GRAN SASSO SCIENCE INSTITUTE
SCHOOL OF ADVANCED STUDIES
Scuola Universitaria Superiore

IMT INSTITUTE FOR ADVANCED STUDIES LUCCA

Doctoral Thesis

Centrality maximization in complex networks

PhD Program in Computer Science: XXIX cycle

Author:

Lorenzo Severini

Supervisors:

Prof. Dr. Pierluigi Crescenzi
Dr. Gianlorenzo D'Angelo

Abstract

One of the main issue in complex networks analysis consists in determining what are the most important nodes in a network. For this reason, researchers have defined several centrality indices in order to measure this concept. In several scenarios, having a high centrality can have a positive impact on the node itself. Hence, in this thesis we study the problem of determining how much a node can increase its centrality by creating a limited amount of new edges incident to it. In particular, we cope with the problem of adopting the best strategy in order to increase the value of two well known centrality indices namely harmonic centrality (CM-H) and betweenness centrality (CM-B). We show that CM-H cannot be approximated in polynomial-time within a factor $1 - \frac{1}{3e}$ in directed graphs and $1 - \frac{1}{15e}$ in undirected graphs, unless $P = NP$. On the other hand, we prove that CM-B cannot be approximated in polynomial time within a factor $1 - \frac{1}{2e}$ in both directed and undirected graphs, unless $P = NP$. We then propose a greedy approximation algorithm for both problems with an almost tight approximation ratio in all the cases except for CM-B in undirected networks. We test the performance of our algorithms on both synthetic and real-world networks and we show that they provide a good solution in every case. Moreover, we design some heuristics in order to speed up the computation and run the algorithm on large graphs with millions of nodes and edges. We also study the problem of improving the ranking according to harmonic centrality (CRM-B) and betweenness centrality (CRM-B) by adding a limited amount of edges incident to a given node and we prove that it does not admit any polynomial-time constant factor approximation algorithm, unless $P = NP$. However, we experimentally show that our greedy algorithms allow a node to reach the top positions in the ranking by adding few new edges.

Acknowledgements

First of all, I would like to thanks Gianlorenzo D'Angelo for his constant and patient support and guidance during my PhD. Without his suggestions and his help this thesis would not have been possible.

Thanks to Pierluigi Crescenzi, for suggesting the topic of my research and for provide me a lot of useful advices.

I am also grateful to the reviewers of this thesis, Paolo Boldi e David Coudert, for their careful and precise comments that helped me to improve the quality of this work.

This work could not have been possible without my wonderful GSSI colleagues with whom I shared joy and pain in this three years.

First and foremost thanks to Yllka for sharing with me most of my research and, above all, for the great time we spent together in the office and in a lot of (mostly rainy) trips. Thanks to my former housemate Stefano R. for the long and interesting discussions on various topics. A special thank to Valentina for having always been a sincere a trustworthy friend. I am indebted to my many others GSSI colleagues whom shared with me all the "social life" in L'Aquila: thanks to Catia, Cosimo, Darko, Elena, Emilio, Feliciano, Gian Luca, Giancarlo, Ivano, Lucia, Matteo C., Matteo D., Marco, Maria, Marta, Michele, Mora, Stefano D., Venkat.

Thanks to Claudia for make me smile every day.

Finally, thanks to my parents and my sister for their constant and warm encouragement.

Contents

Chapter 1

Introduction

1.1 Context and motivations

Nowadays, the analysis of complex networks is a very active area of scientific research: its purpose is to study and model real world systems from different fields like computer science, physics and biology. When we talk about *networks*, we usually refer to a set of multiple items interconnected to each others. We call these items *nodes* or *vertices* and the links between them *edges* or *links* [70] and we usually represents a network as a *graph*. They are called *complex* because of the non trivial topological features that often occurs when modelling real case of systems. World Wide Web, social networks of acquaintance or other connections between individuals, collaboration networks, organizational networks and networks of business relations between companies, neural networks, metabolic networks, food webs, distribution networks such as blood vessels or postal delivery routes, citation networks, and many other are real world systems that can be modelled as complex networks.

The study of this area started in the 60s with the analysis of social networks represented by a set of people or groups of people with some pattern of contacts or interactions between them. Social scientists provided a set of methods for analysing the structure of whole social entities as well as a variety of theories explaining the patterns observed in these structures [93].

One issue in complex networks analysis is to understand how nodes are linked with other nodes. The famous Milgram experiment [66] introduced the concept of *small-world phenomenon* or "six-degree of separation" [50]. Milgram asked to random individuals to forward a letter to a *target* person living in another town. Each person could not send directly the mail to the target but he or she can only forward a letter to a single acquaintance that he or she knew directly. He noticed that the fraction of letters eventually arrived to the target took, in median, six steps. Thus, this experiment demonstrated that short paths are in abundance in social networks (in the case of Milgram experiment, friendship network). Later on, in order to build a model that captures these features, Watts and Strogatz [95] developed an algorithm to generate graphs with small-world properties. However, this model produces an unrealistic degree distribution: it does not consider the fact that many real networks are scale-free. In general, we define a *scale-free* network if the diameter (i.e. the length of longest shortest path) grows proportionally to the logarithm of the number of the nodes. Then, in 2002, Albert and Barabasi [3] built another model, based on preferential attachment, that overcome this limitation and generates scale-free networks.

Another issue in complex network analysis is to understand what are the most important nodes within the network. For this reason, one of the most studied concepts is that of the *centrality*. Informally speaking, a node is considered "central" if it is important within the network and it is believed that the importance that a node has within a network reflects, to some extent, the position of the node in the network and, more generally, the network structure. To mathematically capture these features, several centrality metrics (also known as centrality measures or centrality indices) are defined in literature. It has been experimentally observed that being central for a node, according to some centrality index, has several benefits for the node itself. For example, closeness centrality (the sum of the length of the shortest paths between a given node and all other nodes in the graph) is significantly correlated with citation counts of an author in author-citation networks [98], betweenness centrality (the number of shortest paths that passing through a given node divided th number of all possible shortest paths) is correlated with the efficiency of an airport in transportation networks [62], and both closeness and betweenness are correlated with the efficiency of an individual to propagate the information in a social network [61]. Therefore, a lot of research effort has been put on the problems of efficiently computing the centrality metrics of a given node or determining the most central nodes of a possibly large network, according to some metric.

A lot of research are done considering centrality indices from a *reactive* point of view, designing dynamic algorithms that allow to obtain the centrality of nodes after a network modification without recomputing all the values from scratch. Indeed, in this thesis, we look at centrality indices from a *proactive* point of view, that is we want to *modify* an existing network with the aim of improving the centrality of a given node. A network can be modified by adding or removing edges and nodes. By performing these operations the centrality of a node can vary with respect to the centrality of other nodes. For example, by adding edges, the length of the shortest paths between nodes can decrease affecting the centrality based on the distances between nodes.

This thesis address the issue of adopting the best "strategy" in order to increase the centrality value of a node as much as possible. We formulate this question as an optimization problem which consists in finding a limited amount of edges to be added in a graph in order to maximize the centrality of a given node within a network. We consider the problem of efficiently determining, for a given vertex v, the set of k edges incident to v that, when added to the original graph, maximizes the centrality of v, according to some index. We denote this optimization problem as Centrality Maximization problem (CM). We force a node to only add edges incident to itself because it is realistic in several scenarios like collaboration networks in which nodes represent users and links represent collaboration between users (e.g. authors collaborating in the same papers or actors that acted in the same movie) or social networks in which links represent friendship relation between users. In [28] the authors evaluate how the improvement of the harmonic centrality can have a positive effect to the spreading of information of a certain node to all other vertices in its connected component. To this aim, they perform several experiments on the Linear Threshold Model, which is a widely studied model in network analysis to represent the spread of information [49].

Improving the centrality of a given node can have a positive impact also for the link recommendation problem. The link recommendation task consists in suggesting potential connections to social network users with the aim of increasing their social circle. Link recommendations improve the user experience and at the same time help to increase the connectivity inside the network and speed-up the network growth. Most of the existing link recommendation methods focus on estimating the likelihood that a link is adopted by users and recommend links that are likely to be established [5, 60, 80, 99]. Recently, a new approach has been proposed whose aim is to recommend a set of links that, when added to the network, increases the centrality of a user in a network. In particular, suggesting links that minimize the expected average distance of a node accurately predicts the links that will actually appear in the graph [78]. An important step in this approach is to determine the set of links that, when added to the network, maximizes the specific centrality measure considered. In our experiments we show that we are able to compute a set of nodes that highly increases the harmonic centrality in very large collaboration networks such as those induced by the DBLP and IMDB databases [44, 58]. For these reasons, in this thesis we study the CM problem for to well-known indices namely harmonic centrality and betweenness centrality giving both theoretical and experimental results. We study the computational complexity of the problems, we design approximation algorithms and we experimentally evaluate the performance on both synthetic and real networks. In the next sections we state the main contributions and we show in detail the structure of the document.

1.2 Main contributions

In this thesis we tackle the problem of maximizing the centrality of a predefined node adding a certain amount of edges incident to it. In particular, we focus on two well-known centrality indices namely harmonic centrality (CM-H problem) and betweenness centrality (CM-B problem): for both problems, we study the computational complexity and we design several algorithms to solve them exactly and approximately. Moreover, we evaluate the performance of these algorithms on both synthetic and real-world graphs and we design and implement some heuristics in order to analyse large networks. In the following we list the main contributions of this thesis.

- We prove that CM-H problem is hard to be approximated within an approximation factor greater than $1 - \frac{1}{15e}$ in the undirected graph case (respectively, $1 - \frac{1}{3e}$ in the directed case). We analyse different natural algorithms (e.g the algorithm that adds edges to a set of nodes having highest degree) and we show that the only approach with bounded approximation factor is the greedy one: it yields a $\left(1 - \frac{1}{e}\right)$-approximation algorithm in both undirected and directed cases.
- For CM-B problem, we prove that it is hard to be approximated within an approximation factor greater than $1 - \frac{1}{2e}$ in both directed and undirected graphs. We analyse the same natural algorithms studied in the CM-H problem case and we show that the greedy approach yields a $\left(1 - \frac{1}{e}\right)$-approximation algorithm in directed graphs.
- We study a similar problem where the goal is to improve the position in the ranking induced by the centrality metric, namely Centrality Ranking Maximization problem (CRM), and we prove that it is NP-hard and cannot be approximated with any constant factor smaller than 1.
- For both CM problems, we present several experiments in order to evaluate how good is the approximation factor achieved by the greedy algorithm in the case of relatively small randomly generated graphs. We notice that the greedy algorithm seems to perform much better than the theoretical results, since it often computes an optimal solution and, in any case, it achieves an approximation factor significantly larger than the theoretical one.
- We compare the solution obtained by the greedy algorithm with the ones obtained by three natural algorithms. We show that the greedy approach performs much better than the other heuristics in terms of value and ranking obtained. Moreover, despite we proved that for CM-B on undirected case the greedy algorithm can

have an unbounded approximation ratio, it performs by far better than the other methods.

- Finally, we perform several experiments of the greedy approach applied to real-world collaboration, citation and transportation networks. By applying the greedy algorithm to real-world networks we observe that by adding very few edges a vertex can drastically increase its centrality measure and, hence, its ranking.

1.3 Structure of the thesis

In Chapter 2 we introduce the main definitions and we describe some preliminary results that will be used in the rest of the thesis. We formally define CM problem and the similar problem where the aim is to maximize the ranking instead of the value. Then, we focus on the main results in the literature on the problem of improving graph properties and we describe in detail the results on Centrality Maximization on PageRank and Eccentricity. In the following chapters we focus on CM problem for harmonic centrality (CM-H) and betweenness centrality (CM-B). In Chapter 3 we study the computational complexity of the CM-H problem and of similar problem where the aim is to optimize the ranking instead of the value (CRM-H), we propose several heuristics and we experimentally evaluate the quality of the solution and the performance on both synthetic and real network. In Chapter 4 we do a similar theoretical and experimental study for the CM-B and CRM-B problems. Finally, in Chapter 5, we recapitulate the results achieved in this thesis on CM and we propose several future research directions. All the results on *Centrality Maximization* described in this thesis are summarized in Table 1.1.

Centrality index	**Graph type**	**Inapproximability Upper/Lower bound**	**Approximation algorithms**
Harmonic [27, 28] Chapter 3	*Undirected*	$1 - \frac{1}{15e}$ p. 22, Theorem 3.2	$1 - \frac{1}{e}$ p. 33, Theorem 3.6
	Directed	$1 - \frac{1}{3e}$ p. 20, Theorem 3.1	$1 - \frac{1}{e}$ p. 33, Theorem 3.6
Betweenness [12, 30] Chapter 4	*Undirected*	$1 - \frac{1}{2e}$ p. 58, Theorem 4.2	OPEN
	Directed	$1 - \frac{1}{2e}$ p. 56, Theorem 4.1	$1 - \frac{1}{e}$ p. 65, Theorem 4.5
Eccentricity [32, 79] Chapter 2	Undirected	$\frac{3}{2}$ p. 15, Theorem 2.2	$2 + \frac{1}{OPT}$ $1 + \epsilon$, with $O(k \log \lvert V \rvert)$ edges p. 16, Theorems 2.5, 2.6
	Directed	OPEN	OPEN
Page-rank [4, 71] Chapter 2	*Undirected*	OPEN	OPEN
	Directed	NO FPTAS p. 16, Theorem 2.7	$\left(1 - \alpha^2\right)\left(1 - \frac{1}{e}\right)$ p. 17, Theorem 2.9

Table 1.1: Summary of results on cm problem. In bold our results presented in this thesis.

Chapter 2

Preliminaries

In this section, we give some graph theory definitions and we classify and describe the main centrality metrics. Then, we will survey the main exact and approximated algorithms to compute them. Later on, we will focus on a result on maximization of a class of functions that will be used in the rest of the thesis and we will define the Centrality Maximization problem (CM) and the Centrality Ranking Maximization problem (CRM). Finally, we will describe the state of the art on CM problem.

2.1 Definitions

In the following, we represent complex networks as graphs. Let $G = (V, E)$ be a directed or undirected graph where $|V| = n$ and $|E| = m$. For each node v, if G is directed, N_v^i and N_v^o denote the set of in-neighbours and out-neighbours of v, respectively, i.e. $N_v^i = \{u \mid (u, v) \in E\}$ and $N_v^o = \{u \mid (v, u) \in E\}$. If G is undirected, N_v denotes the set of all neighbours of v, $N_v = \{u \mid \{u, v\} \in E\}$. We denote $|N_v|$ as deg_v, i.e the degree of node v. Similarly, $|N_v^i| = deg_v^i$ and $|N_v^o| = deg_v^o$. Given two nodes s and t, we denote by d_{st}, σ_{st}, and σ_{stv} the distance from s to t in G, the number of shortest paths from s to t in G, and the number of shortest paths from s to t in G that contain v, respectively. If there is no path from u to v, we then set $d_{st} = \infty$. A centrality index induces an ordering of the nodes in V. The *ranking* of a node v according to some centrality index c is the placement of v in the ordering induced by c and it is defined as $r_v^c = |\{u \in V \mid c_u > c_v\}| + 1$.

2.2 Centrality measures

Several *centrality indices* have been proposed in the literature to try to quantitatively capture the notion of importance within a network. Most of the centrality indices are based on distances between nodes, on the number of shortest paths passing through or on spectral properties. In this section we use the classification given by Boldi and Vigna in [17] to introduce the definitions of the main centrality metrics.

2.2.1 Geometric measures

Geometric measures are defined as a function of the distance among nodes. In particular, they depend on how many nodes exist at every distance.

Degree Centrality. The *degree* of a node is the number of incident arcs i.e. the number of nodes at distance one. It is one of the basic measures in complex network analysis. It is important in the studies of popularity and activity of nodes because it is primarily concerned with local point centrality. In a directed network, degree centrality can be assessed for *in-degree* and *out-degree*, where *in* represents other actors connecting to a particular actor and *out* represents that particular actor connecting other actors in the network [94].

Closeness centrality. Defined by Bavelas [9], the *closeness centrality* of a node v is defined as:

$$c_v = \frac{1}{\sum_s d_{sv}}.$$

where d_{sv} is the length of the shortest path from the node s to v. Intuitively, a central node has a greater centrality value as it has a small distance toward most of the other nodes of the network. This index is clearly defined only when the graph is strongly connected (if there are no paths between two nodes s and v, $d_{sv} = \infty$).

Harmonic Centrality. To take into account disconnected pairs of nodes in weakly connected networks, alternative definitions of closeness centrality are introduced. The *harmonic centrality* [17, 63] of a node v is defined as:

$$h_v = \sum_{s \in V \setminus \{v\}} \frac{1}{d_{sv}}.$$

Intuitively, it represents the harmonic mean of the distances from all the other nodes to node v.

Eccentricity. The *Eccentricity* of a node v is defined as

$$e_v = \max_{s \in V} d_{vs},$$

and represents the maximum distance between a node v and every other node in the graph. Using this notion, we can define the diameter D of a network as the maximum eccentricity $D = \max_{w \in v} e_v$ and the radius R as the minimum eccentricity $R = min_{w \in v} e_v$.

2.2.2 Path-based measures

Path based measures take into account not only the existence of a shortest path but all the possible (shortest) paths passing through a single node.

Betweenness centrality. Betweenness centrality was introduced by Freeman [41] and it intuitively measures the influence of a node on the flow circulating through the network, under the assumption that the flow follows shortest paths. For each node v, the *betweenness centrality* of v is defined as

$$b_v = \sum_{\substack{s,t \in V \\ s \neq t; s,t \neq v \\ \sigma_{st} \neq 0}} \frac{\sigma_{stv}}{\sigma_{st}}. \tag{2.1}$$

The node with greater betweenness is an important junction point for the network. There are several contexts in which having a high betweenness can be beneficial for the node itself. For example, in the field of transportation network analysis, the betweenness centrality seems to be positively related to the efficiency of an airport (see [62], where a network of 57 European airports has been analyzed). In a street network, increasing the betweenness of a shop or business would mean more traffic flowing through it and possibly more customers. In social networks, having high betweenness can be extremely beneficial for brokers [38].

2.2.3 Spectral measures

Spectral metrics are computed by extracting eigenvalues and eigenvector of some matrix derived from the graph and they are usually used to assign a certain reputation to a page in Web graphs.

PageRank. PageRank was introduced by Page et al. [72] for ranking web pages. In a directed graph, the *page-rank* of a node v is the probability that a *random surfer walk* that starts at a random node in a graph is at v at a given point in time. A random surfer walk with parameter α, is a walk in the graph defined as follows: start at a random node in G, given by a starting probability distribution; with probability α, move to an edge chosen uniformly at random from those outgoing the current node; with probability $1 - \alpha$, move directly to another node that might be not connected to the current node.

Formally, let us assume that G is a strongly connected directed graph. Let M be a $|V| \times |V|$ matrix where each element m_{uv} is defined as $m_{uv} = \frac{1}{|N_u^o|}$ if $(u, v) \in E$ and $m_{uv} = 0$ otherwise. For a given parameter α, the page-rank is the eigenvector $\bar{p}$ associated to the largest eigenvalue of the matrix

$$Q = \frac{1-\alpha}{|V|}\mathbb{1} + \alpha M.$$

The page rank of a node v is the element p_v in the position associated to v in $\bar{p}$.

2.3 Algorithms for computing centrality measures

Let us consider a graph G stored using adjacency lists [26]. To compute the *degree centrality* of all nodes we visit the entire graph to count the neighbours of each node in $O(n + m)$ time (for a single node, we simply count its neighbours in $O(m)$).

Computing *closeness and harmonic centrality* on unweighted graphs requires to solve All Pairs Shortest Paths problem (APSP): in general the best approach is running a Breadth First Search (BFS) from each node. It requires $O(n(n+m))$ time ($O(n+m)$ for a single node) and is not applicable on large networks (there is a faster algorithm based on fast matrix multiplication working on sparse graphs [96] that solves the APSP in $O(n^{2.3737})$). To overcome this limit, Eppstein and Wang [34] design an approximation algorithms for closeness centrality based on sampling the set of k source node from which running the BFS. With $k = \Theta(\frac{\log(n)}{\epsilon^2})$, the total running time of this algorithm is $O(km)$ within an inverse additive error of ϵD with high probability.

In [15, 16, 24, 25, 48], the authors designed efficient algorithms with a small relative error guarantee, that can handle graphs with billions of edges, using the technique of probabilistic counters introduced by Palmer et al. [73].

Computing the *eccentricity* of all nodes in the graph requires to solve the APSP (for a single nodes, it can be easily computed by a single BFS). Takes and Kosters [88] propose two exact and approximated heuristics in order to compute this metric in large graphs.

The best exact algorithm for computing *betweenness centrality* of all nodes has been designed by Brandes [20]: the core idea is to define dependency $\delta_s(v) = \sum_{t \in V} \frac{\sigma_{stv}}{\sigma_{st}}$ which can be computed in $O(m)$ for each $v \in V$ (the same algorithm is used also to compute the betweenness of a single node). The time complexity is $O(nm)$ and for this reason it is infeasible to run it on large networks. Then, several algorithms exploiting different techniques are developed in order to analyse large graphs. In [36, 85, 92], the authors propose some heuristics with no analytical guarantee. In [46], the authors adapt Eppstein and Wang's approach for computing closeness centrality using Hoeffding's inequality and the union bound technique: they obtain an estimate of the betweenness centrality of every node that is correct up to an additive error λ with probability δ, by sampling $O(\frac{D^2}{\lambda^2} \log \frac{n}{\delta})$ nodes, where D is the diameter of the graph. This approach has been improved by Riondato and Kornaropoulos in [81] and Riondato and Upfal in [82] by sampling single shortest paths instead of the whole dependency of a node, introducing the use of the VC-dimension [91]. This technique decreases the number of sampling to $\frac{c}{\lambda^2}(\lfloor \log_2(D-2) \rfloor + 1 + \log(\frac{1}{\delta}))$, where D is the diameter of the network i.e. the maximum eccentricity. Recently, Borassi and Natale [19] propose a new approximation algorithm based on adaptive sampling technique.

The first algorithm for computing *PageRank* has been proposed by Page et al. [72]. After that, a lot of algorithms are proposed for the exact computation. Kamvar et al [47] propose an iterative linear algebraic approach score: it exploits the fact that, during the computation of PageRank, there are only few nodes taking much longer to converge. They speeds up the computation by nearly 30% on graphs up to 10^6 nodes, avoiding the recomputation of the nodes that converge quickly. Later, Gleich et al [43] propose a parallel iterative method in order to compute PageRank on graphs with billions of nodes and edges. Another parallel approach has been proposed by Kohlschütter et al. [51]. McScherry [64] improves the performance of the sequential algorithm in order to compute the rank on networks with up to 10^7 nodes. Broeder et al. [21] propose an approximation algorithm based on graph aggregation which produces a ranking that has Spearman rank-order correlation of 0.95 with respect to exact PageRank. Chen et. al [22] and Bar-Yossef et al. [6] provide approximation algorithm for the local PageRank i.e compute the score of a target node using only local information provided by a link server [13].

2.4 Maximizing monotone submodular functions

Some of the algorithms reported in this thesis exploit the results of Nemhauser et al. on the approximation of monotone submodular objective functions [69]. A function z defined on subsets of a ground set N, $z : 2^N \to \mathbb{R}$, is *submodular* if the following inequality holds for any pair of sets $S \subseteq T \subseteq N$ and for any element $e \in N \setminus T$

$$z(S \cup \{e\}) - z(S) \geq z(T \cup \{e\}) - z(T).$$

In other words, a submodular function exhibits decreasing marginal gains: the marginal value of adding a new element to a set decreases as the size/cost/weight etc. of the set increases. Let us consider the following optimization problem: given a finite set N, an integer k', and a real-valued function z defined on the set of subsets of N, find a set $S \subseteq N$ such that $|S| \leq k'$ and $z(S)$ is maximum. If z is *monotone and submodular*, then the following greedy algorithm exhibits an approximation of $1 - \frac{1}{e}$ [69]: start with the empty set, and, for k' iterations, add an element that gives the maximal marginal gain, that is if S is a partial solution, choose the element $j \in N \setminus S$ that maximizes $z(S \cup \{j\})$.

