote by c_i the size of the i-th clique in T[1]. As explained earlier, b vertices from T can be marked as *include* in the following manner. First, sort the values of c_i in decreasing order. Since there are at most n cliques in T, this can be done in $\mathcal{O}(n\log(n))$ time using well known algorithms. We then decrease the values $c_1, c_2, \ldots$ by a total of b, so that maximal values are decreased first.

Thus, processing at most 2^k groups can be done in $\mathcal{O}(2^k n\log(n))$ time, and afterwards at most $\mathcal{O}(2^k b)$ vertices are marked as *include*. We can then use the algorithm by Puzis et al. [19] to calculate the GBC of each of the $\binom{2^k b}{b}$ groups of size b in $\mathcal{O}(nm + (2^k b)^b b^3)$ time after $\mathcal{O}(nm)$ preprocessing, which results in the total running time of $\mathcal{O}(nm + 2^k n\log(n) + (2^k b)^b b^3)$. Note that, in order for this approach to produce correct results, during the step when vertices are marked as *include*, they should be also ordered. Then when the GBC of a set C is being calculated, if C contains less then b vertices from C, then the vertices from T should be picked in order. ◻

Corollary 7.4. *There is an* $\mathcal{O}\Big(nm + 2^k n\log(n) + (2^k b)^b e^b b^3\Big)$ *time algorithm for* Weighted Maximum Betweenness Centrality *parameterized by the twin cover number k and the budget b.*

Proof. The proof is analogue to that of Theorem 7.3, however instead of calculating the GBC of all groups of size exactly b, we calculate the GBC of all groups of size at most b. ◻

[1] We assume that a twin cover is known. It is easy to verify that determining the size of the cliques that have the same neighbourhood in M can be done in $n^{\mathcal{O}(1)}$ time by removing M from G and counting the resulting number of independent components.

Chapter 8
Conclusion

8.1 Results

In Section 3.3, we showed that Maximum Betweenness Centrality (and its weighted variant) is not only NP-hard, but also in NP, and thus NP-complete (Theorem 3.1, Corollary 3.2).

From a parameterized perspective, in Section 4.1 we showed that Maximum Betweenness Centrality is W[1]-hard when parameterized by the budget (Theorem 4.1). To complement this result we showed in Section 4.1.1 that there is no ETH time algorithm for Maximum Betweenness Centrality parameterized by the budget, assuming ETH (Theorem 4.3).

The following table lists our algorithmic results. Note that all of the results for Maximum Betweenness Centrality have corresponding results for Weighted Maximum Betweenness Centrality, usually as corollaries of the main result.

Table 8.1 Algorithmic results.

Group Betweenness Centrality

Parameter	*Result*	*Reference*
Group size k	$\mathcal{O}\big(nm + kn^2\big)$ time algorithm	Theorem 2.7

Maximum Betweenness Centrality

Parameter	*Result*	*Reference*
Budget b	$\mathcal{O}\big(nm \cdot n^b\big)$ time trivial XP algorithm	Theorem 3.3
	$\mathcal{O}\big(nm + b^3 n^b\big)$ time XP algorithm	Theorem 3.5
	$\mathcal{O}\big(nm + bn^{b+2}\big)$ time XP algorithm	Theorem 3.7
	$f(b) \cdot n^{o(b)}$ lower bound (under ETH)	Theorem 4.3
Vertex cover no. k	$\mathcal{O}(nm + (2^k k)^k k^3)$ time FPT algorithm	Corollary 5.3
Distance to clique k and budget b	$\mathcal{O}\Big(nm + (2^k b)^b b^3\Big)$ time FPT algorithm	Theorem 6.4
Twin cover no. k and budget b	$\mathcal{O}\Big(nm + 2^k n \log(n) + (2^k b)^b b^3\Big)$ time FPT algorithm	Theorem 7.3

8.2 Further work

We conclude by indicating possible directions for future research.