Theorem 2.1 ([69]). *For a non-negative, monotone submodular function z, let S be a set of size k obtained by selecting elements one at a time, each time choosing an element that provides the largest marginal increase in the value of z. Then S provides a $\left(1 - \frac{1}{e}\right)$-approximation.*

We will exploit such results by showing that some centrality indices c are monotone and submodular with respect to the possible set of edges incident to a given node v.

2.5 Centrality Maximization problem

Given a set S of edges not in E, we denote by $G(S)$ the graph augmented by adding the edges in S to G, i.e. $G(S) = (V, E \cup S)$. For a parameter x of G, we denote by $x(S)$ the same parameter in graph $G(S)$, e.g. the distance from s to t in $G(S)$ is denoted as $d_{st}(S)$. The centrality index of a node v clearly depends on the graph structure: if we augment a graph by adding a set of edges S incident to v, then the centrality of v might change. Generally speaking, adding edges incident to some node v can increase the centrality of v. We are interested in finding a set S of edges incident to a particular node v that maximizes such an increment. Therefore, given a centrality index c, we define the following optimization problem.

Centrality Maximization	
Given:	A directed or undirected graph $G = (V, E)$; a node $v \in V$; and an integer $k \in \mathbb{N}$
Solution:	A set S of edges incident to v, $S = \{(u, v) \mid u \in V \setminus N_v^i\}$ ($S = \{\{u, v\} \mid u \in V \setminus N_v\}$, if G is undirected), such that $|S| \leq k$
Goal:	Maximize $c_v(S)$

In this thesis we study the Centrality Maximization (CM) problem by using harmonic centrality (CM-H problem) and betweenness centrality (CM-B problem) as metrics. In Section 2.6 we report the results from the literature of (CM) problem by using eccentricity (CMI-E), Page-Rank (CM-P)

2.5.1 Maximum Ranking Improvement problem

Given a centrality index c, maximizing the value of a node v does not necessarily lead to maximizing the ranking position of v. Therefore, we also consider the problem of finding a set S of arcs incident to node v that maximizes *the increment of the ranking* of v with respect to its original ranking. We denote such an increment as $\rho_v^c(S)$, that is,

$$\rho_v^c(S) = r_v^c - r_v^c(S).$$

Informally, $\rho(S)$ represents the number of nodes that v "overtakes" by adding arcs in S to G. Therefore, we define the following optimization problem:

Maximum Ranking Improvement (CRM)	
Given:	A directed or undirected graph $G = (V, E)$; a vertex $v \in V$; and an integer $k \in \mathbb{N}$
Solution:	A set S of edges incident to v, $S = \{(u, v) \mid u \in V \setminus N_v^i\}$ ($S = \{\{u, v\} \mid u \in V \setminus N_v\}$, if G is undirected), such that $\lvert S\rvert \leq k$
Goal:	Maximize $\rho_v^c(S)$

2.6 Related Works

In the literature, there are several algorithms that aim at optimizing some property of a graph by adding a limited number of edges.

Meyerson and Tagiku give a constant factor approximation algorithm for the problem of minimizing the average shortest-path distance between all pairs of nodes [65]. The same problem has been studied by Papagelis et al. [76] and Parotsidis et al. [77], who propose new algorithms and experimentally show that they are good in practice. Bauer et al. [8] study the problem of minimizing the average number of hops in shortest paths of weighted graphs, prove that, unless $P = NP$, the problem cannot be approximated within a logarithmic factor, and propose two approximation algorithms with non-constant approximation guarantees. Tong et al. [89] and Saha et al. [83] study the problem of maximizing the leading eigenvalue of the adjacency matrix and give algorithms with proven approximation guarantees. Bilò et al. [14] and Frati et al. [40] present algorithms with proven approximation guarantees for the problem of minimizing the diameter of a graph. Li et al. [59] and Dehghani et al. [31] propose approximation algorithms with proven guarantees for the problem of making the number of triangles in a graph minimum and maximum, respectively. Demaine et al. [32] and Perumal et al. [79] study the problem of minimizing the maximum eccentricity of a graph (we will focus on it in section 2.6.1). In [75], Papagelis studies the problem of minimizing the characteristic path length. Ishakian et al. [45] study the problem of maximizing some centrality measures related to the number of paths passing through a given node on directed acyclic graphs. Finally, Olsen et al. [71] and Avrachenkov et al. [4] works on improving the importance of a predefined vertex using other centrality measures like PageRank (we will explain more in detail these methods in section 2.6.2).

2.6.1 Eccentricity

We now focus on the CMI-E problem studied by Demaine et al. [32] and Perumal et. al [79]. In this case, a node is considered central if its eccentricity is small, therefore the CMI-E problem is a minimization problem, that is we want to find the set of edges S that, when added to G, minimizes the value of $e_v(S)$, for some given node v.

We first show that, unless $P = NP$, the problem cannot be approximated within a certain constant lower bound, we then give an algorithm that guarantees a constant approximation ratio and an algorithm that guarantees an arbitrarily small approximation ratio if an higher number of edges is allowed.

To derive an approximation hardness result for the undirected case, we make use of the *Set Cover* (in short, SC) problem, which is defined as follows: given a set X, a collection $\mathcal{F}$ of subsets of X, and an integer B, find a sub-collection $\mathcal{F}' \subseteq \mathcal{F}$ such that $\cup_{S_j \in \mathcal{F}'} S_j = X$ and $|\mathcal{F}'| \leq B$. It is known that the set cover problem is NP-hard [42].

Given an instance $(X, \mathcal{F})$ of SC, we compute a graph $G = (V, E)$, where $V = \{v, v'\} \cup \{v_{x_i} \mid x_i \in X\} \cup \{v_{S_j} \mid S_j \in \mathcal{F}\}$ and $E = \{\{v, v'\}\} \cup \{\{v', v_{S_j}\} \mid S_j \in \mathcal{F}\} \cup \{\{v_{x_i}, v_{S_j}\} \mid x_i \in S_j\}$. Initially, the eccentricity of v is equal to 3. We prove that there exists a feasible solution for an instance $I_{\text{SC}} = (X, \mathcal{F})$ of SC if and only if there exists a solution S for the instance $I_{\text{CMI-E}} = (G, v, k)$, where $k = B$, of CMI-E such that $e_v(S) = 2$.

If I_{SC} admits a feasible solution $\mathcal{F}'$, then let us consider the solution $S = \{\{v, v_{S_j}\} \mid S_j \in \mathcal{F}'\}$ to $I_{\text{CMI-E}}$. Since $|\mathcal{F}'| \leq B$, then $|S| \leq k$. Moreover, $\cup_{S_j \in \mathcal{F}'} S_j = X$ and then all the nodes v_{x_i} are at distance 2 to v. Therefore, $e_v(S) = 2$.

Let us now assume that $I_{\text{CMI-E}}$ admits a solution S such that $e_v(S) = 2$, without loss of generality, we can assume that S contains only edges $\{v, v_{S_j}\}$ for some $S_j \in \mathcal{F}$ (see [32]

```
Algorithm 1: Approximation algorithm for CMI-E.
Input  : An undirected graph $G = (V, E)$; a node $v \in V$; and an integer $k \in \mathbb{N}$
Output: Set of edges $S \subseteq \{\{u, v\} \mid u \in V \setminus N_v\}$ such that $|S| \leq k$
1 $S := \emptyset$;
2 $U := \{v\}$;
3 for $i = 1, 2, \ldots, k$ do
4     $u_i := \arg\max_{u \in V} \min_{u_j \in U} d_{uu_j}$;
5     $U := U \cup \{u_i\}$;
6     $S := S \cup \{\{u_i, v\}\}$;
7 return $S$;
```

for details). Let $\mathcal{F}'$ be the solution of SC such that $S_j \in \mathcal{F}'$ if and only if $\{v, v_{S_j}\} \in S$. Since $e_v(s) = 2$, the distance between v and all the nodes v_{x_i} is at most 2 and then for each v_{x_i} there exists an edge $\{v, v_{S_j}\} \in S$ such that $x_i \in S_j$. This implies that $\cup_{S_j \in \mathcal{F}'} S_j = X$. Moreover, since $|S| \leq k$, then $|\mathcal{F}'| \leq B$.

Let us assume that there exists an approximation algorithm A for CMI-E that guarantees an approximation factor $\alpha < \frac{3}{2}$ and let S be the solution obtained by applying algorithm A to $I_{\text{CMI-E}}$ derived from I_{SC}. We have that $e_v(S) < \frac{3}{2}OPT$. This implies that, if $(X, \mathcal{F})$ admits a feasible solution, then $e_v(S) < \frac{3}{2} \cdot 2 = 3$, that is $e_v(S) = 2$; otherwise, if $(X, \mathcal{F})$ does not admit a feasible solution, then $e_v(S) = 3$. Therefore, we can determine whether an instance of SC is feasible or not by means of algorithm A. The next theorem follows.

Theorem 2.2 ([32]). *The CMI-E problem on undirected graphs cannot be approximated within a factor smaller than $\frac{3}{2}$, unless $P = NP$.*

In what follows we describe the algorithm given in [79] to solve the CMI-E problem in undirected graphs. The algorithm is based on a former solution to the problem of minimizing the diameter of a graph by adding a limited number of edges [14].

The algorithm is reported in Algorithm 1 and works as follows: first node v is inserted into a set U, then, a for loop of k iterations is run. At each iteration $i = 1, 2, \ldots, k$, a node u_i that maximizes the minimum distance in G between u_i and a vertex in U is selected and inserted into U. The solution S returned is made of edges that connect nodes u_i in $U \setminus \{v\}$ to v, $S = \{\{u_i, v\} \mid u_i \in U \setminus \{v\}\}$.

To analyze the algorithm, we need some further notation. Let $IS(G)$ be the size of a *maximum independent set* of graph $G = (V, E)$, that is the size of a maximum subset of nodes $V' \subseteq V$ such that no two nodes in V' are joined by an edge in E. Given a subset of nodes $U \subseteq V$, the *radius* of U is defined as $r_U = \min_{x \in V} \max_{u \in U} d_{xu}$. Given a graph G and an integer $d \geq 0$, $G^d = (V, E^d)$ is the graph with the same nodes as G and an edge (x, y) if the distance in G between x and y is at most d.

Let S^* be an optimal solution for the instance of CMI-E and let OPT denote $e_v(S^*)$. The diameter of $G(S^*)$ is at most $2OPT$ and therefore $IS((G(S^*))^{2OPT}) = 1$. The next lemma implies that $IS((G(S^*))^{2OPT}) \geq IS(G^{2OPT}) - |S^*|$.

Lemma 2.3 ([14]). *Let G be a graph and let $d \geq 0$. For each $e \in V \times V \setminus E$, $IS((G(\{e\}))^d) \geq IS(G^d) - 1$.*

It follows that $IS(G^{2OPT}) \leq k + 1$. Let $u_0 = v$. We partition the set of nodes V into $k+1$ clusters $U_0, U_1, \ldots, U_k$ as follows: for each $i = 0, 1, \ldots, k$, a node u belongs to U_i if $d_{uu_i} \leq d_{uu_j}$, for each $j = 0, 1, \ldots, k$, ties are arbitrarily broken in order to form a partition. Sets $U_0, U_1, \ldots, U_k$ are called the *clusters* induced by Algorithm 1. The next lemma implies that, for each $i = 0, 1, \ldots, k$, $r_{U_i} \leq 2OPT$.

Lemma 2.4 ([14]). *Let G be a graph, let $d \geq 0$, and let $U_0, U_1, \ldots, U_k$ be the clusters induced by Algorithm 1 on G. If $IS(G^d) \leq k + 1$, then for each $i = 0, 1, \ldots, k$, $r_{U_i} \leq d$.*

Clearly $|S| \leq k$ and the distance between each node $u \in V$ and v in $G(S)$ is at most $2OPT + 1$ to v, therefore, $e_v(S) \leq 2OPT + 1$. The approximation factor guaranteed by Algorithm 1 is then $2 + \frac{1}{OPT}$.

Theorem 2.5 ([79]). *In undirected graphs, the* CMI-E *problem is approximable within a factor $2 + \frac{1}{OPT}$, where OPT is the value of an optimal solution.*

The next theorem shows that if we allow a number of added edges that is higher than k, then we can obtain a solution that is at most $1 + \epsilon$ far from the optimal solution of the case in which only k additional edges are allowed.

Theorem 2.6 ([32]). *For any $\epsilon > 0$, there exists a polynomial-time algorithm that adds $O(k \log |V|)$ edges to reduce the eccentricity of v to at most $1 + \epsilon$ times the optimum eccentricity for the case in which k additional edges are allowed.*

2.6.2 PageRank

We analyse the CM-P problem studied in [4, 71]. The goal is to find the set of edges S that, when added to G, maximize the PageRank of v, for some given node v (CM-P problem).

First of all, the authors show, in the following theorem, that the problem does not admit a polynomial-time approximation scheme.

Theorem 2.7 ([71]). *The* CM-P *problem does not admit an FPTAS, unless $P = NP$.*

```
Algorithm 2: Approximation algorithm for CM-P.
Input  : An undirected graph $G = (V, E)$; a node $v \in V$; and an integer $k \in \mathbb{N}$
Output: Set of edges $S \subseteq \{\{u, v\} \mid u \in V \setminus N_v\}$ such that $|S| \leq k$
1 $S := \emptyset$;
2 $U := \{v\}$;
3 for $i = 1, 2, \ldots, k$ do
4     $u_i := \arg\max_{u \in V} g_v$ in $G = (V, E \cup S)$;
5     $U := U \cup \{u_i\}$;
6     $S := S \cup \{\{u_i, v\}\}$;
7 return $S$;
```

The proof is quite technical and hence it is omitted here, see [71] for details.

Let M be a $|V| \times |V|$ matrix where each element m_{uv} is defined as $m_{uv} = \frac{1}{|N_u^o|}$ if $(u, v) \in E$ and $m_{uv} = 0$ otherwise. Let I denote the $|V| \times |V|$ identity matrix and let us consider the matrix $Z = (I - \alpha M)^{-1}$. Then, the entry z_{uv} of Z is the expected number of visits to node v for a random surfer walk starting at node u [4]. The value $g_v = \frac{p_v}{z_{vv}}$ is the *overall reachability* of node v from all the other nodes, that is the probability that node v is reached by a random surfer walk that starts at some node u, for all $u \in V$ [71]. Let us consider a variant of the CM-P problem where the function to maximize is g_v and let us denote such problem as CM-G. The next theorem implies that problem CM-G can be approximated by the greedy Algorithm 2 with an approximation factor of $1 - \frac{1}{e}$.

Theorem 2.8 ([71])**.** *In directed graphs, for each vertex v, function g_v is monotone and submodular with respect to any feasible solution for* CM-G*.*

Let S be the solution of the Algorithm 2 for problem CM-G and let $OPT = \frac{p_v^{OPT}}{z_{vv}^{OPT}}$ denote the value of an optimal solution for CM-G. The previous theorem implies that

$$\frac{p_v(S)}{z_{vv}(S)} \geq \left(1 - \frac{1}{e}\right) \frac{p_v^{OPT}}{z_{vv}^{OPT}}.$$

Finally, the next theorem follows by the observation that, for any solution S', $z_{vv}(S') \leq \sum_{i=0}^{\infty} \alpha^{2i} = \frac{1}{1-\alpha^2}$ and $z_{vv}(S') \geq 1$.

Theorem 2.9 ([71])**.** *In directed graphs, the* CM-P *problem is approximable within a factor $\left(1 - \alpha^2\right)\left(1 - \frac{1}{e}\right)$.*

Chapter 3

Centrality Maximization problem for harmonic centrality

In this chapter, we study the problem of CM for harmonic centrality (CM-H). Let us recall that, given a graph G, the harmonic centrality of a node u is defined as $h_u = \sum_{s \in V \setminus \{u\}} \frac{1}{d_{su}}$. We will give hardness of approximation results on directed and undirected networks and analyse several intuitive methods to solve the problem. Then, we propose a greedy approximation algorithm with an almost tight approximation ratio in both directed and undirected cases and we evaluate the performance of all the algorithms on both synthetic and real networks. Moreover, we will also study the approximability of the similar problem of improving the ranking induced by the harmonic centrality instead of the value. Most of the results presented in this chapter are included in [27, 28].

3.1 Problem definition

Given a directed (undirected respectively) graph $G = (V, E)$, a vertex $u \in V$, and an integer k, the *Maximum Harmonic Improvement* (in short, CM-H) problem consists in finding a set S of ingoing edges (incident edges respectively) to u not in E, $S \subseteq \{(u, v) : v \in V \setminus N^i_u\}$ ($S \subseteq \{\{(u, v)\} : v \in V \setminus N_u\}$ in undirected graphs) such that $|S| \leq k$ and $h_u(S)$ is maximum.

3.2 Hardness results

3.2.1 Hardness of approximation for directed graphs

In this section, in order to derive our approximation hardness result for the CM-H problem on directed graphs, we will make use of the *Maximum Set Coverage* (in short, MSC) problem, which is defined as follows: given a ground set X of items, a collection $\mathcal{F} = \{S_1, S_2, \dots S_{|\mathcal{F}|}\}$ of subsets of X, and an integer k, find a sub-collection $\mathcal{F}' \subseteq \mathcal{F}$ such that $|\mathcal{F}'| \leq k$ and $s(\mathcal{F}') = |\cup_{S_i \in \mathcal{F}'} S_i|$ is maximised. It is known that the MSC problem cannot be approximated within a factor greater than $1 - \frac{1}{e}$, unless $P = NP$ [39]. We will now use this result in order to show that the CM-H problem has a polynomial-time constant factor approximation scheme.

Theorem 3.1. *For each $\gamma > 1 - \frac{1}{3e}$, there is no γ-approximation algorithm for the* CM-H *problem in directed graphs, unless $P = NP$.*

Proof. We give an L-reduction with parameters a and b [74]. In detail, we will give a polynomial-time algorithm that transforms any instance I_{MSC} of MSC into an instance $I_{\text{CM-H}}$ of CM-H and a polynomial-time algorithm that transforms any solution S for $I_{\text{CM-H}}$ into a solution $\mathcal{F}'$ for I_{MSC} such that the following two conditions are satisfied for some values a and b:

$$OPT(I_{\text{CM-H}}) \leq aOPT(I_{\text{MSC}}) \tag{3.1}$$

$$OPT(I_{\text{MSC}}) - s(\mathcal{F}') \leq b\left(OPT(I_{\text{CM-H}}) - h_u(S)\right). \tag{3.2}$$

where OPT denotes the optimal value of an instance of an optimization problem. If the above conditions are satisfied and there exists a α-approximation algorithm for CM-H, then there exists a $(1 - ab(1-\alpha))$-approximation algorithm for MSC [74]. Since MSC is hard to approximate within a factor greater than $1 - \frac{1}{e}$, then $1 - ab(1-\alpha) < 1 - \frac{1}{e}$, unless $P = NP$. This implies that $\alpha < 1 - \frac{1}{abe}$.

Given an instance $I_{\text{MSC}} = (X, \mathcal{F}, k)$ of MSC, we define an instance $I_{\text{CM-H}} = (G, u, k)$ of CM-H as follows (see Fig. 3.1): $G = (V, A)$, where $V = \{u\} \cup \{v_{x_i} \mid x_i \in X\} \cup \{v_{S_j} \mid S_j \in \mathcal{F}\}$ and $A = \{(v_{x_i}, v_{S_j}) \mid x_i \in S_j\}$.

Without loss of generality, we can assume that any solution S of CM-H contains only arcs (v_{S_j}, u) for some $S_j \in \mathcal{F}$. In fact, if a solution does not satisfy this property, then we can improve it in polynomial time by repeatedly applying the following rule: if S contains an arc (v_{x_i}, u), for some $x_i \in X$, then exchange such arc with an arc (v_{S_j}, u) such that $(v_{S_j}, u) \notin S$ (note that such an arc must exist, since otherwise $|\mathcal{F}| \leq k$ and I_{MSC} could

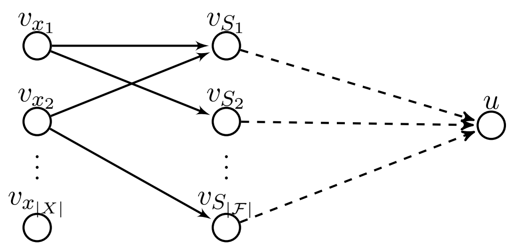


FIGURE 3.1: The reduction used in Theorem 3.2 (in this example, $x_1 \in S_1$, $x_1 \in S_2$, $x_2 \in S_1$, and $x_2 \in S_{|\mathcal{F}|}$). The dashed arcs denote those added in a solution.

be easily solved). The above rule does not decrease the value of $h_u(S)$: indeed, if we exchange an arc (v_{x_i}, u) with an arc (v_{S_j}, u) such that $(v_{S_j}, u) \notin S$, then the harmonic centrality of u decreases by either 1 or $\frac{1}{2}$ (because of the deletion of (v_{x_i}, u)) but certainly increases by at least 1 (because of the insertion of (v_{S_j}, u)).

Given a solution S of CM-H, let $\mathcal{F}'$ be the solution of MSC such that $S_j \in \mathcal{F}'$ if and only if $(v_{S_j}, u) \in S$. We now show that $h_u(S) = \frac{1}{2}s(\mathcal{F}') + k$. To this aim, let us note that the distance from a vertex v_{x_i} to u is equal to 2 if an arc (x_{S_j}, u) such that $x_i \in S_j$ belongs to S, and it is ∞ otherwise. Similarly, the distance from a vertex v_{S_j} to u is equal to 1 if $(x_{S_j}, u) \in S$, and it is ∞ otherwise. Moreover, the set of elements x_i of X such that $d_{v_{x_i}u}(S) < \infty$ is equal to $\{x_i \mid x_i \in S_j \wedge (v_{S_j}, u) \in S\} = \bigcup_{S_j \in \mathcal{F}'} S_j$. Therefore,

$$\begin{aligned} h_u(S) = & \sum_{\substack{v \in V \setminus \{u\} \\ d_{vu}(S) < \infty}} \frac{1}{d_{vu}(S)} = \sum_{\substack{x_i \in X \\ d_{v_{x_i}u}(S) < \infty}} \frac{1}{d_{v_{x_i}u}(S)} + \sum_{\substack{S_j \in \mathcal{F} \\ d_{v_{S_j}u}(S) < \infty}} \frac{1}{d_{v_{S_j}u}(S)} \\ = & \frac{1}{2}|\{x_i \in X \mid d_{v_{x_i}u}(S) < \infty\}| + |\{S_j \in \mathcal{F} \mid d_{v_{S_j}u}(S) < \infty\}| \\ = & \frac{1}{2}\left| \bigcup_{S_j \in \mathcal{F}'} S_j \right| + |\{S_j \mid (v_{S_j}, u) \in S\}| = \frac{1}{2}s(\mathcal{F}') + k. \end{aligned}$$

It follows that Conditions (3.1) and (3.2) are satisfied for $a = \frac{3}{2}$ and $b = 2$. Indeed, $OPT(I_{\text{CM-H}}) = \frac{1}{2}OPT(I_{\text{MSC}}) + k \leq \frac{3}{2}OPT(I_{\text{MSC}})$, where the inequality is due to the fact that $OPT(I_{\text{MSC}}) \geq k$, since otherwise the greedy algorithm would find an optimal solution for I_{MSC}. Moreover, $OPT(I_{\text{MSC}}) - s(\mathcal{F}') = 2\,(OPT(I_{\text{CM-H}}) - k) - 2\,(h_u(S) - k) = 2\,(OPT(I_{\text{CM-H}}) - h_u(S))$. The theorem follows by plugging the values of a and b into $\alpha < 1 - \frac{1}{abe}$. □

We observe that Theorem 3.1 holds also for the related problem in which the edges to be added to the graph are outgoing from u and the harmonic centrality considers distances d_{uv} instead of d_{vu}.

3.2.2 Hardness of approximation for undirected graphs

In this section, in order to derive our approximation hardness result for the CM-H problem on undirected graphs, we will make use of the *Minimum Dominating Set* (in short, MDS) problem, which is defined as follows: given an undirected graph $G = (V, E)$, find a *dominating set* of minimum cardinality, that is, a subset D of V such that $V = D \cup \bigcup_{u \in D} N_u$. It is known that, for any r with $0 < r < 1$, it cannot exist a $(r \ln |V|)$-approximation algorithm for the MDS problem, unless $P = NP$ [33]. We will now use this result in order to show that the CM-H problem does not admit a polynomial-time approximation scheme. To this aim, we will design an algorithm A' that, given an undirected graph $G = (V, E)$ and given the size k of the optimal dominating set of G, by using an approximation algorithm A for the CM-H problem will return a dominating set of G whose approximation ratio is at most $(r \ln |V|)$. Clearly, we do not know the value of k, but we know that this value must be at least 1 and at most $|V|$: hence, we run algorithm A' for each possible value of k, and return the smallest dominating set found. Algorithm A' will run the approximation algorithm A for the CM-H problem multiple times. Each time A will find k nodes $u \in V$ which are the "new" neighbours of the node whose centrality has to be increased: we then add these nodes to the dominating set and create a smaller instance of the CM-H problem (which will contain, among the others, all the nodes in V not yet dominated). We continue until all nodes in V are dominated.

Theorem 3.2. *For each $\gamma > 1 - \frac{1}{15e}$, there is no γ-approximation algorithm for the* CM-H *problem in undirected graphs, unless $P = NP$.*

Proof. We will show that a γ-approximation algorithm A for the CM-H problem, with $\gamma > 1 - \frac{1}{15e}$, would imply a $(r \ln n)$-approximation algorithm A' for the MDS problem, thus proving the theorem. In particular, the algorithm A' is specified in Fig. 3, where k denotes a "guess" of the size of an optimal solution for MDS with input the graph G. In the following, ω will denote the number of times the **while** loop is executed. Since, at each iteration of the loop, we include in the dominating set at most k nodes, at the end of the execution of algorithm A' the set D includes at most $k \cdot \omega$ nodes. Hence, if k is the correct guess of the value of the optimal solution for the MDS instance, then D is a ω-approximate solution for the MDS problem (as we have already noticed, we don't know the correct value of k, but algorithm A' can be executed for any possible value of k, that is, for each $k \in [|V|]$).

The first instruction of the **while** loop of algorithm A' computes a transformed graph G' (to be used as part of the new instance for CM-H) starting from the current graph $G = (V, E_V)$, which is the subgraph of the original graph induced by the set $\{u_1, \ldots, u_n\}$, where $n = |V|$, of still not dominated nodes. This computation is done as follows (see

Algorithm 3: The approximation algorithm for the MDS problem, given a γ-approximation algorithm A for the CM-H problem and a “guess” k for the optimal value of MDS.

```
Algorithm: $A'$
Input : an undirected graph $G = (V, E)$ and an integer $k$
Output: a dominating set $D$
1 $D := \emptyset$;
2 while $V \neq \emptyset$ do
3     Compute graph $G'$ starting from $G$ (see Fig. 3.2);
4     $S := A(G', z, k)$;
5     $D' := \{u : \{z, u\} \in S\}$
6     $D := D \cup D'$;
7     $V := V - D' - \bigcup_{u \in D'} N_u$;
8     $G :=$ subgraph of $G$ induced by $V$;
9 return $D$;
```

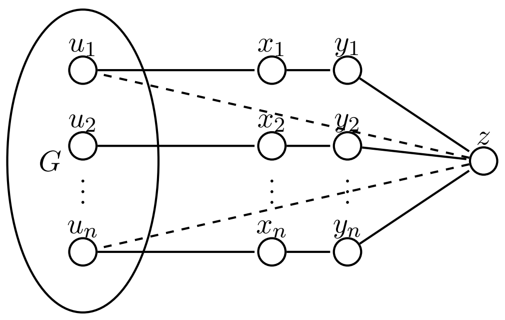


FIGURE 3.2: The reduction used in Theorem 3.2. The dashed edges between node z and nodes u_i denote those added in a solution to CM-H.

Fig. 3.2). We add a new node z and two new nodes x_i and y_i, for each i with $1 \leq i \leq n$. Moreover, we add to E_V the edges $\{z, y_i\}$, $\{x_i, y_i\}$, and $\{x_i, u_i\}$, for each i with $1 \leq i \leq n$. As it is shown in the second line of the **while** loop, z is the node whose centrality h_z has to be increased by adding at most k edges: that is, the CM-H instance is formed by G', z, and k. Observe that any solution for this instance that contains an edge $\{x_i, z\}$ can be modified, without decreasing its measure, by substituting this edge with $\{u_i, z\}$: hence, we can assume that the solution S computed at the second line of the **while** loop of algorithm A' contains only edges connecting z to nodes in V (which are shown by dashed edges between node z and nodes u_i in Fig. 3.2).

First of all, note that, since k is (a guess of) the measure of an optimal solution D^* for MDS with input G, we have that the measure $c^*(G', z, k)$ of an optimal solution S^* for CM-H with input G' satisfies the following inequality:

$$c^*(G', z, k) \geq k + \frac{1}{2}(n-k) + \frac{3}{2}n = \frac{1}{2}k + 2n.$$

This is due to the fact that, by connecting z to all the k nodes in D^*, in the worst case we have that k nodes in G are at distance 1, $n-k$ nodes in G are at distance 2 (since D^* is a dominating set), the n nodes y_i are at distance 1, and the n nodes x_i are at distance 2 from z.

Given the solution S computed by the approximation algorithm A for CM-H, let a and b denote the number of nodes in G at distance 2 and 3, respectively, from z in $G'(S)$. Since all nodes in G' are at distance at most 3 from z, we have that $n = k + a + b$ (we can assume, without loss of generality, that $n \geq k$): hence, $a = n - b - k$. Since A is a γ-approximation algorithm for CM-H, we have that $h_z(S) \geq \gamma c^*(G', z, k)$. That is,

$$k + \frac{1}{2}a + \frac{1}{3}b + \frac{3}{2}n \geq \gamma\left(\frac{1}{2}k + 2n\right).$$

From this inequality, it follows that

$$a \geq \gamma(k + 4n) - 3n - 2k - \frac{2}{3}b.$$

By using the fact that $a = n - b - k$, we have that

$$n - b - k \geq \gamma(k + 4n) - 3n - 2k - \frac{2}{3}b.$$

That is,

$$b \leq 12(1-\gamma)n + 3(1-\gamma)k.$$

Since $k \leq n$, we then have that

$$b \leq 15n(1-\gamma).$$

Assuming $\gamma > 1 - \frac{1}{15e} > \frac{14}{15}$ (which implies $15(1-\gamma) < 1$), then after one iteration of the **while** loop of algorithm A', the number of nodes in G decreases by a factor $15(1-\gamma)$. Hence, after $\omega - 1$ iterations, the number n of nodes in the graph G is at most a fraction $[15(1-\gamma)]^{\omega-1}$ of the number N of nodes in the original graph. Since we can stop as soon as $n < k$, we need to find the maximum value of ω such that $k \leq N[15(1-\gamma)]^{\omega-1}$. By solving this inequality and by recalling that $15(1-\gamma) < 1$, we obtain

$$\omega - 1 \leq \log_{15(1-\gamma)} \frac{k}{N} \leq \log_{15(1-\gamma)} \frac{1}{N} = \frac{\ln(N)}{\ln \frac{1}{15(1-\gamma)}}.$$

One more iteration might be necessary to trivially deal with the remaining nodes, which are less than k. Hence, the total number ω of iterations is at most $\frac{\ln(N)}{\ln \frac{1}{15(1-\gamma)}} + 1$. If $\gamma > 1 - \frac{1}{15e}$, we have that $r' = \frac{1}{\ln \frac{1}{15(1-\gamma)}} < 1$: as a consequence of the observation at the beginning of the proof, the solution reported by algorithm A' is an $(r' \ln N + 1)$-approximate solution. Clearly, for any r with $0 < r' < r < 1$, there exists N_r sufficiently large, such that for any $N > N_r$, $r' \ln N + 1 \leq r \ln N$: hence, algorithm A' would be an

$r \ln N$-approximation algorithm for MDS, and, because of the result of [33], P would be equal to NP. Thus, we have that, if $P \neq NP$, then γ has to be not greater than $1 - \frac{1}{15e}$ and the theorem is proved. $\square$

3.2.3 Improving the position in the ranking

We study the problem of improving the position of a node v in the ranking obtained by sorting all the nodes in non-increasing order according to their harmonic centrality. The *ranking* of a node v according to harmonic centrality is the placement of v in the ordering induced by h and it is defined as $r^h_v = |\{u \in V \mid h_u > h_v\}| + 1$. Given a directed graph $G = (V, E)$, a vertex $u \in V$, and an integer k, the *Maximum Harmonic Ranking Improvement* (in short, CRM-H) problem consists in finding a set S of ingoing edges to u not in E, $S \subseteq \{(u, v) : v \in V \setminus N^i_u\}$ such that $|S| \leq k$ and $\rho^h_v(S)$ is maximum, where $\rho^h_v(S) = r^h_v - r^h_v(S)$. We show that, unless $P = NP$, we cannot find a polynomial time approximation algorithm for CRM-H with a constant approximation guarantee.

Theorem 3.3. *For any constant $\alpha \leq 1$, there is no α-approximation algorithm for the* CRM-H *problem in directed graphs, unless $P = NP$.*

Proof. By contradiction, let us assume that there exists a polynomial time algorithm A that guarantees an approximation factor of α. We show that we can use A to determine whether an instance I of the *exact cover by 3-sets* problem (X3C) admits a feasible solution or not. Problem X3C is known to be NP-complete [42] and therefore this implies a contradiction. In the X3C problem we are given a finite set X with $|X| = 3q$ and a collection C of 3-element subsets of X, with $|C| = c$, and we ask whether C contains an exact cover for X, that is, a subcollection $C' \subseteq C$ such that every element of X occurs in exactly one member of C'. Note that we can assume without loss of generality that $c > q$.

Given an instance $I = (X, C)$ of X3C where $|X| = n = 3q$ and $|C| = c$, we define an instance $I' = (G, v, k)$ of CRM-H as follows.

- $G = (V, E)$;
- $V = \{v\} \cup \{v_{x_i} \mid x_i \in X\} \cup \{v_{T_j} \mid T_j \in C\} \cup \{v_{m_\ell} \mid \ell = 1, 2, \dots, M\} \cup \{z\}$;
- $E = \{(v_{x_i}, v_{T_j}) \mid x_i \in T_j\} \cup \{(v_{m_\ell}, v_{x_i}) \mid x_i \in X,\ \ell = 1, 2, \dots, M\} \cup \{(v_{m_\ell}, v_{T_j}) \mid T_j \in C,\ \ell = 1, 2, \dots, M\} \cup \{(z, v)\}$
- $k = q$;

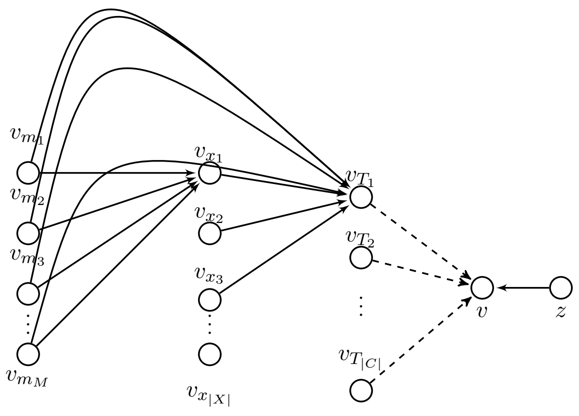


FIGURE 3.3: The reduction used in Theorem 3.3. The dashed arcs denote those added in a solution to CRM-H.

where $M = 5q + 2$. See Figure 3.3 for a visualization.

The proof proceeds by showing that I admits an exact cover if and only if I' admits a solution S such that $\rho^h_v(S) > 0$. This implies that, if OPT is an optimal solution for I', then $\rho^h_v(OPT) > 0$ if and only if I admits an exact cover. Hence, the statement follows by observing that algorithm A outputs a solution S such that $\rho^h_v(S) > \alpha\rho^h_v(OPT)$ and hence $\rho^h_v(S) > 0$ if and only if I admits an exact cover.

In I', $h_v = 1$, $h_{v_{T_j}} = M + 3 = 5q + 5$, for each $T_j \in C$, $h_{v_{x_i}} = M = 5q + 2$ for each $x_i \in X$, and $h_w = 0$, for each other node w.

Therefore, $r^h_{v_{T_j}} = 1$, for each $T_j \in C$, $r^h_{v_{x_i}} = c + 1$, for each $x_i \in X$, $r^h_v = c + n + 1$ and $r^h_w = n + c + 2$, for any other node w. In the proof we will use the observation that, in instance I', adding arcs incident to v does not decrease the harmonic centrality of any node, that is for any node $w \in V$ and for any solution S to I', $h_w(S) \geq h_w$.

If instance I of X3C admits an exact cover C', then consider the solution $S = \{(v_{T_j}, v) \mid T_j \in C'\}$ to I'. Note that $|S| = q = k$ and therefore we only need to show that $\rho^h_v(S) > 0$. Indeed, in the following we show that $\rho^h_v(S) = n > 0$. Since C' is an exact cover, then all nodes v_{T_i} are connected to the 3 nodes v_{x_i} at distance 2. The same holds for nodes v_{T_j} such that $T_j \in C'$. Since there are n nodes v_{x_i} at distance 2, q nodes v_{T_j} such that $T_j \in C'$ at distance 1, and M nodes v_{m_j} at distance 2, then the harmonic centrality of v increases to $h_v(S) = \frac{n}{2} + \frac{M}{2} + q + 1 = 5q + 2$. Any other node does not change its harmonic centrality. Therefore, the only nodes that have a value higher than v are the n nodes v_{T_j}. It follows that $r^h_v(S) = c + 1$ and $\rho^h_v(S) = n + c + 1 - (c + 1) = n > 0$.

Let us now assume that I' admits solution S such that $|S| \leq k$ and $\rho^h_v(S) > 0$. We first prove that S is only made of arcs in the form (v_{T_j}, v) and that $h_v(S) \geq 5q + 2$ or that it can be transformed in polynomial time into a solution with such a form without increasing its size. Assume that S has arcs not in this form, then we can apply one of the following transformations to each of such arcs $e = (w, v)$.

- If $w = v_{x_i}$ for some $x_i \in X$ and there exists a node v_{T_j} such that $x_i \in T_j$ and $(v_{T_j}, v) \notin S$, then remove e and add arc (v_{T_j}, v) to S;
- If $w = v_{x_i}$ for some $x_i \in X$ and $(v_{T_j}, v) \in S$ for all T_j such that $x_i \in T_j$, then remove e and add to S an arc $(v_{T_k}, v) \notin S$;
- if $w = v_{m_\ell}$ for some $\ell = 1, 2, \ldots, M$, then remove e and add to S an arc $(v_{T_k}, v) \notin S$.

Let us denote by S' and S the original solution and the solution that is eventually obtained by applying the above transformations, respectively. All the above transformations remove an arc and possibly add another arc, therefore the size of the transformed solution is at most the original size, that is $|S| \leq |S'| \leq k$. Moreover, the above transformations add only arcs incident to v then, since adding arcs incident to v does not decrease the harmonic centrality of any node then $h_v(S') \geq h_v(S)$. Since v has not outgoing edges, the transformations do not affect the centrality of each other nodes $w \neq v$, then $h_w(S') = h_w(S)$. Therefore, the ranking of node v does not increase then $r^h_v(S') \leq r^h_v(S)$. Since $\rho^h_v(S) = r^h_v - r^h_v(S) > 0$ then $\rho^h_v(S') = r^h_v - r^h_v(S') \geq \rho^h_v(S) > 0$.

It remains to show that $\rho^h_v(S') > 0$ implies $h_v(S) \geq 5q + 2$. Indeed, observe that v is initially in position $n + c + 1$ and the only nodes that have an harmonic value higher than v are the c nodes v_{x_i} and the n nodes v_{T_ℓ}. Therefore, since $\rho^h_v(S') > 0$, and $h_{v_{T_j}} > h_{v_{x_\ell}}$ there is at least a node v_{x_i} such that $h_v(S') \geq h_{v_{x_i}}(S')$. Moreover, all the transformations do not decrease the value of h_v and then $h_v(S) \geq h_v(S')$ and, considering that $h_{v_{x_i}}(S') \geq h_{v_{x_i}} = 5q + 2$, we obtain $h_v(S) \geq 5q + 2$.

We now prove that the solution $C' = \{T_j \mid (v_{T_j}, v) \in S\}$ to I is an exact cover. By contradiction, let us assume that an element in X is not contained in any set in C' or that an element in X is contained in more than one set in C'. The latter case implies the former one since $|C'| = q$, all the sets in C' contain exactly 3 elements, and $|X| = 3q$. Hence, we assume that an element in $|X|$ is not contained in any set in C'. This implies that there exists a node $v_{x_i} \in V$ such that $d_{v_{x_i}v} = \infty$ and therefore the harmonic centrality of v is at most $5q + \frac{3}{2}$, which is a contradiction to $h_v(S) \geq 5q + 2$. □

3.3 Naive algorithms

In this section we list three heuristics to solve the CM-H problem and we prove that they have an unbounded approximation ratio. In particular we analyse the following algorithms:

- RANDOM algorithm: connect u to a set of k nodes extracted uniformly at random.
- DEGREE algorithm: connect u to a set of k nodes having the highest degree.
- TOP-K algorithm: connect u to a set of k nodes having the highest harmonic centrality.

The first two algorithms are easy to describe and implement efficiently. In the next section we will give more details on the implementation of the TOP-K algorithms.

Efficient implementation of TOP-K algorithm. The classical algorithm to find the k nodes having the highest value of centrality, consists, for each node v, in determining all the distances to v by running a Breadth First Search (BFS) and computing h_v. With such an approach computing the k nodes having the highest value of centrality requires $O(n \cdot (n+m))$. We give an algorithm [28] that reduces the computation time by using a branch-and-bound technique that prunes the unnecessary BFS by comparing the intermediate results of centrality with a properly defined upper bound (Algorithm 5).

Let C_k be the set of k nodes having the highest harmonic centrality. We represent C_k with a min-heap in order to find the minimum in constant time. First of all, TOP-K algorithm inserts the k nodes with highest degree in C_k and computes their centrality. Then, it computes the harmonic centrality of other nodes v by performing a BFS starting at each v. The algorithm uses the minimum value of centrality in C_k as a lower bound and prunes the BFS from v when such lower bound is greater than an upper bound (to be defined later) on h_v. Such upper bound is computed every time a node is extracted from the BFS queue. If the BFS is completed without any pruning, it removes the minimum from C_k and it inserts the node v in it.

The upper bound estimates the value of the harmonic centrality of a node v. The main idea is that, at each BFS step, when we extract a node x at distance d_{vx} from v, we can maintain the exact number of nodes that are at distance d_{vx} and that are not visited yet. Moreover, we can upper bound the distance to any other node. When x is extracted from the queue, let V_x be the set of nodes (represented as a queue) at distance d_{vx} from the source v that are not visited, *visited* be the set of nodes currently visited during

Algorithm 4: Algorithm PrunedBFS.
Input : An undirected graph $G(S)$; a node v, a double min_c
Output: c_v

```
1  $V_x := \emptyset$;
2  $d_{uv}(S \cup \{u,v\}) := 0$;
3  foreach $x \in N_v(S)$ do
4      $d_{ux}(S \cup \{u,v\}) := 1$;
5      $V_x.push(x)$;
6  $visited := \{u,v\} \cup N_u(S)$;
7  $h_v := 0$;
8  while $\neg V_x.empty()$ do
9      $x := V_x.pop()$;
10     $h_v := \frac{1}{d_{ux}(S)}$;
11     foreach $y \in N_x(S)$ do
12         if $(y \notin visited) \wedge (d_{uy}(S) > d_{ux}(S \cup \{u,v\} + 1))$ then
13             $d_{uy}(S \cup \{u,v\}) := d_{ux}(S \cup \{u,v\}) + 1$;
14             $V_x.push(y)$;
15             $visited := visited \cup \{y\}$;
16     $UB_v := h_v + |V_x| \cdot \frac{1}{d_{vx}} + (|V| - |visited| - |V_x|) \cdot \frac{1}{d_{vx}+1}$;
17     if $UB_v \leq min_c$ then
18         return
19 return $c_v$
```

the BFS, and $Currc$ be the value of the harmonic centrality at the current step, that is $Currc = \sum_{y \in visited} \frac{1}{d_{vy}}$. Then, we have that $|V_x|$ nodes are at distance d_{vx} from v, while the remaining $|V| - |visited| - |V_x|$ nodes are at distance at least $d_{vx} + 1$ from v. Hence the upper bound is defined as:

$$UB_v = Currc + |V_x| \cdot \frac{1}{d_{vx}} + (|V| - |visited| - |V_x|) \cdot \frac{1}{d_{vx} + 1}.$$

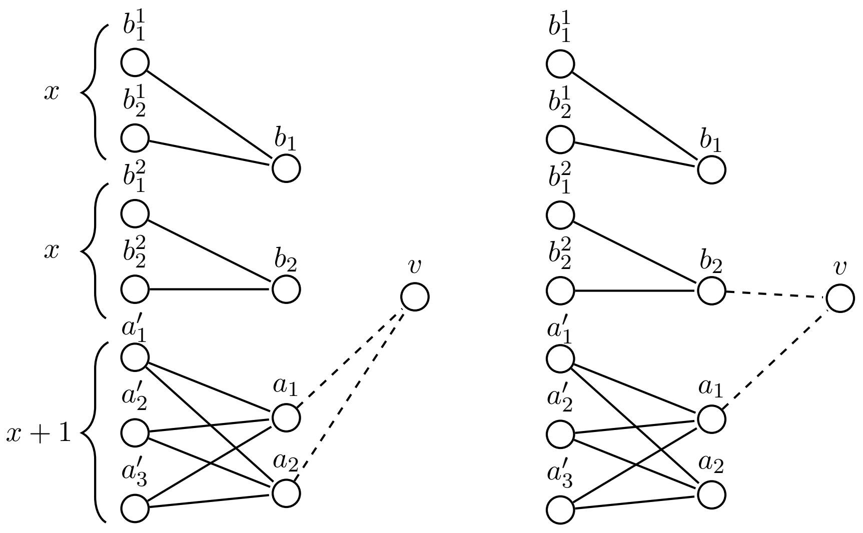
3.3.1 Worst case approximation ratio of the naive algorithms

We now show that the naive algorithms have an arbitrary small approximation ratio. We provide counterexamples for Degree and Top-k algorithms. Such counterexamples are valid also for the Random algorithm because it extracts a set of k nodes uniformly at random that can be equal to the sets returned by Degree or Top-k algorithms with probability $p > 0$. Given an integer $x > 0$, consider the following instance of cm-h (see Figure 3.4 for an example).

- graph $G = (V, E)$.

Algorithm 5: Algorithm TOP-K.
Input : an undirected graph $G = (V, E)$; and an integer $k \in \mathbb{N}$
Output: set of nodes $S \subseteq V$ such that $|S| = k$

```
1  $S$: Min priority queue;
2  Let $u_1, u_2, \ldots, u_{|V|}$ be the nodes of $G$ sorted according to their degree in non-ascending order;
3  for $j = 1, 2, \ldots, k$ do
4      Compute the harmonic centrality $h_{u_j}$ of node $u_j$;
5      $S.push(u_j, h_{u_j})$;
6  for $j = k+1, k+2, \ldots, |V|$ do
7      $min_c := S.getMin()$;
8      $h_{u_j} := PrunedBFS(G, u_j, min_c)$;
9      if $h_{u_j} > min_c$ then
10         $S.pop()$;
11         $S.push(u_j, c_{h_j})$;
12 return $S$
```



(a) DEGREE and TOP-K algorithms (b) Optimal solution

FIGURE 3.4: Counterexample for the naive algorithms for $x = 2$ on undirected graphs. The dashed edges are those in a solution to CM-H. The DEGREE and TOP-K algorithms (left) add the edges $\{a_1, v\}, \{a_2, v\}$ since a_1 and a_2 are the 2 nodes with highest degree and harmonic centrality. After adding such edges, the value of h_v becomes $7/2$. An optimal solution (right), has value $h_v(\{\{a_1, v\}, \{b_2, v\}\}) = 29/6$.