Finding a more efficient algorithm for computing the GBC of a group would result in more efficient algorithms for Maximum Betweenness Centrality, since all algorithms we described rely on reducing the search space and then computing the GBC of all subsets of valid size.

Although we showed that Maximum Betweenness Centrality is W[1]-hard when parameterized by the budget, a good first step for a future multivariate analysis of Maximum Betweenness Centrality would be to show whether it is W[1]-hard when parameterized by the threshold. In Section 4.2, we introduced two variants of Maximum Betweenness Centrality, for which the threshold is potentially smaller than n. Studying the parameterized complexity of these problems parameterized by the threshold might give useful insights regarding the complexity of the unmodified Maximum Betweenness Centrality problem.

In Section 5.2, we conjecture that there is a stronger upper bound for the number of vertices that are necessary in order to find a group with maximal GBC. Although showing that this conjecture is true will not lead to a dramatic asymptotic improvement, the intuition behind it might be of use when only an approximate result is required.

On a practical note, implementing the algorithms we described might show that some algorithms with poor asymptotic running times perform generally well in practice.

Finally, many additional parameters exist, such that the complexity of Maximum Betweenness Centrality parameterized by a particular parameter or combination of parameters might be analyzed. A parameter we studied, without any results, is the treewidth [23], which indicates how close to a tree a given graph is. Fink and Spoerhase [28] describe a polynomial time algorithm for Maximum Betweenness Centrality on trees, and although their approach cannot be applied directly to Maximum Betweenness Centrality parameterized by the treewidth, we believe that progress may be made in this direction.

Bibliography

1. BAVELAS, Alex. Communication Patterns in Task-Oriented Groups. *The Journal of the Acoustical Society of America*. 1950, vol. 22, no. 6, pp. 725–730. Available from DOI: 10.1121/1.1906679.
2. FREEMAN, Linton C. Centrality in social networks conceptual clarification. *Social networks*. 1978, vol. 1, no. 3, pp. 215–239.
3. LAL, Kumari Nidhi; KUMAR, Anoj. A popularity based content eviction scheme via betweenness-centrality caching approach for content-centric networking (CCN). *Wireless Networks*. 2019, vol. 25, no. 2, pp. 585–596.
4. MBIYA, Stephanie Mwika; HANCKE, Gerhard P.; SILVA, Bruno. An Efficient Routing Algorithm for Wireless Sensor Networks based on Centrality Measures. *Acta Polytechnica Hungarica*. 2020, vol. 17, no. 1, pp. 83–99. ISSN 17858860. Available from DOI: 10.12700/APH.17.1.2020.1.5.
5. TUZCU, Atakan; ARSLAN, Hilal. Betweenness Centrality in Sparse Real World and Wireless Multi-hop Networks. In: KAHRAMAN, Cengiz; CEBI, Selcuk; CEVIK ONAR, Sezi; OZTAYSI, Basar; TOLGA, A. Cagri; SARI, Irem Ucal (eds.). *Intelligent and Fuzzy Techniques for Emerging Conditions and Digital Transformation*. Cham: Springer, 2022, vol. 307, pp. 217–224. ISBN 978-3-030-85625-0. Available from DOI: 10.1007/978-3-030-85626-7_27.
6. PENG, Peng; CHENG, Shifen; CHEN, Jinhai; LIAO, Mengdi; WU, Lin; LIU, Xiliang; LU, Feng. A fine-grained perspective on the robustness of global cargo ship transportation networks. *Journal of Geographical Sciences*. 2018, vol. 28, no. 7, pp. 881–889. Available from DOI: 10.1007/s11442-018-1511-z.
7. LIU, Weiyan; LI, Xin; LIU, Tao; LIU, Bin. Approximating betweenness centrality to identify key nodes in a weighted urban complex transportation network. *Journal of Advanced Transportation*. 2019, vol. 2019. Available from DOI: 10.1155/2019/9024745.
8. DANIEL, Cecile; FURNO, Angelo; EL FAOUZI, Nour-Eddin; SHARMA, Rajesh; ZIMEO, Eugenio. Fast Computation of Betweenness Centrality to enable Real-time Resilience Assessment and Improvement of Complex Transport Networks. *HEART*. 2020.
9. GONZÁLEZ, Ana M Martín; DALSGAARD, Bo; OLESEN, Jens M. Centrality measures and the importance of generalist species in pollination networks. *Ecological complexity*. 2010, vol. 7, no. 1, pp. 36–43.
10. XIONG, Wei; XIE, Luyu; ZHOU, Shuigeng; LIU, Hui; GUAN, Jihong. The centrality of cancer proteins in human protein-protein interaction network: A revisit. *International Journal of Computational Biology and Drug Design*. 2014, vol. 7, no. 2-3, pp. 146–156.