- $V = \{v\} \cup A \cup A' \cup B \cup \bigcup_{z=1}^{x} B'_z$, where $A = \{a_i\}_{i=1}^{x}$, $A' = \{a'_j\}_{j=1}^{x+1}$, $B = \{b_i\}_{i=1}^{x}$, $B'_z = \{b^z_r\}_{r=1}^{x}$;
- $E = \{\{a_i, a'_j\} \mid a_i \in A \ \wedge \ a'_j \in A'\} \cup \{\{b_z, b^j_z\} \mid b_z \in B \ \wedge \ b^j_z \in B'_z\}$;
- $k = x$.

The initial values of harmonic centrality are $h_v = 0$, $h_{a_i} = \frac{3x+1}{2}$, $h_{b_i} = x$, $h_{a'_i} = \frac{3}{2}x$, and $h_{b^z_j} = \frac{x+1}{2}$ for each $z, j = 1, 2, \ldots, x$. On the other hand, the initial degree are $deg_v = 0$, $deg_{a_i} = x + 1$, $deg_{a'_i} = deg_{b_i} = x$, and $deg_{b^i_j} = 1$ for each $i, j = 1, 2, \ldots, x$. Therefore the $k = x$ nodes with the highest harmonic centrality (Top-k algorithm) and the highest degree (Degree algorithm) are the nodes $\{a_i\}_{i=1}^{x}$. The solution obtained with both algorithms has a value $h_v(\{\{a_i\}_{i=1}^{x}, v\}) = \frac{3}{2}x + \frac{1}{2}$, since there are x nodes at distance 1 in A and $x + 1$ nodes at distance 2 in A'. Let j be an integer such that $1 \leq j \leq x$ and $\overline{B} \subset B$, $\overline{B} = \{b_i\}_{i=1}^{x-1}$, the optimal solution include edges $\{\{b_i\}_{i=1}^{x-1}, v\}$ and $\{a_j, v\}$, instead increases h_v by $1 + \frac{x+1}{2} + \frac{x-1}{3} + x - 1 + \frac{x(x-1)}{2} = x + \frac{x^2+1}{2} + \frac{x-1}{3}$, since there are x nodes at distance 1 in $\overline{B} \cup \{a_j\}$ and $x^2 + 1$ nodes at distance 2 in $A \cup \bigcup_{z=1}^{x-1} B'_z$ and $x - 1$ nodes at distance 3 in A. Therefore, in undirected graphs, the approximation ratio of the Degree and Top-k algorithms tends to be arbitrarily small as x increases.

Let us consider the directed case: in Figure 3.4, we modify the set of edges in as following:

$$E = \{(a'_j, a_i) \mid a'_j \in A' \ \wedge \ a_i \in A\} \cup \{(b^z_i, b_i) \mid b^j_i \in B'_z \ \wedge \ b_i \in B\}$$

In this case, the initial values of harmonic centrality are $h_v = 0$, $h_{a_i} = x + 1$, $h_{b_i} = x$, $h_{a'_i} = 0$, and $h_{b^z_j} = 0$ for each $z, j = 1, 2, \ldots, x$. On the other hand, the initial degree are $deg^i_v = 0$, $deg^i_{a_i} = x + 1$, $deg^i_{b_i} = x$, and $deg_{a'_i} = deg^i_{b^i_j} = 0$ for each $i, j = 1, 2, \ldots, x$. Therefore the $k = x$ nodes with the highest harmonic centrality (Top-k algorithm) and the highest degree (Degree algorithm) are the nodes $\{a_i\}_{i=1}^{x}$. As in the undirected case, the solution obtained with both algorithms has a value $h_v(\{\{a_i\}_{i=1}^{x}, v\}) = \frac{3}{2}x + \frac{1}{2}$, since there are x nodes at distance 1 in A and $x + 1$ nodes at distance 2 in A'. Let j be an integer such that $1 \leq j \leq x$ and $\overline{B} \subset B$, $\overline{B} = \{b_i\}_{i=1}^{x-1}$, the optimal solution include edges $\{\{b_i\}_{i=1}^{x-1}, v\}$ and $\{a_j, v\}$, instead increases h_v by $x + \frac{x^2+1}{2}$, since there are x nodes at distance 1 in $\overline{B} \cup \{a_j\}$ and $x^2 + 1$ nodes at distance 2 in $A \cup \bigcup_{z=1}^{x-1} B'_z$. Therefore, in directed graphs, the approximation ratio of the Degree and Top-k algorithms tends to be arbitrarily small as x increases.

3.4 Greedy approximation algorithm

Let us consider the following optimisation problem. Given a set X and an integer k, find a subset Y of X of cardinality at most k that maximises the value $f(Y)$, where $f : 2^X \to \mathbb{N}$ is a specific *objective function*. As we show in Section 2.4 in Chapter 2, if f is *monotone submodular*, that is, if, for any pair of sets $S \subseteq T \subseteq X$ and for any element $e \in X \setminus T$, $f(S \cup \{e\}) - f(S) \geq f(T \cup \{e\}) - f(T)$, then the following greedy algorithm approximates the above problem within a factor $1 - \frac{1}{e}$ [69]: start with the empty set, and repeatedly add an element that gives the maximal marginal gain. In this section, we exploit this result by showing that h_u is monotone and submodular with respect to the possible set of edges incident to u. Hence, Algorithm 6 provides a $\left(1-\frac{1}{e}\right)$-approximation. Note that the computational complexity of such algorithm is $O(k \cdot n \cdot g(n, m+k))$, where $g(n, m+k)$ is the complexity of computing h_u in a graph with n nodes and $m+k$ edges.

Theorem 3.4. *Given a directed graph G, for each vertex v, function h_v is monotone and submodular with respect to any feasible solution for* CM-H.

Proof. To show that h_v is monotone increasing, as shown in [17], it is enough to observe that for each solution S to CM-H, each vertex u such that $(u,v) \notin E \cup S$, and each $s \in V \setminus \{v\}$ such that $d_{sv}(S \cup \{(u,v)\}) \neq \infty$, then $d_{sv}(S \cup \{(u,v)\}) \leq d_{sv}(S)$ and therefore $\frac{1}{d_{sv}(S \cup \{(u,v)\})} \geq \frac{1}{d_{sv}(S)}$. We now show that for each pair S and T of solutions to CM-H such that $S \subseteq T$ and for each vertex u such that $(u,v) \notin T \cup E$,

$$h_v(S \cup \{(u,v)\}) - h_v(S) \geq h_v(T \cup \{(u,v)\}) - h_v(T).$$

To simplify notation, we assume that $\frac{1}{d_{st}(X)} = 0$ whenever $d_{st}(X) = \infty$, for any solution X to CM-H. We prove that each term of h_v is submodular, that is, that, for each vertex $s \in V \setminus \{v\}$ such that $d_{sv}(T \cup \{(u,v)\}) \neq \infty$, we show that

$$\frac{1}{d_{sv}(S \cup \{(u,v)\})} - \frac{1}{d_{sv}(S)} \geq \frac{1}{d_{sv}(T \cup \{(u,v)\})} - \frac{1}{d_{sv}(T)}. \tag{3.3}$$

Let us consider the shortest paths from s to v in $G(T \cup \{(u,v)\})$. The following two cases can arise:

1. The last edge of a shortest path from s to v in $G(T \cup \{(u,v)\})$ is (u,v) or belongs to $S \cup E$. In this case, such a path is a shortest path also in $G(S \cup \{(u,v)\})$, as it cannot contain edges in $T \setminus S$. Then, $d_{sv}(S \cup \{(u,v)\}) = d_{sv}(T \cup \{(u,v)\})$ and $\frac{1}{d_{sv}(S \cup \{(u,v)\})} = \frac{1}{d_{sv}(T \cup \{(u,v)\})}$. Moreover, $d_{sv}(S) \geq d_{sv}(T)$ and, therefore, $-\frac{1}{d_{sv}(S)} \geq -\frac{1}{d_{sv}(T)}$.
2. The last edge of all shortest paths from s to v in $G(T \cup \{(u,v)\})$ belongs to $T \setminus S$. In this case, $d_{sv}(T) = d_{sv}(T \cup \{(u,v)\})$ and, therefore, $\frac{1}{d_{sv}(T \cup \{(u,v)\})} - \frac{1}{d_{sv}(T)} = 0$. As $\frac{1}{d_{sv}(S)}$ is monotone increasing, then $\frac{1}{d_{sv}(S \cup \{(u,v)\})} - \frac{1}{d_{sv}(S)} \geq 0$.

In both cases, we have that the inequality (3.3) is satisfied and, hence, the theorem follows. □

The proof of Theorem 3.4 can be easily adapted to the undirected graph case, and the following result holds.

Algorithm 6: Greedy algorithm for CM-H on directed graphs.

```
Input  : A directed graph $G = (V, E)$; a node $v \in V$; and an integer $k \in \mathbb{N}$
Output: Set of edges $S \subseteq \{(u, v) \mid u \in V \setminus N_v^i\}$ such that $|S| \leq k$
1 $S := \emptyset$;
2 for $i = 1, 2, \ldots, k$ do
3     foreach $u \in V \setminus N_v^i(S)$ do
4         Compute $h_v(S \cup \{(u, v)\})$;
5     $u_{\max} := \arg\max\{h_v(S \cup \{(u, v)\}) \mid u \in V \setminus N_v^i(S)\}$;
6     $S := S \cup \{(u_{\max}, v)\}$;
7 return $S$;
```

Theorem 3.5. *Given an undirected graph G, for each vertex u, function h_u is monotone and submodular with respect to any feasible solution for* CM-H.

Corollary 3.6. *The* CM-H *problem is approximable within a factor $\left(1 - \frac{1}{e}\right)$ on both directed and undirected graphs.*

In Section 3.5 we will analyse the performance, in terms of solution quality, of the greedy algorithm on relatively small real-world and synthetic graphs.

3.4.1 Improving the greedy algorithm running time

In this section we show how to improve the running time of GreedyImprovement on directed graph case. Indeed, the algorithm can be easily adapted to the undirected graph case. This algorithm requires $O(k \cdot n \cdot g(n, m+k))$ computational time, where $g(n, m+k)$ is the complexity of computing h_u in a graph with n nodes and $m+k$ edges. The classical algorithm to compute h_u consists in determining all the distances to u by running a BFS starting from u. Therefore, with such an approach, GreedyImprovement requires $O(k \cdot n \cdot (n + m + k))$ in the worst case. In this section we provide a dynamic algorithm to reduce the time required to compute h_u. Note that the idea of incrementally updating the closeness centrality as been already explored in the literature [84]. However, we consider the harmonic mean to compute the harmonic centrality instead of the arithmetic mean that is used in other works in literature. The motivation is that the harmonic mean has been showed to be more robust in the case of undirected disconnected networks or directed not-strongly connected networks [17]. Therefore, we cannot directly use the algorithms in the literature and we devise a new dynamic algorithm. Furthermore, we show how to exploit the submodularity of h_u in order to reduce the running time of iterations $i \geq 2$ of the **for** loop at line 2 of GreedyImprovement.

Let us assume that we add an edge $\{u, v\} \notin E \cup S$ to graph $G(S)$. The dynamic algorithm aims at computing only the distances between u and any other node that change as a consequence of the addition of edge $\{u, v\}$ (i.e. nodes w such that $d_{uw}(S) \neq d_{uw}(S \cup \{u, v\})$) and keep the old distances to any other node in the graph. The algorithm is based on the following observation: if we add an edge $\{u, v\}$ to $G(S)$, then $d_{uw}(S) \neq d_{uw}(S \cup \{u, v\})$, for some $w \in V$, only if the shortest path between u and w in $G(S \cup \{u, v\})$ contains edge $\{u, v\}$. Therefore, we can determine the nodes that change their distance to u by finding all the shortest paths passing through edge $\{u, v\}$ in $G(S \cup \{u, v\})$. To this aim, the dynamic algorithm executes a BFS starting from node v and prunes the search as soon as a node that does not change its distance to u is extracted from the queue. We report the dynamic algorithm *DynamicBFS* in Figure 7. In detail, Procedure *DynamicBFS* returns the value ΔClo which corresponds to the increment to $h_u(S)$ which is obtained by adding edge $\{u, v\}$. To compute ΔClo, the algorithm computes the distances between u and any node y such that $d_{uy}(S) \neq d_{uy}(S \cup \{u, v\})$. First, it computes the distances of u and its neighbors (lines 3–5) and the initial increment ΔClo that is equal to the difference between the reciprocal of the new distance and that of the old distance (line 8). Then, it pushes in queue Q the neighbors of u (line 6) and performs the BFS starting from v (lines 9–16).

For each extracted node, it updates ΔClo by subtracting the reciprocal of the old distance and adding the new one (line 11). After that, it enqueues a neighbor y of the extracted node x only if the old distance $d_{uy}(S)$ is greater than the length of the path made of the shortest path from u to x in $G(S \cup \{u, v\})$ and the edge $\{x, y\}$ (that is $d_{ux}(S \cup \{u, v\}) + 1$, see the test at line 13). Note that this condition is satisfied only if the shortest path between u and y passes through edge $\{u, v\}$. The procedure repeats this process until the queue is empty.

We give an example of execution of Algorithm *DynamicBFS* in Figure 3.5.

In order to analyse the computational complexity of Algorithm *DynamicBFS*, let us define as $\gamma_{uv}(S)$ as the set of nodes that change their distance to u as a consequence of the addition of edge $\{u, v\}$ to $G(S)$, that is

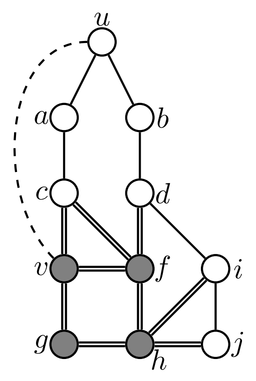
$$\gamma_{uv}(S) = \{w \in V \mid d_{xu}(S) \neq d_{xu}(S \cup \{\{u, v\}\})\}.$$

Moreover, let $\Gamma_{uv}(S)$ be the number of edges incident to nodes in $\gamma_{uv}(S)$, that is $\Gamma_{uv}(S) = \sum_{w \in \gamma_{uv}(S)} |N(w)|$. Parameters $|\gamma_{uv}(S)|$ and $\Gamma_{uv}(S)$ measure the minimal number of nodes and edges, respectively, that must be visited in order to update all the distance to u after the addition of edge $\{u, v\}$. Note that $\Gamma_{uv}(S) = O(m + n)$ in the worst case, however it is much smaller than m in many practical cases as shown in the next section. In Figure 3.5, the nodes in $\gamma_{uv}(S)$ are represented in gray, while the number of double edges is $\Gamma_{uv}(S)$. The next theorem gives the computational complexity of Algorithm *DynamicBFS* as a function of $O(\Gamma_{uv}(S))$.

```
Algorithm 7: Algorithm $DynamicBFS$.
Input  : An undirected graph $G(S)$; edge $\{u,v\}$; distances $d_{ux}(S)$, for each $x \in V$
Output: $\Delta Clo$, the increment to $c_u$ obtained when adding edge $\{u,v\}$ to $G(S)$
1  $Q := \emptyset$;
2  $visited := \emptyset$;
3  $d_{uv}(S \cup \{u,v\}) := 1$;
4  foreach $x \in N_v(S)$ do
5      $d_{ux}(S \cup \{u,v\}) := \min\{2, d_{ux}(S)\}$;
6      $Q.push(x)$;
7  $visited := \{u,v\} \cup N_u(S)$;
8  $\Delta Clo := 1 - \frac{1}{d_{uv}(S)}$;
9  while $\neg Q.empty()$ do
10     $x := Q.pop()$;
11     $\Delta Clo := \Delta Clo + \frac{1}{d_{ux}(S \cup \{u,v\})(x)} - \frac{1}{d_{ux}(S)}$;
12     foreach $y \in N_x(S)$ do
13         if $(y \notin visited) \wedge (d_{uy}(S) > d_{ux}(S \cup \{u,v\}) + 1)$ then
14             $d_{uy}(S \cup \{u,v\}) := d_{ux}(S \cup \{u,v\}) + 1$;
15             $Q.push(y)$;
16             $visited := visited \cup \{y\}$;
17 return $\Delta Clo$
```

Theorem 3.7. *Algorithm DynamicBFS requires $O(\Gamma_{uv}(S))$ time.*

Proof. Lines 1–6 require $O(N_v(S)) = O(\Gamma_{uv}(S))$ time. In the loop at lines 9–16, variable *visited* ensures that each node is inserted into Q at most once. Therefore, the overall time requirement of such loop is equal to the sum of $N_x(S)$, for all the nodes x that are inserted into Q. Hence, to prove the statement, we show that all the nodes inserted into Q belong to $\gamma_{uv}(S)$. We first show that, for each $x \in \gamma_{uv}(S)$ all the distances $d_{xu}(S \cup \{\{u,v\}\})$ between u and x in $G(S \cup \{\{u,v\}\})$ are correctly computed by Algorithm *DynamicBFS*. By contradiction, suppose that the distance between some node in $\gamma_{uv}(S)$ and u is not correctly computed and consider a node $y \in \gamma_{uv}(S)$ having minimal distance to u among such nodes. At the last iteration when y is inserted into Q, there exists a node $x \in N(y)$ such that $d_{uy}(S) > d_{ux}(S \cup \{u,v\}) + 1$. It follows that $d_{uy}(S \cup \{u,v\}) = d_{ux}(S \cup \{u,v\}) + 1$ (see the test at line 13). Since the distance between y and u is minimal among those that are not correctly computed by the algorithm, then $d_{ux}(S \cup \{u,v\})$ is correct. It follows that the distance between y and u is correctly computed at line 14, a contradiction. By contradiction, suppose that some node not in $\gamma_{uv}(S)$ is inserted into Q and consider a node $y \notin \gamma_{uv}(S)$ having minimal distance to u among such nodes. Since y has minimal distance to u among the nodes not in $\gamma_{uv}(S)$ inserted into Q, then the node x for which the condition at line 13 is satisfied when y is inserted into Q must belong to $\gamma_{uv}(S)$. By the previous arguments, $d_{ux}(S \cup \{u,v\})$ is correctly computed by the algorithm and then



(a) Graph G, the dashed edge is the newly added edge $\{u, v\}$, gray nodes and double edges are visited by Algorithm *DynamicBFS*.

Iter.	**Node extracted from** Q	ΔClo	Q
0	v	2/3	(c, f, g)
1	c	2/3	(f, g)
2	f	5/6	(g, h, d)
3	g	13/12	(h, d)
4	h	7/6	(d, i, j)
5	d	7/6	(i, j)
6	i	7/6	(j)
7	j	7/6	$\emptyset$

(b) Iterations of the algorithm: the second column is the node extracted from Q, the last two columns represent the status of ΔClo and Q at the end of the iteration. Iteration 0 corresponds to lines 1–6 of Algorithm *DynamicBFS*.

x	d_{xu}	$d_{xu}(\{u, v\})$
u	0	0
a	1	1
b	1	1
c	2	2
d	2	2
v	3	1
f	3	2
g	4	2
h	4	3
i	3	3
j	4	4

(c) Distances before and after the edge addition.

FIGURE 3.5: Example of execution of Algorithm *DynamicBFS*.

$d_{uy}(S \cup \{u, v\}) = d_{ux}(S \cup \{u, v\}) + 1 < d_{uy}(S)$, a contradiction to the fact that y does not belong to $\gamma_{uv}(S)$. □

The new dynamic algorithm can now be obtained by the GREEDYIMPROVEMENT shown in Figure 6, by doing the following modifications.

- Before line 2, we compute h_u in G.
- At line 4, we incrementally compute $h_u(S \cup \{u, v\})$ by making use of algorithm *DynamicBFS* instead of a full BFS.

Note that, for each $v \in V$, $\Gamma_{uv}(S)$ is maximized when $S = \emptyset$. By contradiction, suppose that there exists a set S' such that $\Gamma_{uv}(S') > \Gamma_{uv}(\emptyset)$. This means that there exists at least a node x such that $d_{ux}(S' \cup \{u, v\}) < d_{ux}(S')$ and $d_{ux}(\{u, v\}) = d_{ux}(\emptyset)$. Note that if $d_{ux}(S' \cup \{u, v\}) < d_{ux}(S')$ the addiction of $\{u, v\}$ decreases the distance between u and x.

Since, by definition, all the edges in S' are incident to u, the shortest path between u and x pass through $\{u, v\}$ and do not traverse any edges in S'. It follows that, for each $S'' \subseteq S'$, $d_{ux}(S' \cup \{u, v\}) = d_{ux}(S'' \cup \{u, v\})$ and $d_{ux}(S') \leq d_{ux}(S'')$ then $d_{ux}(S'' \cup \{u, v\}) < d_{ux}(S'')$. Since $\emptyset \subseteq S'$ then $d_{ux}(\{u, v\}) < d_{ux}(\emptyset)$, a contradiction. Therefore, the algorithm requires an overall $O(k \cdot n\Gamma)$ computational time, where $\Gamma = \max_{v \in V}\{\Gamma_{uv}(\emptyset)\}$.

We now show how to exploit the definition of submodularity to reduce the running time of iterations $i \geq 2$ of the **for** loop at line 2 of GreedyImprovement. The idea of speeding up the greedy algorithm through this *lazy evaluation* was originally proposed by [67]. The same idea was exploited in [56] in order to solve the problem of detecting outbreaks in a network. Let $\Delta h_u(S \cup \{\{u, v\}\})$ be the increment to the centrality of node u after adding the edge $\{u, v\}$ to graph $G(S)$. Since h_u is submodular, then $\Delta h_u(S \cup \{\{u, v\}\})$ is monotonic non-increasing. It follows that $\Delta h_u(S \cup \{\{u, v\}\})$ is upper bounded by $\Delta h_u(S' \cup \{\{u, v\}\})$, where $S' \subseteq S$. We exploit this observation in algorithm DynamicGreedyImprovement given in Figure 8.

Algorithm 8: Algorithm DynamicGreedyImprovement.
Input : An Undirected graph $G = (V, E)$; a vertex $v \in V$; and an integer $k \in \mathbb{N}$
Output: Set of edges $S \subseteq \{\{u, v\} \mid u \in V \setminus N_v\}$ such that $|S| \leq k$

```
1  Compute $c_u$ by using full BFS;
2  foreach $v \in V \setminus N_u$ do
3      $\Delta h_v(\{\{u, v\}\}) := 0$;
4  $S := \emptyset$;
5  $S' := \emptyset$;
6  for $i = 1, 2, \ldots, k$ do
7      $LB := 0$;
8      foreach $v \in V \setminus N_u(S)$ do
9          if $(i = 1) \vee (LB < \Delta h_v(S' \cup \{\{u, v\}\}))$ then
10             $\Delta h_v(S \cup \{\{u, v\}\}) := DynamicBFS(G(S), \{u, v\}, \{d_{ux}(S)\}_{x \in V})$;
11             if $\Delta h_v(S \cup \{\{u, v\}\}) > LB$ then
12                 $LB := \Delta h_v(S \cup \{\{u, v\}\})$;
13                 $max := v$;
14     $S' := S$;
15     $S := S \cup \{\{u, max\}\}$;
16     Compute distances $d_{ux}(S)$, for each $x \in V$;
17 return $S$
```

First, we compute h_u and initialize Δh_u (lines 1–3). For each iteration i of the **for** loop at lines 6-16, we use the variable LB (line 7) to maintain the maximum improvement to harmonic found so far (adding the edge $\{u, max\}$), that is LB is a lower bound to the improvement that will be found at the end of iteration i. If at iteration $i \geq 2$, for some node $v \in V \setminus N_u(S)$, we have that $LB \geq \Delta h_u(\{S' \cup \{\{u, v\}\}\})$ (line 9), where S' is the value of S at iteration $i-1$, then edge $\{u,v\}$ cannot increase the value of h_u more than the maximum found so far. Therefore, in this case we prune the search. Otherwise, we compute $\Delta h_u(S \cup \{\{u,v\}\})$ and check whether it improves LB or not (line 11). In the affirmative case, we update LB (line 12). The DynamicGreedyImprovement algorithm can be adapted to the directed case running the (pruned) BFSs on the transpose graph of G to reduce the time required to compute c_u and to improve the computational complexity of GreedyImprovement.