11. ZITO, Alessandra; LUALDI, Marta; GRANATA, Paola; COCCIADIFERRO, Dario; NOVELLI, Antonio; ALBERIO, Tiziana; CASALONE, Rosario; FASANO, Mauro. Gene Set Enrichment Analysis of Interaction Networks Weighted by Node Centrality. *Frontiers in Genetics*. 2021, vol. 12. ISSN 1664-8021. Available from DOI: `10.3389/fgene.2021.577623`.
12. EVERETT, Martin; BORGATTI, Stephen. The Centrality of Groups and Classes. *Journal of Mathematical Sociology*. 1999, vol. 23, pp. 181–201. Available from DOI: `10.1080/0022250X.1999.9990219`.
13. DOLEV, Shlomi; ELOVICI, Yuval; PUZIS, Rami; ZILBERMAN, Polina. Incremental deployment of network monitors based on Group Betweenness Centrality. *Information Processing Letters*. 2009, vol. 109, no. 20, pp. 1172–1176. ISSN 0020-0190. Available from DOI: `https://doi.org/10.1016/j.ipl.2009.07.019`.
14. PUZIS, Rami; ELOVICI, Yuval; DOLEV, Shlomi. Finding the most prominent group in complex networks. *AI Communications*. 2007, vol. 20, pp. 287–296. 4.
15. MORSELLI, Carlo. Assessing Vulnerable and Strategic Positions in a Criminal Network. *Journal of Contemporary Criminal Justice*. 2010, vol. 26, no. 4, pp. 382–392. Available from DOI: `10.1177/1043986210377105`.
16. VEREMYEV, Alexander; PROKOPYEV, Oleg A.; PASILIAO, Eduardo L. Finding groups with maximum betweenness centrality. *Optimization Methods and Software*. 2016, vol. 32, no. 2, pp. 369–399. Available from DOI: `10.1080/10556788.2016.1167892`.
17. NEWMAN, M. E. J. Scientific collaboration networks. II. Shortest paths, weighted networks, and centrality. *Physical Review E*. 2001, vol. 64, no. 1. Available from DOI: `10.1103/physreve.64.016132`.
18. BRANDES, Ulrik. A faster algorithm for betweenness centrality. *The Journal of Mathematical Sociology*. 2001, vol. 25, no. 2, pp. 163–177. Available from DOI: `10.1080/0022250x.2001.9990249`.
19. PUZIS, Rami; ELOVICI, Yuval; DOLEV, Shlomi. Fast algorithm for successive computation of group betweenness centrality. *Physical Review E*. 2007, vol. 76, no. 5. Available from DOI: `10.1103/physreve.76.056709`.
20. MAHMOODY, Ahmad; TSOURAKAKIS, Charalampos E.; UPFAL, Eli. Scalable Betweenness Centrality Maximization via Sampling. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Francisco California USA: ACM, 2016, pp. 1765–1773. ISBN 978-1-4503-4232-2. Available from DOI: `10.1145/2939672.2939869`.
21. LAGOS, Tomás; PROKOPYEV, Oleg A.; VEREMYEV, Alexander. Finding groups with maximum betweenness centrality via integer programming with random path sampling. *Journal of Global Optimization*. 2023. Available from DOI: `10.1007/s10898-022-01269-2`.
22. ANGRIMAN, Eugenio; MEYERHENKE, Henning; BOJCHEVSKI, Aleksandar; ZÜGNER, Daniel; GÜNNEMANN, Stephan; GRINTEN, Alexander van der. Group Centrality Maximization for Large-scale Graphs. In: *2020 Proceedings of the Symposium on Algorithm Engineering and Experiments (ALENEX)*. 2020, pp. 56–69. Available from DOI: `10.1137/1.9781611976007.5`.
23. CYGAN, Marek; FOMIN, Fedor V.; KOWALIK, Łukasz; LOKSHTANOV, Daniel; MARX, Dániel; PILIPCZUK, Marcin; PILIPCZUK, Michał; SAURABH, Saket. *Parameterized Algorithms*. Springer, 2015. Available from DOI: `10.1007/978-3-319-21275-3`.
24. DIESTEL, Reinhard. *Graph Theory*. Vol. 173. 5th ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2017. Graduate Texts in Mathematics. ISBN 978-3-662-53621-6. Available from DOI: `10.1007/978-3-662-53622-3`.