We can improve the performance of DynamicGreedyImprovement by means of two further heuristics. First, we sort the nodes of $N_v(S)$, for each $v \in V$, in non-increasing order of distance from u and we stop the **for** loop of lines 12-16 of algorithm $DynamicBFS$ when a node y such that $d_{uy}(S) \leq d_{ux}(S \cup \{u,v\}) + 1$ is extracted. In fact, for any other node adjacent to x with a distance to u greater than $d_{uy}(S)$ the condition at line 13 is not satisfied. Then, we can easily parallelize algorithm DynamicGreedyImprovement over p processors since $V \setminus (N_v(S))$ can be divided into sets of $\left\lfloor \frac{|V \setminus (N_v(S))|}{p} \right\rfloor$ nodes and the **for** loop at lines 8-13 of algorithm DynamicGreedyImprovement can be executed in parallel for each set. In this case, LB is given by the maximum over each subset.

3.5 Experimental results

In this section we analyse the greedy algorithm from an experimental point of view. First, we compare the solution of the greedy algorithm with the optimal solution computed by using an integer program formulation of the cm-h problem, in order to assess its real performance in terms of solution quality. Then, we compare the greedy algorithm with the naive algorithms described in Section 3.3 (in Section 3.3 we propose an efficient implementation of Top-k algorithm). Moreover, we show the results of our experiments on real-world graphs by measuring the improvement in the value of harmonic centrality and in the ranking.

All our experiments have been performed on a computer equipped with two Intel Xeon E5-2643 CPUs, each with 6 cores clocked at 3.4GHz and 128GB of main memory, and our programs have been implemented in C++ (gcc compiler v4.8.2 with optimization level O3).

3.5.1 Directed graphs

We now analyse the performance of the greedy algorithm on both synthetic and real world directed networks.

Evaluating the solution quality. We measured the approximation ratio of the greedy algorithm on five types of randomly generated directed networks, namely directed Preferential Attachment (in short, PA) [18], Erdős-Rényi (in short, ER) [35], Copying (in short, COPY) [52], Compressible Web (in short, COMP) [23] and Forest Fire (in short, FF) [55]. The size of the graphs is reported in Table 3.1. For each combination (n, m), we generated five random directed graphs. We focused our attention on twenty vertices u (also called pivots), which have been chosen on the basis of their original harmonic centrality ranking. In particular, we have divided the list of vertices, sorted by their original ranking, in four intervals, and chosen five random vertices uniformly at random in each interval: we denote by $u_{X\%}$ the average value of the vertices in the interval of the top Xth percentile. The value of k ranged from 1 to 10. In the experiments, we measured the ratio between the value of the solution found by the greedy algorithm and the optimal value computed by using the integer programming formulation of the CM-H problem, defined as follows.

$$\begin{array}{lll}
\text{Maximize} & \displaystyle\sum_{\substack{s\in V\setminus\{u\}\\ v\in V\setminus N_u}} \left(\frac{1}{d_{su}((u,v))} - \frac{1}{d_{su}}\right) y_{sv} + \sum_{s\in V\setminus\{u\}} \frac{1}{d_{su}} & \\
\text{subject to} & \displaystyle\sum_{v\in V\setminus N_u} y_{sv} \leq 1, & \forall\ s \in V\setminus\{u\} \\
& y_{sv} \leq x_v, & \forall\ s\in V\setminus\{u\}, v\in V\setminus N_u, \\
& \displaystyle\sum_{v\in V\setminus N_u} x_v \leq k, & \\
& x_v, y_{sv} \in \{0,1\}, & \forall\ s\in V\setminus\{u\}, v\in V\setminus N_u.
\end{array}$$

The decision variables x_v and y_{sv} specify a solution S of the CM-H problem as follows. For any $v \in V \setminus N_u$,

$$x_v = \begin{cases} 1 & \text{if } (u,v)\in S, \\ 0 & \text{otherwise,} \end{cases}$$

and, for each $s \in V \setminus \{u\}$ and $v \in V \setminus N_u$,

$$y_{sv} = \begin{cases} 1 & \text{if a shortest path from } s \text{ to } u \text{ in } G((u,v)) \text{ passes through edge } \{u,v\}, \\ 0 & \text{otherwise.} \end{cases}$$

The first constraint of the integer program ensures that each node s can be covered by at most one edge (u,v) and, hence, that the distance from s to u is counted only once in the objective function, while the second constraint ensures that if $y_{sv} = 1$, then $x_v = 1$ and, hence, that the shortest path from s to u passing through $\{u,v\}$ is considered only

Network	$n = \|V\|$	$m = \|E\|$	**Min Approx. Ratio**
PA	100	130	0.9816
PA	500	650	0.9956
PA	1000	1300	1
ER	100	200	0.9668
ER	100	500	0.9744
ER	100	1000	0.9780
ER	500	5000	0.9890
ER	500	12500	0.9819
ER	500	25000	0.9994
COMP	100	200	0.9968
COMP	100	500	0.9764
COMP	100	1000	1
COMP	500	5000	0.9848
COMP	500	12500	1
COMP	500	25000	1
COPY	100	200	0.9911
COPY	100	500	0.9753
COPY	100	1000	0.9820
COPY	500	5000	0.9825
COPY	500	12500	0.9726
COPY	500	25000	0.9690
FF	100	200	0.9911
FF	200	400	0.9714
FF	500	1000	0.9892

TABLE 3.1: Comparison between the GREEDYIMPROVEMENT algorithm and the optimum. The first three columns report the type and size of the graphs; the fourth column reports the approximation ratio.

if $\{u, v\} \in S$. Finally, note that in the objective function, the value of $\frac{1}{d_{su}((u,v))}$ and $\frac{1}{d_{su}}$ can be preprocessed, and that the term $\sum_{s \in V \setminus \{u\}} \frac{1}{d_{su}}$ is a constant. We solved the above integer program by using the GLPK solver [2].

The results are reported in Table 3.1 where we show the minimum (i.e. worst-case) approximation ratio obtained by the greedy algorithm. The experiments clearly show that the experimental approximation ratio is by far better than the theoretical one proven in Corollary 3.6. In fact, in the worst case the ratio is 0.9668. Notice that it is not possible to process graphs with more than 600 nodes with the exact method.

The analysis of the improvement in the value and in the ranking. We used real-world citation networks obtained from the Arnetminer database [1] (see Table 3.2 for details). In the Arnetminer's networks, there is a vertex for each author and an arc from vertex x to vertex y if the author corresponding to vertex x cited in his paper one paper written by the author corresponding to y. We parsed the Arnetminer database in

Network	$n = \|V\|$	$m = \|E\|$	Time ($k = 10$) [s]
Software Engineering	3141	14787	0.01
Information Security	1067	4253	0.87
Computer Graphics Multimedia	8336	41925	1.79
Theoretical Computer Science	4172	14272	0.43
Artificial Intelligence	27617	268460	0.01
High-Performance Computing	4869	35036	0.04
Computer Networks	9420	53003	0.10
Interdisciplinary Studies	577	1504	0.10

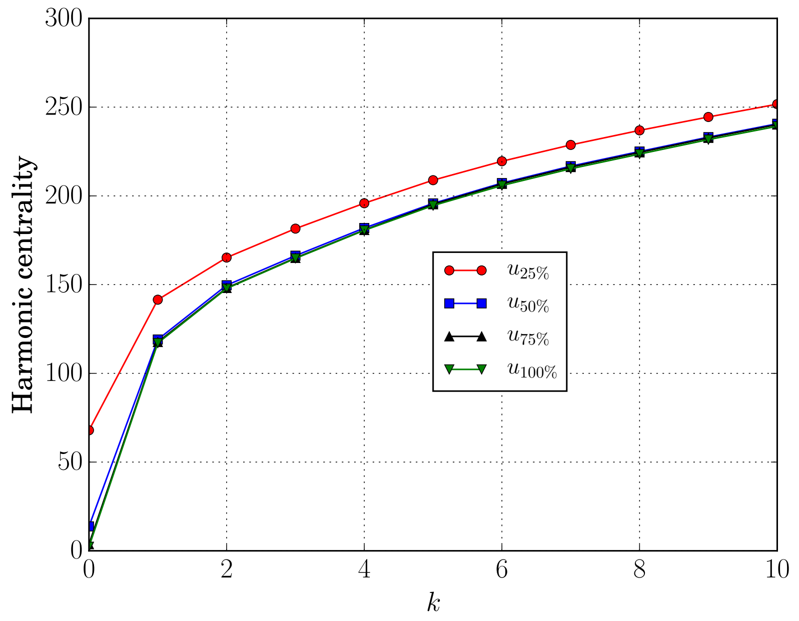
TABLE 3.2: Collaboration networks obtained from Arnetminer database and running time of the DYNAMICGREEDYIMPROVEMENT algorithm with $k = 10$.

order to select a sub-network induced by the authors that published at least a paper in one of the main conferences or journals. As in the previous experiment, for each graph, we used twenty vertices as u. The value of k ranges from 1 to 10.

The results for the citation network Information Security are plotted in Fig. 3.6. In the two charts we plot the harmonic centrality and the ranking of vertex u as a function of k. We observe that any vertex becomes central by adding just few arcs. For example a vertex with the smallest harmonic centrality which initially has harmonic 0 and is ranked 509, improves its value and ranking to 213.32 and 1, respectively, by adding only 7 arcs.

The comparison with naive algorithms. In the chart in Fig. 3.7 we compare the greedy algorithm with the other approaches. We report the comparison of the average percentage ranking position i.e. the ranks multiplied by $\frac{100}{n}$, since n is the maximum rank a node can have in a graph with n nodes reached by the nodes. Notice that each point represents the average over the 10 pivots. The experiments show that the greedy algorithm outperforms the other approaches. Similar results hold if we use the harmonic value instead of the ranking position to compare the greedy algorithm against the other approaches.

We further used the DYNAMICGREEDYIMPROVEMENT algorithm to analyse a web network [54]. The results for the web network uk-2007 ($n = 100000$, $m = 3050615$) are plotted in Fig. 3.8. In the right chart we report the execution time of the algorithm. The DYNAMICGREEDYIMPROVEMENT algorithm is up to 10^3 times faster than the basic GREEDYIMPROVEMENT algorithm and for all the iteration it visits only the 0.18% of the arcs of the graph: using the DYNAMICGREEDYIMPROVEMENT algorithm, it is possible to solve the CM-H problem on very large graphs where it is impossible to obtain a solution using the GREEDYIMPROVEMENT algorithm.



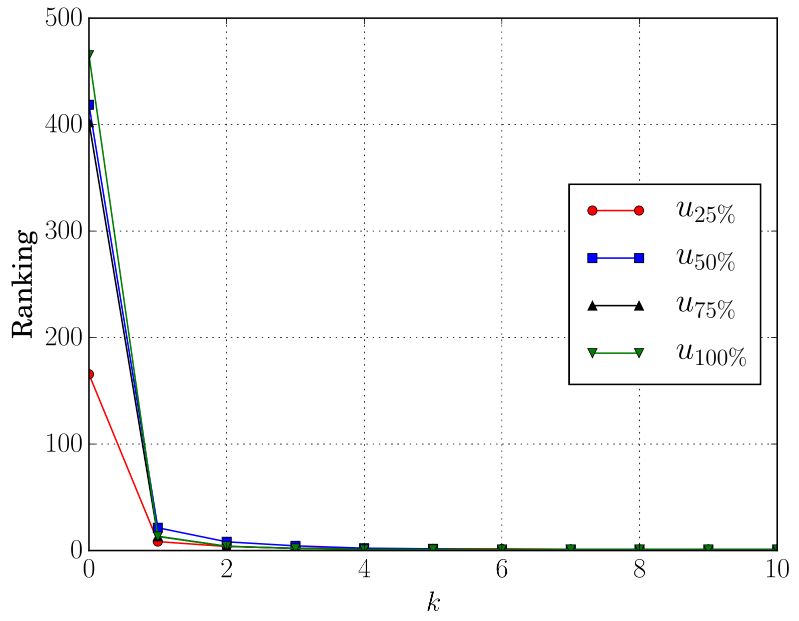


FIGURE 3.6: Performance of the DIRECTEDGREEDYIMPROVEMENT algorithm on network Information Security.

The analysis of the parallel algorithm. In order to test the scalability of the parallelized version of DYNAMICGREEDYIMPROVEMENT algorithm, we run the same set of experiments with different numbers of cores and we measured the execution time and the speedup i.e. the ratio between the execution time with 1 core and the execution time with p cores for $p = 2, 4, 6, 8$. The results are reported in Table 3.3. We notice that the parallel algorithm shows a good scalability in terms of execution time.

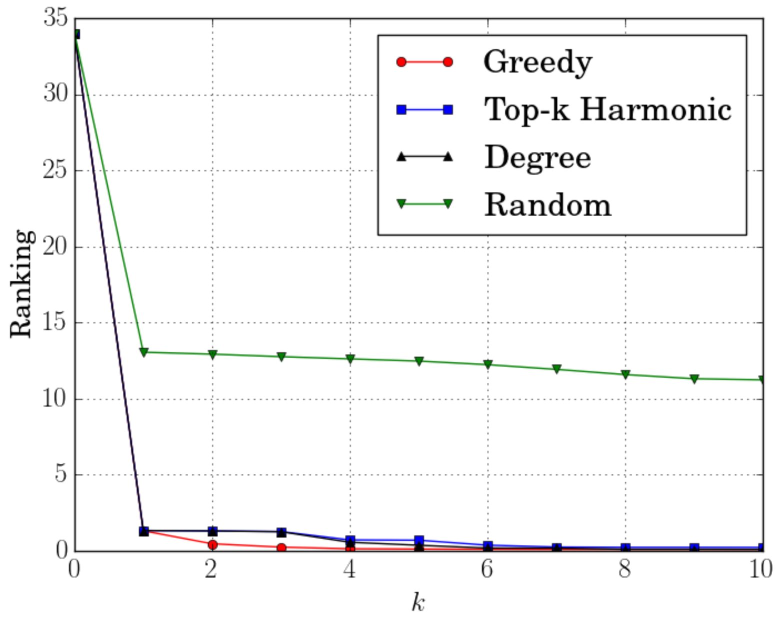


FIGURE 3.7: Average percentage ranking as a function of the number k of inserted edges for the four heuristics for the Information_security network.

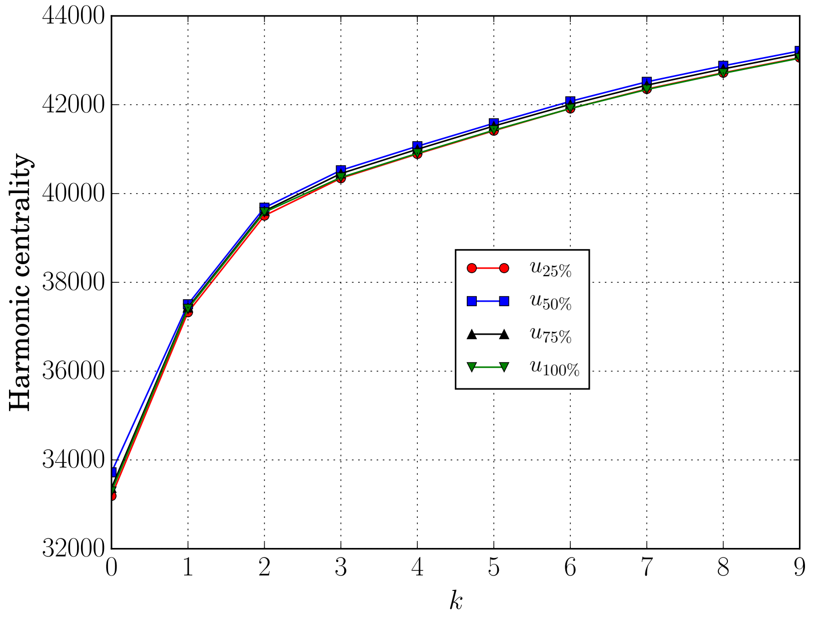
Network	Avg. Execution Time (1 core)	Avg. Speedup (2 cores)	Avg. Speedup (4 cores)	Avg. Speedup (6 cores)	Avg. Speedup (8 cores)
uk-2007	1382 s	1,83	3,70	5,58	7,44

TABLE 3.3: Execution time and speedup of DYNAMICGREEDYIMPROVEMENT algorithm on uk-2007 network with different number of cores.

3.5.2 Undirected graphs

We now focus on the performance of the greedy algorithm on both synthetic and real world undirected networks.

The evaluation of the solution quality. In this section we evaluate the quality of the solution produced by the greedy algorithm by measuring its approximation ratio on several, relatively small, randomly generated networks and on four real-world networks. In particular, we considered four random graph generating models, that is, undirected Preferential Attachment (in short, PA) [7], Erdős-Rényi (in short, ER) [35], Configuration Model (in short, CM) [11, 68], and Watts-Strogatz model (in short WS) [95]. The size of the generated graphs is reported in Table 3.4. For each combination (n, m), we generated five random undirected graphs. Moreover, we considered the four real-world graphs, whose size is reported in Table 3.5. The first graph is the collaboration network between Jazz musicians that have played together in a band, and it has been obtained



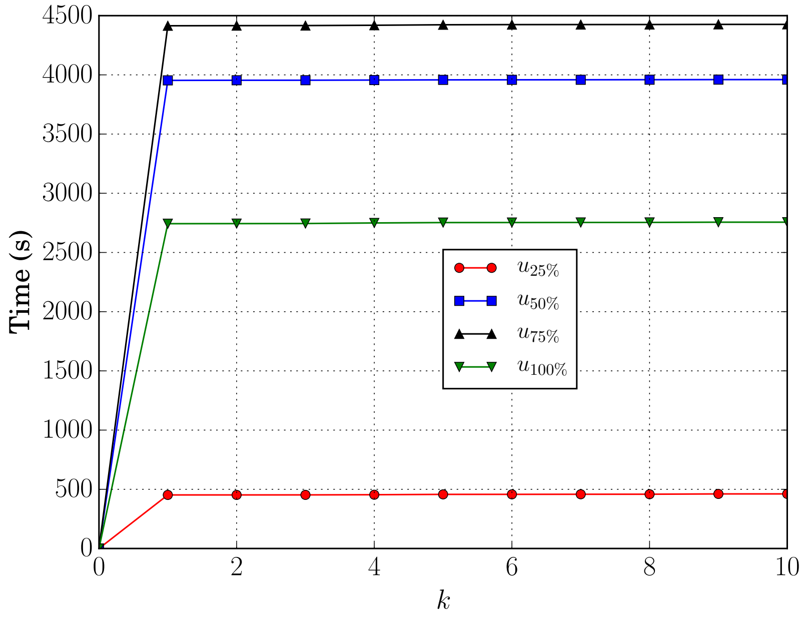


Figure 3.8: Performance of the DynamicGreedyImprovement algorithm on network uk-2007.

from the Konect database [53], while the last three graphs have been downloaded from the Uri AlonLab [90] database: in particular, s838_st is an electronic network, while the other two graphs are biological networks.

We show, similarly to the directed graph case, the minimum ratio obtained. The experiments show that in the worst case the ratio is 0.9798.

Network	$n = \|V\|$	$m = \|E\|$	**Min Approx.**
PA	100	130	0.9939
PA	500	650	0.9921
ER	100	200	0.9828
ER	100	500	0.9938
ER	100	1000	0.9970
ER	500	5000	0.9971
ER	500	12500	0.9991
ER	500	25000	1
CM	100	200	0.9946
CM	500	1000	0.9995
WS	100	500	0.9798
WS	100	600	0.9798
WS	100	800	0.9856
WS	100	1200	0.9946

Table 3.4: Comparison between the GreedyImprovement algorithm and the optimum in random graphs. The first three columns reports the type and size of the graphs; the fourth column reports the minimum measured approximation ratio.

Network	$n = \|V\|$	$m = \|E\|$	**Min Approx.**
s838_ st	512	819	0.9862
jazz	198	2742	0.9968
coli1	328	456	0.9947
celegans_metabolic	346	1493	0.9981

Table 3.5: Comparison between the GreedyImprovement algorithm and the optimum in real-world graphs. The first three columns reports the name and size of the graphs; the fourth column reports the minimum measured approximation ratio.

In Fig. 3.9 and Fig. 3.10, instead, we plot the average harmonic centrality and ranking of vertices u as a function of k in a small real-world network, namely the s838_st electrical network. We observe that the charts on the top, where the values are computed using the GreedyImprovement algorithm, and the charts on the bottom, in which we used the optimal algorithm, are almost identical. Indeed, the approximation ratio in the worst case is 0.9862: that is, the GreedyImprovement algorithm performs very well in practice.

Finally, we tested our algorithm on several artificial instances generated by the Erdős-Rényi and the Watts-Strogatz models with $n = 100$ and $n = 500$. In the former model we can choose appropriate values of the graph density, while in the latter one we can choose the clustering coefficient. It turned out that the performance of our algorithm is not influenced by these two factors. Indeed, the approximation ratio ranges in $[0.9798, 1]$ and improves a little when the density is very high (i.e. $m > 0.5n^2$).

The analysis of the improvement in the value and in the ranking. We measured the improvement in the value of harmonic centrality of u and in the harmonic ranking of u within the network. In particular, we studied two large real-world networks obtained from the DBLP [58] and IMDB database [44]. In such networks, the nodes are authors or actors and there is an edge connecting vertex x and vertex y if the author, or actor, corresponding to vertex x collaborated with y for writing a paper or for acting in the same movie. For each graph, we used twenty pivots as u but, differently from the experiments on random graphs, these vertices have been chosen on the basis of their degree ranking: in particular, we divided the list of vertices sorted by their ranking in 4 parts and chosen randomly five vertices for each interval. The value of k ranges from 1 to 10.

The analysis of these two large networks has been possible only by using the DynamicGreedyImprovement algorithm, since this algorithm visits only few edges of the graph (as explained in the previous section): in particular, for all the iterations $i = 1, 2, \ldots, k$ the algorithm visits only 0.09% of the edges. The results for the DBLP network ($n = 1305445$, $m = 6108727$) are plotted in Fig. 3.11, while the results for the IMDB network ($n = 1797446$, $m = 72880156$) are similar and are shown in Fig. 3.12. In the chart on the left we plot the harmonic centrality of vertex u as a function of k. We observe that any vertex improves its harmonic value by adding just few edges. In order to provide a lower bound on the value of the harmonic centrality we compute the diameter D using the iFUB algorithm [29]: the lower bound LB_h is equal to $1 + \frac{n-D}{D}$. In DBLP network, $D = 23$ and $LB_h = 56758$ and in IMDB network $D = 14$ and $LB_h = 128389$. In the right chart we plot the execution time of the algorithm DynamicGreedyImprovement. We notice that the computational effort is high for $k = 1$ but then it is almost constant for $k > 1$: this is due to the submodularity property.

The great difference in execution time between different nodes is due to the dynamic algorithm: indeed the most a node has a low degree the most it is likely that, adding and edge incident to a central node u, can affect the majority of the distances between u and the other nodes (running the dynamic algorithms on nodes with small degree has complexity $O(\Gamma_{uv}(S)) \sim O(m + n)$). Due to the fact that the degree distribution in the big networks analysed is not uniform, the difference between the execution time of the between $u_{25\%}$ and $u_{50\%}$, $u_{50\%}$ and $u_{75\%}$, $u_{75\%}$ and $u_{100\%}$ is not proportional to the difference of the positions in the initial ranking.

The comparison with naive algorithms. In order to compare the solution obtained by the GreedyImprovement algorithm with that obtained by using the other aforementioned approaches, described in Section 3.3, we run all the algorithms on several real-world networks reported in Tables 3.5 and 3.7. In what follows we compare

Network	Avg. Execution Time (1 core)	Avg. Speedup (2 cores)	Avg. Speedup (4 cores)	Avg. Speedup (6 cores)	Avg. Speedup (8 cores)
DBLP	17671 s	1.91	3.01	4.01	4.63
IMDB	10710 s	1.57	2.10	3.12	3.27

Table 3.6: Execution time and speedup of DynamicGreedyImprovement algorithm on DBLP and IMDB networks with different number of cores.

Network	$n = \|V\|$	$m = \|E\|$	Time ($k = 10$) [s]
advogato	5272	45903	0.07
ca-AstroPh	17903	196972	4.32
ca-CondMat	21363	91286	3.86
ca-HepPh	11204	117619	0.88
ca-HepTh	8638	24806	0.44
dip20090126	19928	41202	8.10
Newman-Cond_mat_95-99	22015	58578	3.19
PGPgiantcompo	10680	24316	1.23

Table 3.7: Real-world graphs used in the comparison between the GreedyImprovement algorithm and the other baselines and running time of the DynamicGreedyImprovement algorithm with $k = 10$.

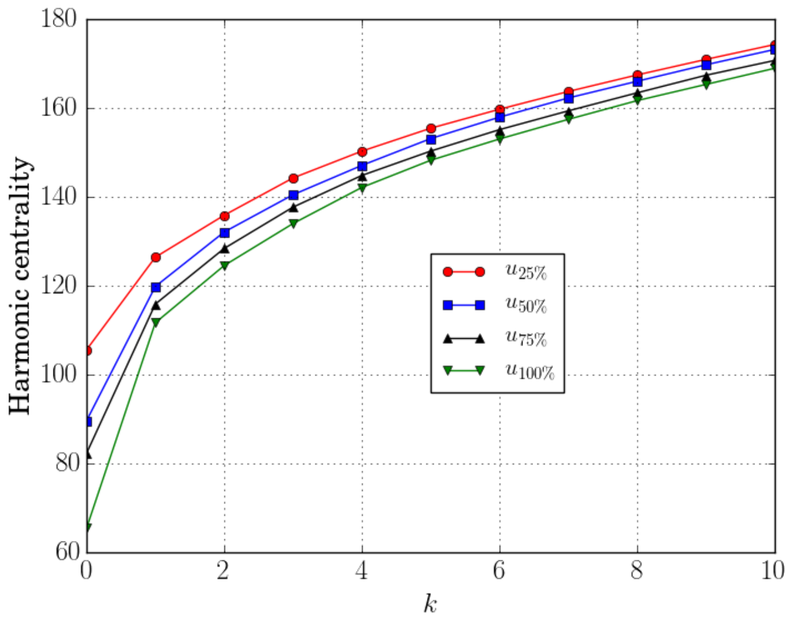
GreedyImprovement with Degree, Random, and Top-k on network ca-HepPh, which is a well known collaboration network obtained from the SNAP database [57]. The results for the other networks are similar.

Regarding the greedy algorithm, in Fig. 3.13 we plot the harmonic centrality and the ranking of vertex u as a function of k on network ca-HepPh. We observe that any vertex becomes central by adding just few edges. In Fig. 3.14, we compare the ranking obtained with the solution given by our algorithm with that obtained with the solution given by the other approaches on the same network. In particular, we show the average percentage ranking as in the directed graph case. We observe that the greedy algorithm significantly outperforms Random, Degree, and Top-k, whenever $k > 1$.

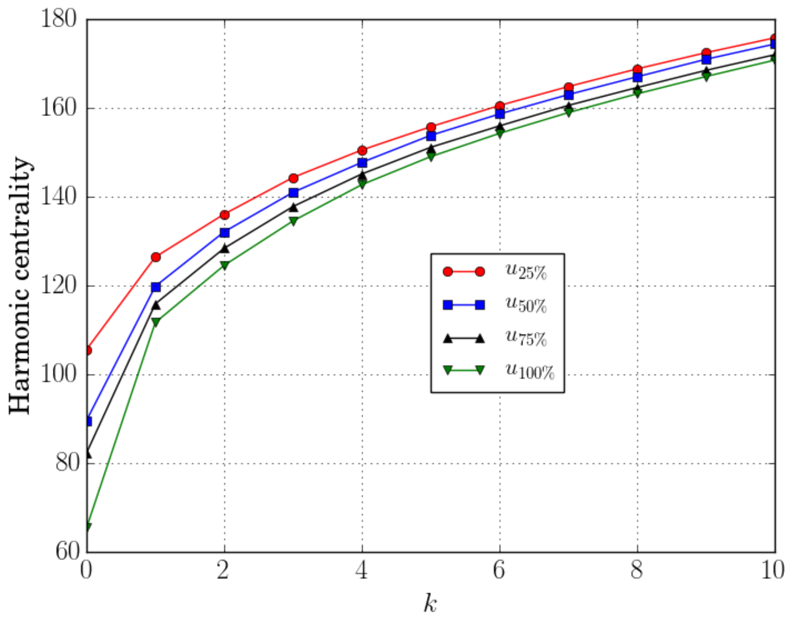
We confirm that our algorithm is by far better than the other approaches.

We observe that similar results hold if we use the harmonic value instead of the ranking position to compare the greedy algorithm against the other baselines.

The analysis of the parallel algorithm. Like in the directed case, we run the same set of experiments with different numbers of cores and we measured the execution time and the speedup i.e. the ratio between the execution time with 1 core and the execution time with p cores for $p = 2, 4, 6, 8$. The results are reported in Table 3.6. We notice that the parallel algorithm shows a good scalability in terms of execution time. Note that the small increase in the case of 8 cores is due to the fact that in our machine, each CPU has 6 physical cores and hence in the case of 8 cores the computation is performed by two different processors.

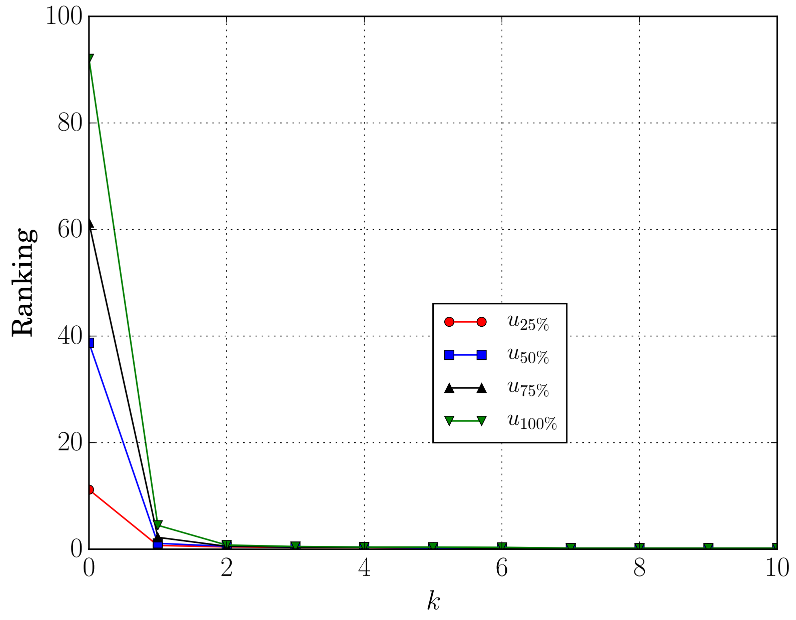


(a) GreedyImprovement algorithm.

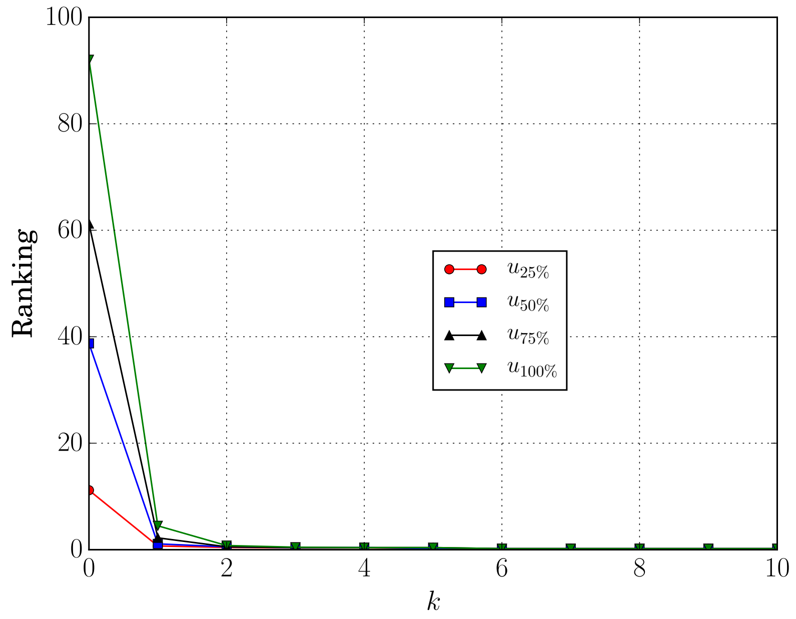


(b) Optimal algorithm.

Figure 3.9: Harmonic centrality of vertices in the four intervals u as a function of k in the network s838_st. Comparison between the GreedyImprovement algorithm and the optimal one.



(a) GreedyImprovement algorithm.



(b) Optimal algorithm.

Figure 3.10: Ranking of vertices in the four intervals u as a function of k in the network s838_st. Comparison between the GreedyImprovement algorithm and the optimal one.

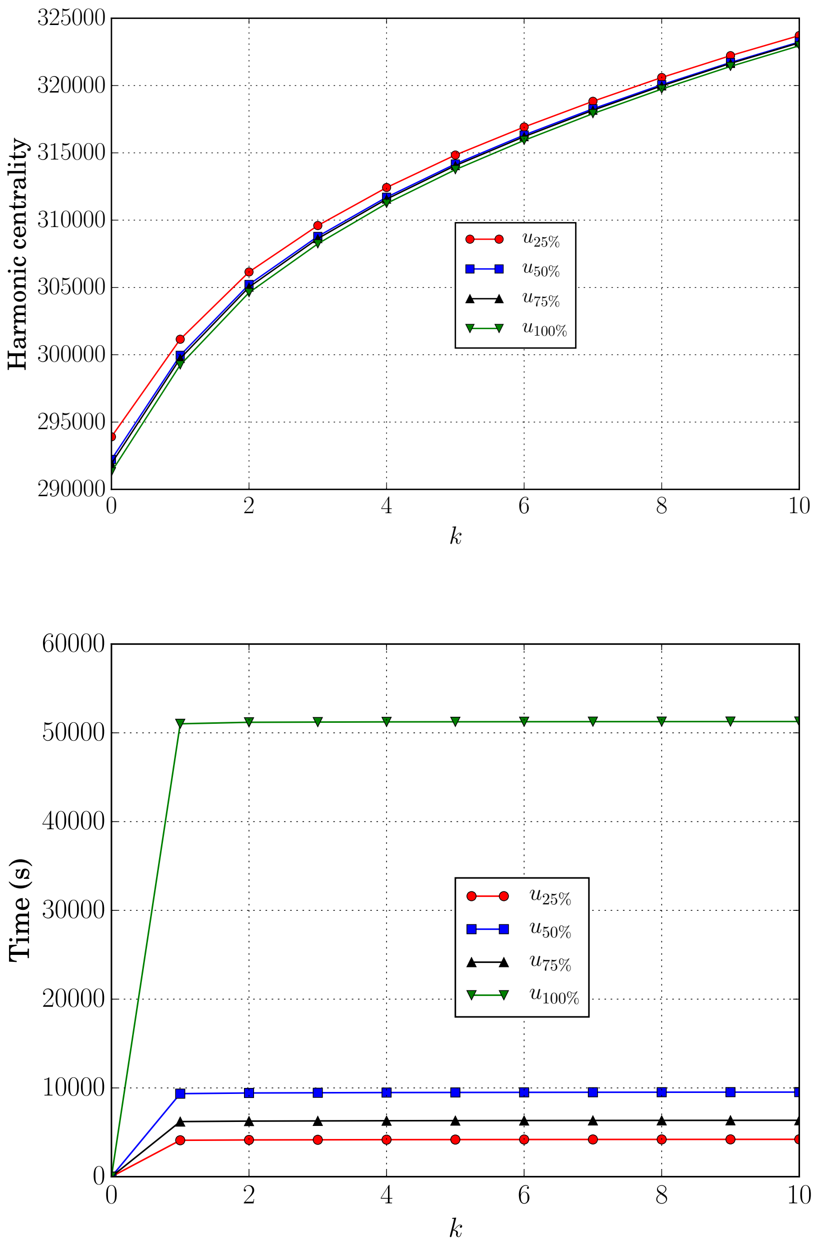
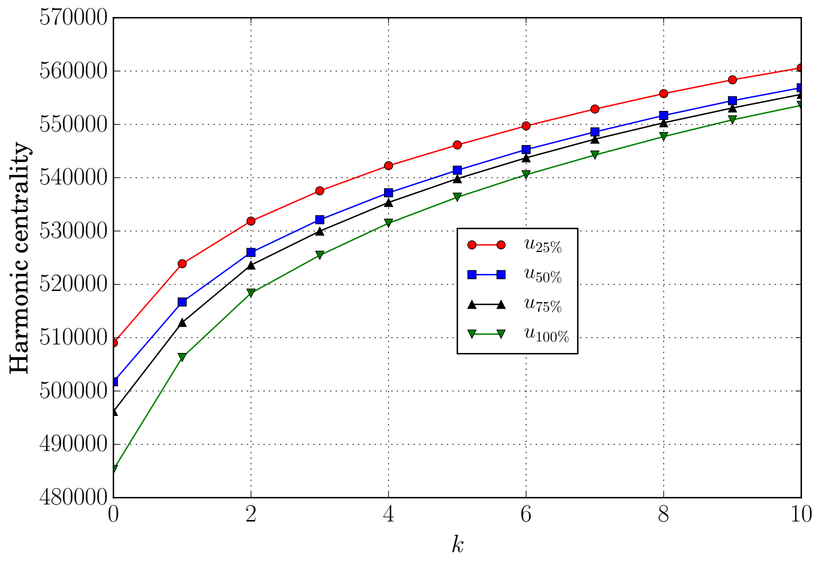


FIGURE 3.11: Performance of DYNAMICGREEDYIMPROVEMENT algorithm on network DBLP.



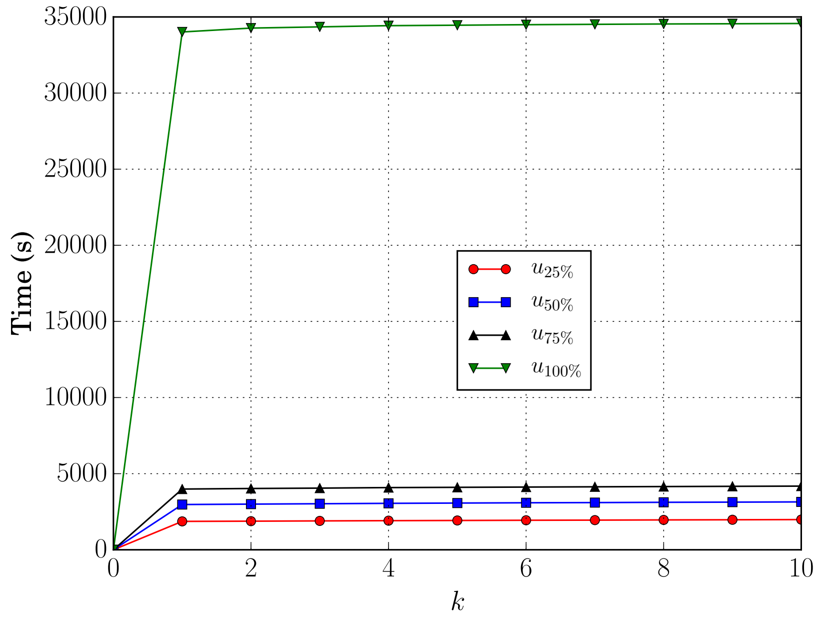
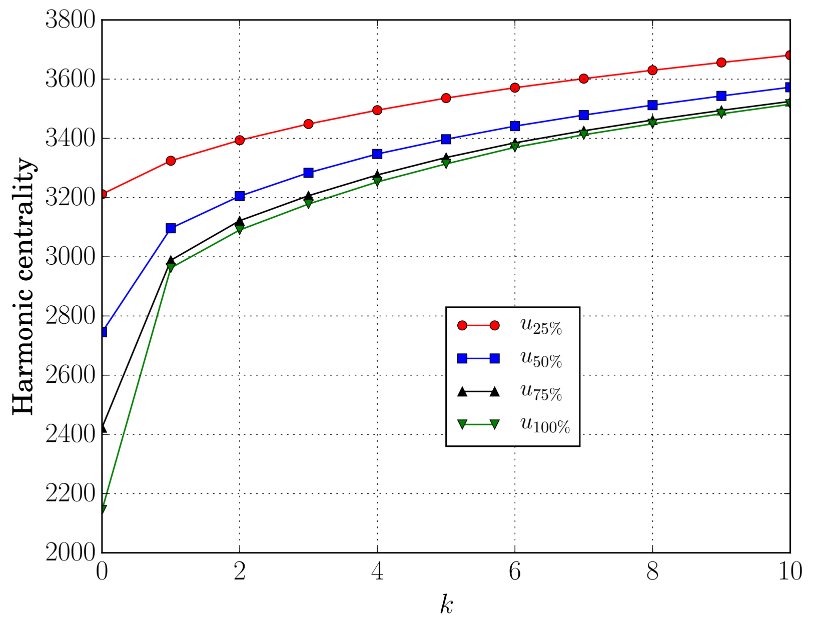


FIGURE 3.12: Performance of DYNAMICGREEDYIMPROVEMENT algorithm on network IMDB.



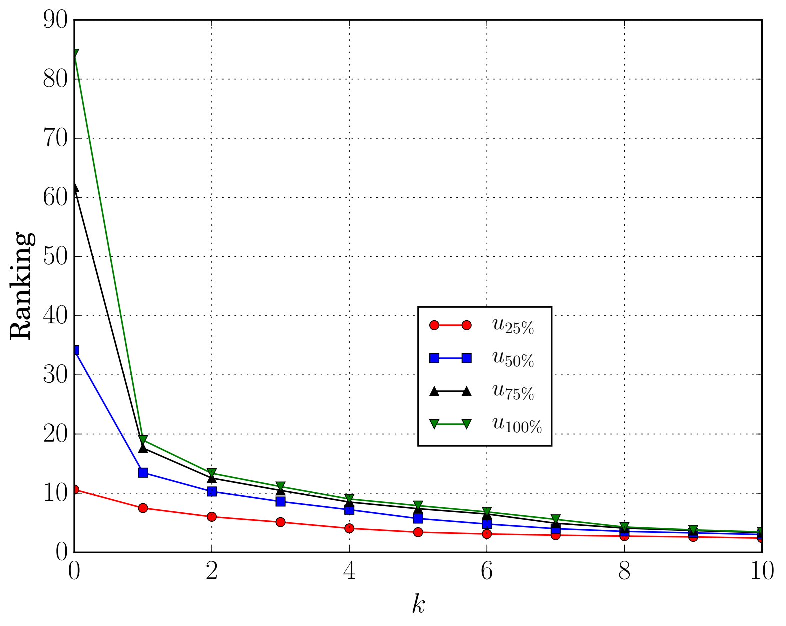


FIGURE 3.13: Harmonic centrality and ranking of vertex u as a function of k in network ca-HepPh.

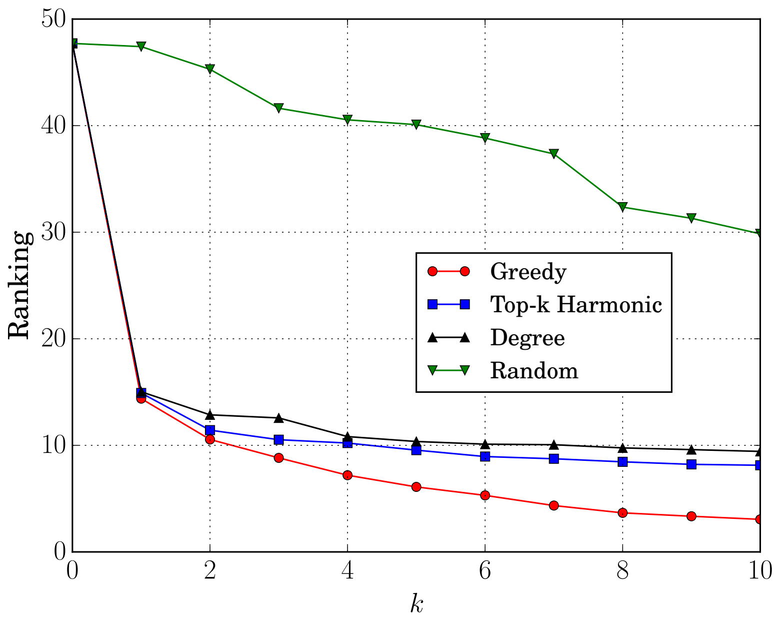


FIGURE 3.14: Percentage ranking position between the Ranking of the solution obtained by the GREEDYIMPROVEMENT algorithm and the different baselines in network ca-HepPh.

Chapter 4

Centrality Maximization problem for betweenness centrality

In this chapter, we study the CM problem on betweenness centrality (CM-B). We first theoretically analyse different algorithms and then we experimentally evaluate their performance. Unlike the CM-H case, we have an approximation algorithm with exact guarantee only for directed graphs. However, also in the undirected graphs case, the algorithm that select edges in a greedy way has experimentally the best performance in terms of value and ranking compared to previous proposals. Most of the results presented in this chapter are included in [12, 30].

4.1 Problem definition

Given a directed (respectively undirected) graph $G = (V, E)$, a vertex $u \in V$, and an integer k, the *Maximum Betweenness Improvement* (in short, CM-B) problem consists in finding a set S of ingoing edges (respectively incident edges) to u not in E (that is, $S \subseteq \{(v, u) : v \in V \setminus N_u\}$) such that $|S| \leq k$ and $b_u(S)$ is maximum.

4.2 Hardness results

4.2.1 Hardness of approximation for directed graphs

In this section we first show that it is NP-hard to approximate problem CM-B within a factor greater than $1 - \frac{1}{2e}$. Then, we focus on the *Maximum Betweenness Ranking*

Improvement problem (CRM-B) and show that it cannot be approximated within any multiplicative constant bound, unless $P = NP$.