25. ARORA, Sanjeev; BARAK, Boaz. *Computational complexity: a modern approach.* Cambridge, New York: Cambridge University Press, 2009. ISBN 978-0-521-42426-4.

26. DOWNEY, Rodney G; FELLOWS, Michael R; STEGE, Ulrike. Parameterized complexity: A framework for systematically confronting computational intractability. In: *Contemporary trends in discrete mathematics: From DIMACS and DIMATIA to the future*. 1999, vol. 49, pp. 49–99.

27. GAREY, Michael R.; JOHNSON, David S. *Computers and intractability: a guide to the theory of NP-completeness.* 27. print. New York [u.a]: Freeman, 2009. A series of books in the mathematical sciences. ISBN 978-0-7167-1045-5.

28. FINK, Martin; SPOERHASE, Joachim. Maximum Betweenness Centrality: Approximability and Tractable Cases. In: KATOH, Naoki; KUMAR, Amit (eds.). *WALCOM: Algorithms and Computation.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, vol. 6552, pp. 9–20. ISBN 978-3-642-19093-3. Available from DOI: 10.1007/978-3-642-19094-0_4.

29. GUO, Jiong; NIEDERMEIER, Rolf; WERNICKE, Sebastian. Parameterized Complexity of Vertex Cover Variants. *Theory of Computing Systems.* 2007, vol. 41, no. 3, pp. 501–520. Available from DOI: 10.1007/s00224-007-1309-3.

30. KNEIS, Joachim; LANGER, Alexander; ROSSMANITH, Peter. Improved Upper Bounds for Partial Vertex Cover. In: BROERSMA, Hajo; ERLEBACH, Thomas; FRIEDETZKY, Tom; PAULUSMA, Daniel (eds.). *Graph-Theoretic Concepts in Computer Science.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, vol. 5344, pp. 240–251. ISBN 978-3-540-92247-6. Available from DOI: 10.1007/978-3-540-92248-3_22.

31. BLÄSER, Markus. Computing small partial coverings. *Information Processing Letters.* 2003, vol. 85, pp. 327–331. Available from DOI: 10.1016/S0020-0190(02)00434-9.

32. GANIAN, Robert. Twin-Cover: Beyond Vertex Cover in Parameterized Algorithmics. In: MARX, Dániel; ROSSMANITH, Peter (eds.). *Parameterized and Exact Computation.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, vol. 7112, pp. 259–271. ISBN 978-3-642-28049-8. Available from DOI: 10.1007/978-3-642-28050-4_21.