Theorem 4.1. *For each $\gamma > 1 - \frac{1}{2e}$, there is no γ-approximation algorithm for the* CM-B *problem in directed graphs, unless $P = NP$.*

Proof. The proof is similar to the proof of Theorem 3.1. We give an L-reduction with parameters a and β [97, Chapter 16] to the *maximum set coverage problem* (MSC) defined in Section 3.2.1. In detail, we will give a polynomial-time algorithm that transforms any instance I_{MSC} of MSC into an instance $I_{\text{CM-B}}$ of CM-B and a polynomial-time algorithm that transforms any solution $S_{\text{CM-B}}$ for $I_{\text{CM-B}}$ into a solution S_{MSC} for I_{MSC} such that the following two conditions are satisfied for some values a and β:

$$OPT(I_{\text{CM-B}}) \leq aOPT(I_{\text{MSC}}) \tag{4.1}$$

$$OPT(I_{\text{MSC}}) - s(S_{\text{MSC}}) \leq \beta\left(OPT(I_{\text{CM-B}}) - b_v(S_{\text{CM-B}})\right), \tag{4.2}$$

where OPT denotes the optimal value of an instance of an optimization problem. If the above conditions are satisfied and there exists an α-approximation algorithm $A_{\text{CM-B}}$ for CM-B, then there exists a $(1 - a\beta(1 - \alpha))$-approximation algorithm A_{MSC} for MSC [97, Chapter 16]. Since it is NP-hard to approximate MSC within a factor greater than $1 - \frac{1}{e}$ [39], then the approximation factor of A_{MSC} must be smaller than $1 - \frac{1}{e}$, unless $P = NP$. This implies that $1 - a\beta(1 - \alpha) < 1 - \frac{1}{e}$ that is, the approximation factor α of $A_{\text{CM-B}}$ must satisfy $\alpha < 1 - \frac{1}{a\beta e}$, unless $P = NP$. In the following, we give an L-reduction and determine the constant parameters a and β. In the reduction, each element x_i and each set S_j in an instance of MSC corresponds to a vertex in an instance of CM-B, denoted by v_{x_i} and v_{S_j}, respectively. There is an arc from v_{x_i} to v_{S_j} if and only if $x_i \in S_j$. The CM-B instance contains two further nodes v and t and an arc (v, t). A solution to such an instance consists of arcs from nodes v_{S_j} to v and the aim is to cover with such arcs the maximum number of shortest paths from nodes v_{x_i} to t. We will prove that we can transform a solution to CM-B into a solution to MSC such that any node v_{x_i} that has a shortest path passing trough v corresponds to a covered element $x_i \in X$. We give more detail in what follows.

Given an instance $I_{\text{MSC}} = (X, \mathcal{F}, k')$ of MSC, where $\mathcal{F} = \{S_1, S_2, \ldots S_{|\mathcal{F}|}\}$, we define an instance $I_{\text{CM-B}} = (G, v, k)$ of CM-B, where:

- $G = (V, E)$;
- $V = \{v, t\} \cup \{v_{x_i} \mid x_i \in X\} \cup \{v_{S_j} \mid S_j \in \mathcal{F}\}$;
- $E = \{(v, t)\} \cup \{(v_{x_i}, v_{S_j}) \mid x_i \in S_j\}$;

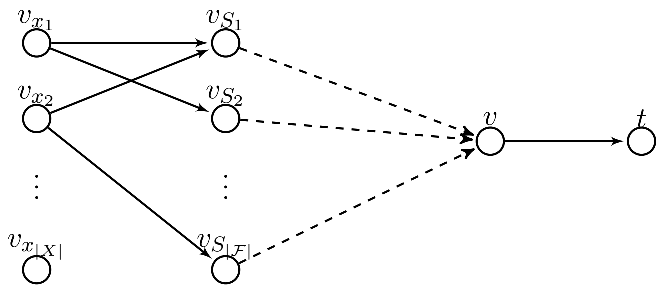


FIGURE 4.1: Reduction used in Theorem 4.1. In the example $x_1 \in S_1$, $x_1 \in S_2$, $x_2 \in S_1$, and $x_2 \in S_{\mathcal{F}}$. The dashed arcs denote those added in a solution.

- $k = k'$.

See Figure 4.1 for a visualization.

Without loss of generality, we can assume that any solution $S_{\text{CM-B}}$ to CM-B contains only arcs (v_{S_j}, v) for some $S_j \in \mathcal{F}$. In fact, if a solution does not satisfy this property, then we can improve it in polynomial time by repeatedly applying the following transformation: for each arc $e = (v_{x_i}, v)$ in $S_{\text{CM-B}}$ such that $x_i \in X$, exchange e with an arc (v_{S_j}, v) such that $x_i \in S_j$ and $(v_{S_j}, v) \notin S_{\text{CM-B}}$ if it exists or remove e otherwise. Note that if no arc (v_{S_j}, v) such that $x_i \in S_j$ and $(v_{S_j}, v) \notin S_{\text{CM-B}}$ exists, then all the shortest paths from x_i to t pass through v and therefore the arc (v_{x_i}, v) can be removed without changing the value of $b_v(S_{\text{CM-B}})$. Such a transformation does not decrease the value of $b_v(S_{\text{CM-B}})$ in fact, all the shortest paths passing through v in the original solution still pass through v in the obtained solution. Moreover, if Condition (4.2) is satisfied for the obtained solution, then it is satisfied also for the original solution. In such a solution, all the paths (if any) from v_{x_i} to t, for each $x_i \in X$, and from v_{S_j} to t, for each $S_j \in \mathcal{F}$ pass through v and therefore the ratio $\frac{\sigma_{stv}(S_{\text{CM-B}})}{\sigma_{st}(S_{\text{CM-B}})}$ is 1, for each $s \in V \setminus \{v, t\}$ such that $\sigma_{st}(S_{\text{CM-B}}) \neq 0$. We can further assume, again without loss of generality, that any solution $S_{\text{CM-B}}$ is such that $|S_{\text{CM-B}}| = k$, in fact, if $|S_{\text{CM-B}}| < k$, then we can add to $S_{\text{CM-B}}$ an arc (v_{S_j}, v) that is not yet in $S_{\text{CM-B}}$. Note that such an arc must exists otherwise $k > |\mathcal{F}|$ and this operation does not decrease the value of $b_v(S_{\text{CM-B}})$.

Given a solution $S_{\text{CM-B}} = \{(v_{S_j}, v) \mid S_j \in \mathcal{F}\}$ to CM-B, we construct the solution $S_{\text{MSC}} = \{S_j \mid (v_{S_j}, v) \in S_{\text{CM-B}}\}$ to MSC. By construction, $|S_{\text{MSC}}| = |S_{\text{CM-B}}| = k = k'$. Moreover, the set of elements x_i of X such that $\sigma_{v_{x_i}t}(S_{\text{CM-B}}) \neq 0$ is equal to $\{x_i \in S_j \mid (v_{S_j}, v) \in$

$S_{\text{CM-B}}\} = \bigcup_{S_j \in S_{\text{MSC}}} S_j$. Therefore, the betweenness centrality of v in $G(S_{\text{CM-B}})$ is:

$$\begin{aligned}
b_v(S_{\text{CM-B}}) = & \sum_{\substack{s \in V\setminus\{v,t\} \\ \sigma_{st}(S_{\text{CM-B}})\neq 0}} \frac{\sigma_{stv}(S_{\text{CM-B}})}{\sigma_{st}(S_{\text{CM-B}})} \\
= & \sum_{\substack{x_i \in X \\ \sigma_{v_{x_i}t}(S_{\text{CM-B}})\neq 0}} \frac{\sigma_{v_{x_i}tv}(S_{\text{CM-B}})}{\sigma_{v_{x_i}t}(S_{\text{CM-B}})} + \sum_{\substack{S_j \in \mathcal{F} \\ \sigma_{v_{S_j}t}(S_{\text{CM-B}})\neq 0}} \frac{\sigma_{v_{S_j}tv}(S_{\text{CM-B}})}{\sigma_{v_{S_j}t}(S_{\text{CM-B}})} \\
= & |\{x_i \in S_j \mid (v_{S_j}, v) \in S_{\text{CM-B}}\}| + |\{S_j \mid (v_{S_j}, v) \in S_{\text{CM-B}}\}| \\
= & \left| \bigcup_{S_j \in S_{\text{MSC}}} S_j \right| + |S_{\text{MSC}}| \\
= & s(S_{\text{MSC}}) + k.
\end{aligned}$$

It follows that Conditions (4.1) and (4.2) are satisfied for $a = 2$, $\beta = 1$ since: $OPT(I_{\text{CM-B}}) = OPT(I_{\text{MSC}}) + k \leq 2OPT(I_{\text{MSC}})$ and $OPT(I_{\text{MSC}}) - s(S_{\text{MSC}}) = OPT(I_{\text{CM-B}}) - b_v(S_{\text{CM-B}})$, where the first inequality is due to the fact that $OPT(I_{\text{MSC}}) \geq k$. Notice that if $OPT(I_{\text{MSC}}) < k$, then the greedy algorithm finds an optimal solution for MSC. The statement follows by plugging the values of a and b into $\alpha < 1 - \frac{1}{a\beta e}$. $\square$

4.2.2 Hardness of approximation for undirected graphs

In this section we study the CM-B problem on undirected graphs Set Coverage (MSC) problem defined as follows. Given a ground set X, a family of subsets of X, $\mathcal{F} = \{S_1, S_2, \dots S_{|\mathcal{F}|}\}$, and an integer k', find a family $\mathcal{F}' \subseteq \mathcal{F}$ such that $|\mathcal{F}'| \leq k'$ that maximize $s(\mathcal{F}') = |\cup_{S_i \in \mathcal{F}'} S_i|$.

Theorem 4.2. *For each $\gamma > 1 - \frac{1}{2e}$, there is no γ-approximation algorithm for the* CM-B *problem in undirected graphs, unless $P = NP$.*

The proof is similar to that of the directed graph case.

4.2.3 Improving the position in the ranking

We study the problem of improving the position of a node v in the ranking obtained by sorting all the nodes in non-increasing order according to their harmonic centrality. The *ranking* of a node v according to betweenness centrality is the placement of v in the ordering induced by b and it is defined as $r_v^b = |\{u \in V \mid b_u > b_v\}| + 1$. Given a directed graph $G = (V, E)$, a vertex $u \in V$, and an integer k, the *Maximum Betweenness Ranking Improvement* (in short, CRM-B) problem consists in finding a set S of ingoing edges to

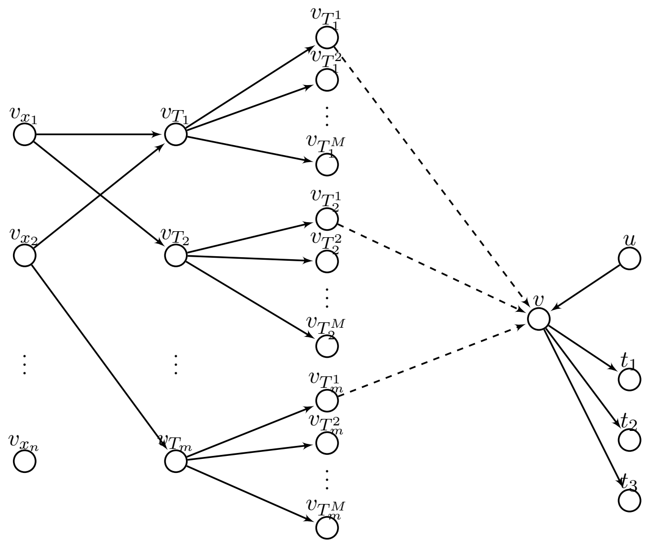


FIGURE 4.2: The reduction used in Theorem 4.3. The dashed arcs denote those added in a solution to CRM-B.

u not in E, $S \subseteq \{(u,v) : v \in V \setminus N_u^i\}$ such that $|S| \leq k$ and $\rho_v^b(S)$ is maximum, where $\rho_v^b(S) = r_v^b - r_v^b(S)$. We show that, unless $P = NP$, we cannot find a polynomial time approximation algorithm for CRM-B with a constant approximation guarantee.

Theorem 4.3. *For any constant $\alpha \leq 1$, there is no α-approximation algorithm for the* CRM-B *problem in directed graphs, unless $P = NP$.*

Proof. The proof is similar to the proof of Theorem 3.3. By contradiction, let us assume that there exists a polynomial time algorithm A that guarantees an approximation factor of α. As in the harmonic case, we show that we can use A to determine whether an instance I of the *exact cover by 3-sets* problem (X3C) admits a feasible solution or not.

Note that we can assume without loss of generality that $m > q$.

Given an instance $I = (X, C)$ of X3C where $|X| = n = 3q$ and $|C| = m$, we define an instance $I' = (G, v, k)$ of CRM-B as follows.

- $G = (V, E)$;
- $V = \{v, u, t_1, t_2, t_3\} \cup \{v_{x_i} \mid x_i \in X\} \cup \{v_{T_j} \mid T_j \in C\} \cup \{v_{T_j^\ell} \mid T_j \in C,\ \ell = 1, 2, \ldots, M\}$;
- $E = \{(v_{x_i}, v_{T_j}) \mid x_i \in T_j\} \cup \{(v_{T_j}, v_{T_j^\ell}) \mid T_j \in C,\ \ell = 1, 2, \ldots, M\} \cup \{(u, v), (v, t_1), (v, t_2), (v, t_3)\}$;
- $k = q$.

where $M = 5q + 1$. See Figure 4.2 for a visualization.

The proof proceeds by showing that I admits an exact cover if and only if I' admits a solution S such that $\rho_v^b(S) > 0$. This implies that, if OPT is an optimal solution for I', then $\rho_v^b(OPT) > 0$ if and only if I admits an exact cover. Hence, the statement follows by observing that algorithm A outputs a solution S such that $\rho_v^b(S) > \alpha\rho_v^b(OPT)$ and hence $\rho_v^b(S) > 0$ if and only if I admits an exact cover.

In I', $b_v = 3$, $b_{v_{T_j}} = 3M = 15q + 3$, for each $T_j \in C$, and $b_w = 0$, for any other node w. Therefore, $r_{T_j} = 1$, for each $T_j \in C$, $r_v = m + 1$, and $r_w = m + 2$, for any other node w. In the proof we will use the observation that, in instance I', adding arcs incident to v does not decrease the betweenness value of any node, that is for any node $w \in V$ and for any solution S to I', $b_w(S) \geq b_w$.

If instance I of X3C admits an exact cover C', then consider the solution $S = \{(v_{T_j^1}, v) \mid T_j \in C'\}$ to I'. Note that $|S| = q = k$ and therefore we only need to show that $\rho_v^b(S) > 0$. Indeed, in the following we show that $\rho_v^b(S) = m - q > 0$. Since C' is an exact cover, then all nodes v_{x_i} are connected to the 3 nodes t_i and all the paths connecting them pass through v. The same holds for nodes v_{T_j} and $v_{T_j^1}$ such that $T_j \in C'$. Since there are $3q$ nodes v_{x_i}, q nodes v_{T_j} such that $T_j \in C'$, and q nodes $v_{T_j^1}$ such that $T_j \in C'$, then the betweenness centrality of v increases to $b_v(S) = 3(5q + 1) = 15q + 3$. Nodes v_{T_j} and $v_{T_j^1}$ such that $T_j \in C'$ increase their centrality to $b_{v_{T_j}}(S) = 3(M + 4) = 15q + 15$ and $b_{v_{T_j^1}}(S) = 16$, respectively. Any other node does not change its betweenness centrality. Therefore the only nodes that have a betweenness higher than v are the q nodes $v_{T_j^1}$ such that $T_j \in C'$. It follows that $r_v^b(S) = q + 1$ and $\rho_v^b(S) = m + 1 - (q + 1) = m - q > 0$.

Let us now assume that I' admits solution S such that $|S| \leq k$ and $\rho_v^b(S) > 0$. We first prove that S is only made of arcs in the form $(v_{T_j^1}, v)$ and that $b_v(S) \geq 15q + 3$ or that it can be transformed in polynomial time into a solution with such a form without increasing its size. Assume that S has arcs not in this form, then we can apply one of the following transformations to each of such arcs $e = (w, v)$.

- If $w = v_{x_i}$ for some $x_i \in X$ and there exists a node $v_{T_j^1}$ such that $x_i \in T_j$ and $(v_{T_j^1}, v) \notin S$, then remove e and add arc $(v_{T_j^1}, v)$ to S;
- If $w = v_{x_i}$ for some $x_i \in X$ and $(v_{T_j^1}, v) \in S$ for all T_j such that $x_i \in T_j$, then remove e;
- If $w = v_{T_j}$ for some $T_j \in C$ and $(v_{T_j^1}, v) \notin S$, then remove e and add arc $(v_{T_j^1}, v)$ to S;
- If $w = v_{T_j}$ for some $T_j \in C$ and $(v_{T_j^1}, v) \in S$, then remove e;
- If $w = v_{T_j^i}$ for some $T_j \in C$ and $i > 1$, and $(v_{T_j^1}, v) \notin S$, then remove e and add arc $(v_{T_j^1}, v)$ to S;

- If $w = v_{T_j^i}$ for some $T_j \in C$ and $i > 1$, and $(v_{T_j^1}, v) \in S$, then remove e and add arc $(v_{T_{j'}^1}, v)$ to S for some j' such that $(v_{T_{j'}^1}, v) \notin S$;[1]
- If $w = t_i$ for $i \in \{1, 2, 3\}$, then remove e and add arc $(v_{T_{j'}^1}, v)$ to S for some j' such that $(v_{T_{j'}^1}, v) \notin S$.

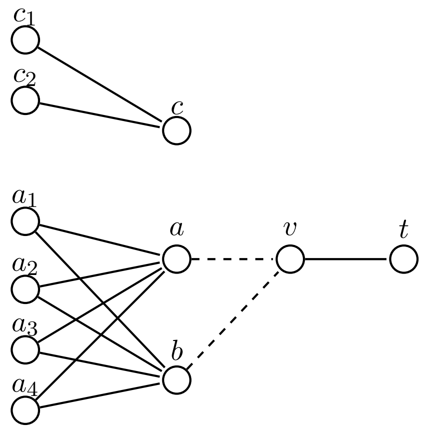
Let us denote by S' and S the original solution and the solution that is eventually obtained by applying the above transformations, respectively. All the above transformations remove an arc and possibly add another arc, therefore the size of the transformed solution is at most the original size, that is $|S| \leq |S'| \leq k$. It remains to show that $\rho_v^b(S') > 0$ implies $b_v(S) \geq 15q + 3$. Indeed, observe that v is initially in position $m + 1$ and the only nodes that have a betweenness value higher than v are the m nodes v_{T_j}. Therefore, since $\rho_v^b(S') > 0$, there is at least a node v_{T_j} such that $b_v(S') \geq b_{v_{T_j}}(S')$. Moreover, all the transformations do not decrease the value of b_v and then $b_v(S) \geq b_v(S')$ and, considering that $b_{v_{T_j}}(S') \geq b_{v_{T_j}} = 15q + 3$, we obtain $b_v(S) \geq 15q + 3$.

We now prove that the solution $C' = \{T_j \mid (v_{T_j^1}, v) \in S\}$ to I is an exact cover. By contradiction, let us assume that an element in X is not contained in any set in C' or that an element in X is contained in more than one set in C'. The latter case implies the former one since $|C'| = q$, all the sets in C' contain exactly 3 elements, and $|X| = 3q$. Hence, we assume that an element in $|X|$ is not contained in any set in C'. This implies that there exists a node $v_{x_i} \in V$ that has no path to nodes t_i and therefore the betweenness of v is at most $3(1 + 3q - 1 + 2q) = 15q$, which is a contradiction to $b_v(S) \geq 15q + 3$. $\square$

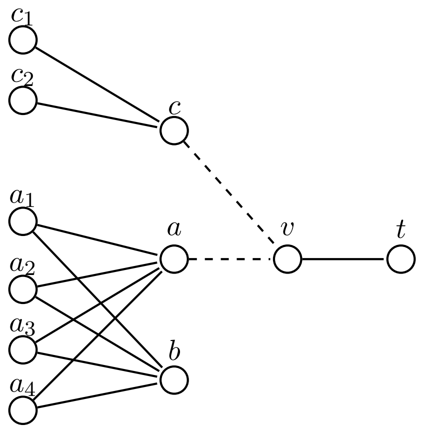
4.3 Naive algorithms

We now study and analyse the algorithms to solve CM-B problem [12, 27, 30].

- RANDOM algorithm: connect u to a set of k nodes extracted uniformly at random.
- DEGREE algorithm: connect u to a set of k nodes having the highest degree.
- TOP-K algorithm: connect u to a set of k nodes having the highest betweenness centrality.



(a) Degree and Top-k algorithms



(b) Optimal solution

Figure 4.3: Counterexample for the Degree and Top-k algorithms for $x = 4$. The dashed edges are those in a solution to cm-b. The Degree and Top-k algorithms (left) adds edges $\{a, v\}$ and $\{b, v\}$ and the value $b_v(\{a, v\}, \{b, v\})$ is 6.20. An optimal solution (right), has value $b_v(\{\{a, v\}, \{c, v\}\}) = 27$.

4.3.1 Worst case approximation ratio of the naive algorithms

We now show that the naive algorithms have an arbitrary small approximation ratio. We provide counterexamples for Degree and Top-k algorithms. Such counterexamples are valid also for the Random algorithm because it extracts a set of k nodes uniformly at random that can be equal to the sets returned by Degree or Top-k algorithms with probability $p > 0$. Consider the following instance of cm-b on undirected graphs, see Figure 4.3 for an example.

- graph $G = (V, E)$.
- $V = \{v, t, a, b, c\} \cup A \cup C$, where $A = \{a_i\}_{i=1}^{x}$, $C = \{c_i\}_{i=1}^{y}$, and $y = x - 2$, for some $x > 2$;
- $E = \{\{v, t\}\} \cup \{\{a_i, a\} \mid a_i \in A\} \cup \{\{a_i, b\} \mid a_i \in A\} \cup \{\{c_i, c\} \mid c_i \in C\}$;
- All the edges have weight 1;
- $k = 1$.

The initial values of degree are $deg_v = 1$, $deg_a = deg_b = x$, $deg_c = y$, $b_{a_i} = 2$, for each $i = 1, 2, \ldots, x$ The initial values of betweenness are $b_v = 0$, $b_a = b_b = \frac{x(x-1)}{4}$, $b_c = \frac{y(y-1)}{2}$, $b_{a_i} = \frac{1}{x}$, for each $i = 1, 2, \ldots, x$. Therefore, for $x > 2$, the two nodes with the highest betweenness and the highest degree are a and b and Degree and Top-k algorithms add edges $\{a, v\}$ and $\{b, v\}$. The solution obtained has a value $b_v(\{a, v\}) = x + 2 + \frac{1}{x+1}$,

[1]Note that such j' must exists, otherwise $m < q$.

Algorithm 9: GreedyImprovement algorithm.

```
Input  : A directed graph $G = (V, E)$; a vertex $v \in V$; and an integer $k \in \mathbb{N}$
Output: Set of edges $S \subseteq \{(u, v) \mid u \in V \setminus N_v\}$ such that $|S| \leq k$
1 $S \leftarrow \emptyset$;
2 for $i = 1, 2, \ldots, k$ do
3     foreach $u \in V \setminus (N_v(S))$ do
4         Compute $b_v(S \cup \{(u, v)\})$
5     $u_{\max} \leftarrow \arg\max\{b_v(S \cup \{(u, v)\}) \mid u \in V \setminus (N_v(S))\}$;
6     $S \leftarrow S \cup \{(u_{\max}, v)\}$;
7 return $S$;
```

since there are $x + 2$ paths between nodes in $A \cup \{a, b\}$ and t passing through v and 1 path over $x + 1$ paths from a to b passing through v. Adding edges $\{a, v\}$ and $\{c, v\}$, instead increases b_v by $y \cdot x + y + 1 + x + 1 + y + x + 1 + x = x^2 + 3x - 1$, since all the paths from $A \cup \{a, b\}$ to $C \cup \{c\}$ pass through v. Therefore, the approximation ratios of the Degree and Top-k algorithms tend to be arbitrarily small as x increases. Let us consider the directed case: we modify the Figure 4.3 adding two opposite directed edges for each undirected arc and adding only incident edges to node v. It is easy to see that also in this case we can obtained an unbounded approximation ratio.

4.4 Greedy approximation algorithm for CM-B

In this section we propose an algorithm that guarantees a constant approximation ratio for the CM-B problem. We show that the objective function b_v is monotone and submodular with respect to the possible set of arcs incident to v. Hence, we define a greedy algorithm, reported in Algorithm 9, that provides a $\left(1 - \frac{1}{e}\right)$-approximation. Algorithm 6 iterates k times and, at each iteration, it adds to a solution S an arc (u, v) that, when added to $G(S)$, gives the largest marginal increase in the betweenness of v, that is, $b_v(S \cup \{(u, v)\})$ is maximum among all the possible arcs not in $E \cup S$ incident to v. The next theorem shows that the objective function is monotone and submodular.

Theorem 4.4. *For each node v, function b_v is monotone and submodular with respect to any feasible solution for* CM-B *in directed graphs.*

Proof. We prove that each term of the sum in the formula of b_v is monotone increasing and submodular. For each pair $s, t \in V$ such that $s \neq t$ and $s, t \neq v$, we denote such term by $b_{stv}(X) = \frac{\sigma_{stv}(X)}{\sigma_{st}(X)}$, for each solution X to CM-B.

We first give two observations that will be used in the proof. Let X, Y be two solutions to CM-B such that $X \subseteq Y$.

- Any shortest path from s to t in $G(X)$ exists also in $G(Y)$. It follows that $d_{st}(Y) \leq d_{st}(X)$.
- If $d_{st}(Y) < d_{st}(X)$, then any shortest path from s to t in $G(Y)$ pass through arcs in $Y \setminus X$. Therefore, all such paths pass through v. It follows that if $d_{st}(Y) < d_{st}(X)$, then $b_{stv}(Y) = 1$.

We now show that b_v is monotone increasing, that is for each solution S to CM-B and for each node u such that $(u, v) \notin S \cup E$,

$$b_{stv}(S \cup \{(u, v)\}) \geq b_{stv}(S).$$

If $d_{st}(S) > d_{st}(S \cup \{(u, v)\})$, then $b_{stv}(S \cup \{(u, v)\}) = 1$ and since by definition $b_{stv}(S) \leq 1$, then the statement holds. If $d_{st}(S) = d_{st}(S \cup \{(u, v)\})$, then either (u, v) does not belong to any shortest path from s to t and then $b_{stv}(S \cup \{(u, v)\}) = b_{stv}(S)$, or (u, v) belongs to a newly added shortest path from s to t with the same weight and $b_{stv}(S \cup \{(u, v)\}) = \frac{\sigma_{stv}(S)+\delta}{\sigma_{st}(S)+\delta} > \frac{\sigma_{stv}(S)}{\sigma_{st}(S)} = b_{stv}(S)$, where $\delta \geq 1$ is the number of shortest paths from s to t that pass through arc (u, v) in $G(S \cup \{(u, v)\})$. In any case the statement holds.

We now show that b_{stv} is submodular, that is for each pair of solutions to CM-B S, T such that $S \subseteq T$ and for each node u such that $(u, v) \notin T \cup E$,

$$b_{stv}(S \cup \{(u, v)\}) - b_{stv}(S) \geq b_{stv}(T \cup \{(u, v)\}) - b_{stv}(T).$$

We analyze the following cases:

- $d_{st}(S) > d_{st}(T)$. In this case, $b_{stv}(T \cup \{(u, v)\}) - b_{stv}(T) = 0$ since in any case $b_{stv}(T \cup \{(u, v)\}) = b_{stv}(T) = 1$. As b_{stv} is monotone increasing, then $b_{stv}(S \cup \{(u, v)\}) - b_{stv}(S) \geq 0$.
- $d_{st}(S) = d_{st}(T)$.
  - $d_{st}(S) > d_{st}(S \cup \{(u, v)\})$. In this case, $d_{st}(T) > d_{st}(T \cup \{(u, v)\})$ and $b_{stv}(T \cup \{(u, v)\}) = b_{stv}(S \cup \{(u, v)\}) = 1$. Moreover $b_{stv}(T) \geq b_{stv}(S)$. Therefore $b_{stv}(S \cup \{(u, v)\}) - b_{stv}(S) \geq b_{stv}(T \cup \{(u, v)\}) - b_{stv}(T)$.
  - $d_{st}(S) = d_{st}(S \cup \{(u, v)\})$. In this case $d_{st}(T) = d_{st}(T \cup \{(u, v)\})$. Let us denote $b_{stv}(S) = \frac{\alpha}{\beta}$, then we have that $b_{stv}(T) = \frac{\alpha+\gamma}{\beta+\gamma}$, $b_{stv}(S \cup \{(u, v)\}) = \frac{\alpha+\delta}{\beta+\delta}$, and $b_{stv}(T \cup \{(u, v)\}) = \frac{\alpha+\gamma+\delta}{\beta+\gamma+\delta}$, where γ and δ are the number of shortest paths between s and t in $G(T)$ that pass through arcs in $T \setminus S$ and arc (u, v), respectively. The statement follows since $\frac{\alpha+\delta}{\beta+\delta} - \frac{\alpha}{\beta} \geq \frac{\alpha+\gamma+\delta}{\beta+\gamma+\delta} - \frac{\alpha+\gamma}{\beta+\gamma}$ for any $\alpha \leq \beta$, i.e. $\sigma_{stv}(S) \leq \sigma_{st}(S)$.

□

Corollary 4.5. *Algorithm 9 provides a $\left(1 - \frac{1}{e}\right)$-approximation for the* CM-B *problem in directed graphs.*

It is easy to compute the computational complexity of Algorithm GreedyImprovement. Line 2 iterates over all the numbers from 1 to k. Then, in Line 3, all the nodes u that are not yet neighbors of v are scanned. The number of these nodes is clearly $O(n)$. Finally, in Line 4, for each node u in Line 3, we add the edge $\{u, v\}$ to the graph and compute the betweenness in the new graph. Since computing betweenness requires $O(nm)$ operations in unweighted graphs, the total running time of GreedyImprovement is $O(kn^2m)$.

4.4.1 Improving the greedy algorithm running time

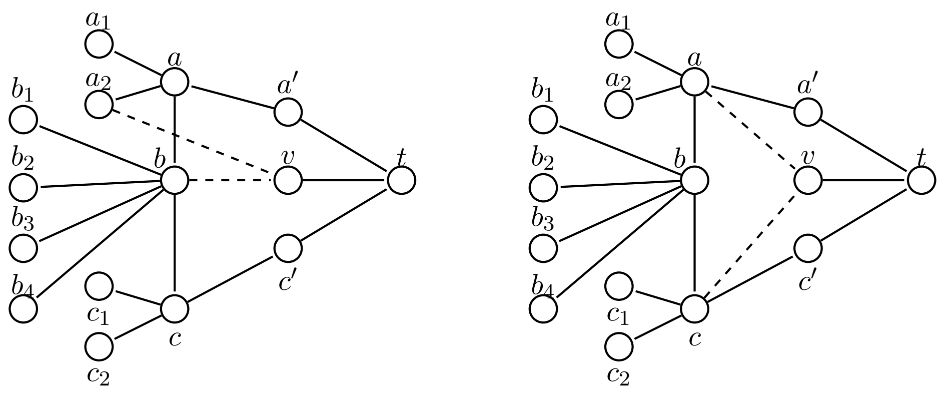
To speed up the computation of GreedyImprovement algorithm, we use the dynamic algorithm described in [12]. This algorithm is based on QUINCA algorithm [86] for incremental APSP. The main idea is to keep track of information regarding the graph and just update the parts that have changed as a consequence of the edge insertion. The dynamic algorithm updates the betweenness centrality of the nodes in linear time with respect to the number of pairs of nodes u, v which distance d_{uv} is affected by the edge insertion.

In [12], the authors show that this algorithm outperforms the static and dynamic algorithms in the literature with respect to the recomputation of the betweenness of a single node after and edge update. We refer to this algorithms has DynamicGreedyBetweenness algorithm.

4.4.2 Worst case approximation ratio of the greedy algorithm on undirected graph

We now show that the greedy algorithm exhibits an arbitrary small approximation ratio in undirected graphs. Consider the following instance of CM-B, see Figure 4.4 for an example.

- Graph $G = (V, E)$.
- $V = \{v, t, a, b, c, a', b', c'\} \cup A \cup B \cup C$, where $A = \{a_i\}_{i=1}^{y}$, $B = \{b_i\}_{i=1}^{x}$, $C = \{c_i\}_{i=1}^{y}$, and $y = x - 2$, for some $x > 2$;



(a) Greedy algorithm (b) Optimal solution

FIGURE 4.4: Counterexample for the greedy algorithm for $x = 4$. The dashed edges are those in a solution to CM-B. The greedy algorithm (left) in the first iteration adds edge $\{b, v\}$ since it increases the most the centrality of v. After adding such edge the new value of b_v is 5. In the second iteration the algorithm adds edge $\{a_2, v\}$, then the value of b_v becomes 11. An optimal solution (right), has value $b_v(\{\{a, v\}, \{c, v\}\}) = 13$.

- $E = \{\{v,t\},\{a,b\},\{b,c\},\{a,a'\},\{b,b'\},\{c,c'\},\{a',t\},\{b',t\},\{c',t\}\} \cup \{\{a_i, a\} \mid a_i \in A\} \cup \{\{b_i, b\} \mid b_i \in B\} \cup \{\{c_i, c\} \mid c_i \in C\}$;
- All the edges have weight 1;
- $k = 2$.

The initial value of b_v is zero. The greedy algorithm first chooses edge $\{b, v\}$ and then edge $\{a_i, v\}$, for some $a_i \in A$ (or equivalently $\{c_i, v\}$, for some $c_i \in A$). The value of $b_v(\{b, v\}, \{a_i, v\})$ is $2x + 3$. In fact, the following pairs have shortest paths passing through v in $G(\{b, v\}, \{a_i, v\})$: nodes in $B \cup \{b\}$ and t ($x + 1$ shortest paths), a_i and t (1 shortest path), a_i and nodes in $B\cup\{b\}$ ($\frac{x+1}{2}$ shortest paths), a_i and nodes in $C\cup\{c\}$ ($\frac{y+1}{2}$ shortest paths), and a_i and c' (1 shortest path). An optimal solution, instead, is made of edges $\{a, v\}$ and $\{c, v\}$ where $b_v(\{\{a, v\}, \{c, v\}\}) = \frac{x^2+3x-2}{2}$, where the quadratic term comes from the fact that there are $(y + 1)^2$ paths passing through v between nodes in $A\cup\{a\}$ and nodes in $C\cup\{c\}$. Therefore, the approximation ratio of the greedy algorithm tends to be arbitrarily small as x increases. The bad approximation ratio of the greedy algorithm is due to the fact that it does not consider the shortest paths that pass through v by using both edges.

4.5 Experimental results

In this section we evaluate GREEDYIMPROVEMENT in terms of accuracy and we compare it both with the optimum and with the naive algorithms described in Section 4.3. All algorithms compared in our experiments are implemented in C++, building on the open-source NetworKit [87] framework. The experiments were done on a machine equipped with 256 GB RAM and a 2.7 GHz Intel Xeon CPU E5-2680 having 2 processors with 8 cores each. To make the comparison with previous work more meaningful, we use only one of the 16 cores. The machine runs 64 bit SUSE Linux and we compiled our code with g++-4.8.1 and OpenMP 3.1.

Since computing the optimum by examining all possible k-tuples would be too expensive even on small graphs, we use an Integer Programming (IP) formulation, described in the following Section.

4.5.1 Directed graphs

Evaluating the solution quality. We measure the approximation ratio of the greedy algorithm on three types of randomly generated directed networks, namely directed Preferential Attachment (in short, PA) [18], Copying (in short, COPY) [52], Compressible Web (in short, COMP) [23]. For each graph type, we generate 5 different instances with the same size. We focus our attention on twenty vertices v, which have been chosen on the basis of their original betweenness ranking. In particular, we divide the list of vertices, sorted by their original ranking, in four intervals, and choose five random vertices uniformly at random in each interval. In each experiment, we add $k = \{1, 2, ..., 7\}$ edges. We evaluate the quality of the solution produced by the greedy algorithm by measuring its approximation ratio and we report the results in Table 4.1. We write the following Nonlinear program to compute the optimal solution and we solve it with the Mixed-Integer Nonlinear Programming Solver Couenne [10].

Let S be a solution to an instance of CM-B. Given a node v, we define a variable x_u for each node $u \in V \setminus (N_v \cup \{v\})$

$$x_u = \begin{cases} 1 & \text{if } (u, v) \in S \\ 0 & \text{otherwise.} \end{cases}$$

We define a variable y_{st} for each $s, t \in V \setminus \{v\}$, $s \neq t$.

$$y_{st} = \begin{cases} 1 & \text{If all shortest paths from } s \text{ to } t \text{ in } G(S) \text{ pass through node } v \\ 0 & \text{otherwise.} \end{cases}$$

For each pair of nodes $s, t \in V \setminus \{v\}$, $s \neq t$, we denote by $A(s, t)$ the set of nodes u not in N_v such that all the shortest paths between s and t in $G(\{(u, v)\})$ pass through edge

(u,v) and hence through node v. Note that in this case, $d_{st} > d_{st}(\{(u,v)\})$ and hence $A(s,t)$ is defined as $A(s,t) = \{u \mid d_{st} > d_{st}(\{(u,v)\})\}$. Set $B(s,t)$ is defined as the set of nodes u not in N_v such that at least a shortest path between s and t in $G(\{(u,v)\})$ does not pass through edge (u,v) and hence $B(s,t) = V \setminus (A(s,t) \cup N_v)$. We denote by $\bar{\sigma}_{stv}(u)$ the number of shortest paths from s to t in $G(\{(u,v)\})$ passing thorough edge (u,v).

The following non linear formulation solves the CM-B problem:

$$\max \sum_{\substack{s,t\in V\\ s\neq t; s,t\neq v}} \left((1-y_{st}) \frac{\sigma_{stv} + \sum_{u\in B(s,t)} x_u \bar{\sigma}_{stv}(u)}{\sigma_{st} + \sum_{u\in B(s,t)} x_u \bar{\sigma}_{stv}(u)} + y_{st} \right) \tag{4.3}$$

$$\text{subject to} \sum_{u\in A(s,t)} x_u \geq y_{st}, \qquad s,t \in V \setminus \{v\}, s \neq t \tag{4.4}$$

$$\sum_{u\in V\setminus(N_v\cup\{v\})} x_u \leq k,$$

$$x_u, y_{st} \in \{0,1\} \qquad s \in V \setminus \{v\}, t \in V \setminus \{v,s\}$$

Let us consider a solution S to the above formulation. In the case that $y_{st} = 1$, for some pair of nodes $s,t \in V \setminus \{v\}$, $s \neq t$, then Constraint (4.4) implies that, for at least a node $u \in A(s,t)$, variable x_u must be set to 1 and hence all the shortest paths between s and t in $G(S)$ pass through v. In this case, the term corresponding to pair (s,t) in the objective function (4.3) is correctly set to be equal to 1.

If $y_{st} = 0$ and $x_u = 0$, for each $u \in A(s,t)$, then the number of shortest paths between s and t in $G(S)$ passing trough v is equal to $\sigma_{stv} + \sum_{u\in B(s,t)} x_u \bar{\sigma}_{stv}(u)$ and the overall number of shortest paths between s and t in $G(S)$ is equal to $\sigma_{st} + \sum_{u\in B(s,t)} x_u \bar{\sigma}_{stv}(u)$. In this case, the term corresponding to pair (s,t) in the objective function (4.3) is correctly set to be equal to $\frac{\sigma_{stv} + \sum_{u\in B(s,t)} x_u \bar{\sigma}_{stv}(u)}{\sigma_{st} + \sum_{u\in B(s,t)} x_u \bar{\sigma}_{stv}(u)}$.

Note that, $\frac{\sigma_{stv} + \sum_{u\in B(s,t)} x_u \bar{\sigma}_{stv}(u)}{\sigma_{st} + \sum_{u\in B(s,t)} x_u \bar{\sigma}_{stv}(u)} \leq 1$ and therefore a solution in which $y_{st} = 0$ and $x_u = 1$, for some $u \in A(s,t)$ is at least as good as a solution in which y_{st} is set to 1 instead of 0 and the other variables are unchanged. Hence, we can assume without loss of generality that the case in which $y_{st} = 0$ and $x_u = 1$, for some $u \in A(s,t)$, cannot occur in an optimal solution.

The experiments clearly show that the experimental approximation ratio is by far better than the theoretical one proven in Section 4.4, that is $1 - \frac{1}{e} \sim 0.63$. In fact, in all our tests, the experimental ratio is always greater than 0.96. Notice that is not possible to process graphs with more than about 200 nodes in reasonable time.

Table 4.1: Comparison between the GreedyImprovement algorithm and the optimum. The first three columns report the type and size of the graphs; the fourth column reports the approximation ratio.

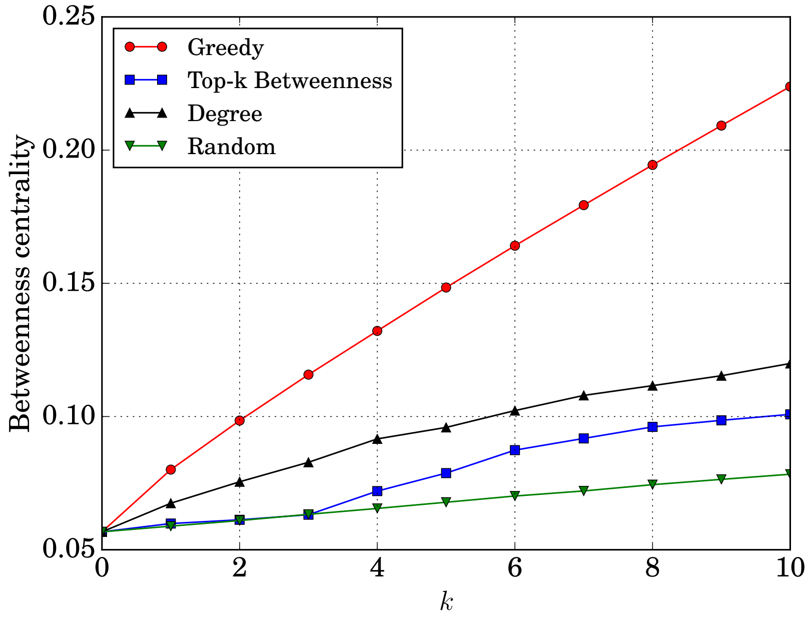
Network	$n = \|V\|$	$m = \|E\|$	Min. approx. ratio
PA	100	130	1
COPY	100	200	0.98
COMP	100	200	0.98
COMP	100	500	0.96

The comparison with naive algorithms. We also analyze the performance of GreedyImprovement on the real-world directed networks of Table 4.2. Since finding the optimum on these networks would take too long, we compare the solution of GreedyImprovement with the algorithms described in Section 4.3.

Table 4.2: Running times of the betweenness algorithms on directed real-world graphs using the incremental algorithm for the betweenness of all nodes described in [12].

Network	$n = \|V\|$	$m = \|E\|$	Time ($k = 10$) [s]
subelj-jung	6 120	50 535	9.64
wiki-Vote	7 115	100 762	8.64
elec	7 118	103 617	13.28
freeassoc	10 617	63 788	140.14
dblp-cite	12 591	49 728	348.16
subelj-cora	23 166	91 500	348.16
ego-twitter	23 370	33 101	5.79
ego-gplus	23 628	39 242	45.20
munmun-digg	30 398	85 247	732.67
linux	30 837	213 424	453.36

For each graph, we pick one node at random, compute its betweenness on the initial graph and try to increase it with the four heuristics. We refer to the selected node as *pivot*. Since the results may vary depending on the initial betweenness of the pivot, we also repeat each experiment with 10 different pivots and report the average results over the different pivots. In each experiment, we add $k = \{1, 2, ..., 10\}$ edges and compute the ranking and betweenness of the pivot after each insertion. Figure 4.5 reports the results for two directed graphs, namely munmun-digg-reply (top) and linux (bottom). We define the percentage betweenness of a node v as $b_v \cdot \frac{100}{(n-1)(n-2)}$, where b_v is the betweenness of v and $(n-1)(n-2)$ represents the maximum theoretical betweenness a node can have in a graph with n nodes. For each value of k, the plots show the average percentage betweenness of a pivot after the insertion of k edges (each point represents the average over the 10 pivots). Clearly, the pivot's betweenness after k insertions is a non-decreasing function of k, since the insertion of an edge can only increase (or leave unchanged) the betweenness



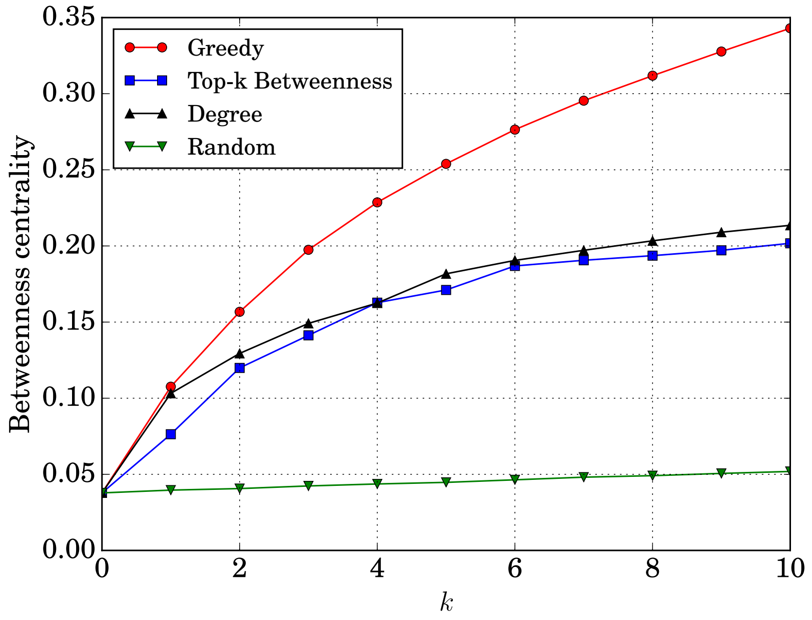
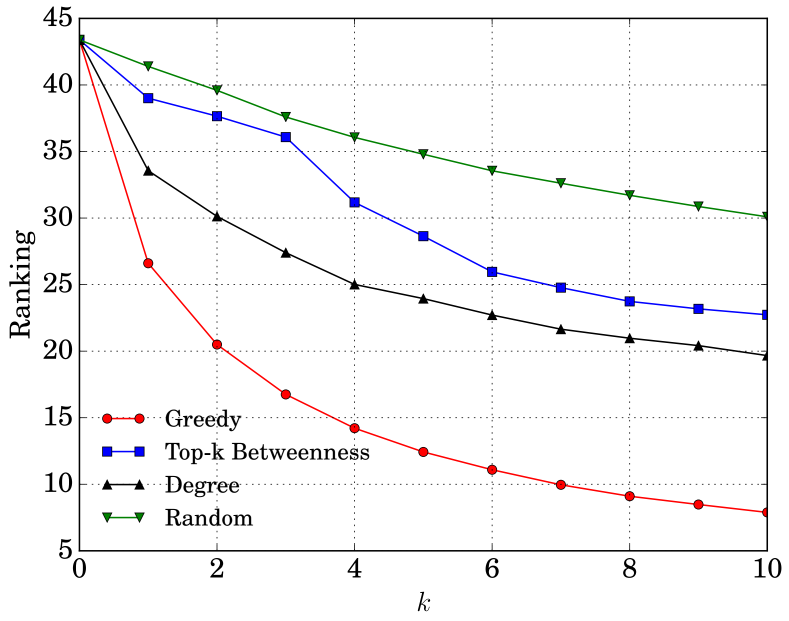


FIGURE 4.5: Percentage betweenness of the pivot as a function of the number k of inserted edges for the four heuristics. Top: results for the munmun-digg-reply graph. Bottom: results for the linux graph.

of one of its endpoints. In both plots, GREEDYIMPROVEMENT outperforms the other heuristics. For example, after k edge insertions, the average betweenness of a pivot in the munmun-digg-reply graph is 81460 with GREEDYIMPROVEMENT, 43638 with DEGREE, 36690 with TOP-K and 28513 with RANDOM. A similar behaviour can be observed for the average ranks of the pivots, reported in Figure 4.6. The figures report the percentage ranks, i.e. the ranks multiplied by $\frac{100}{n}$, since n is the maximum rank a node can have in



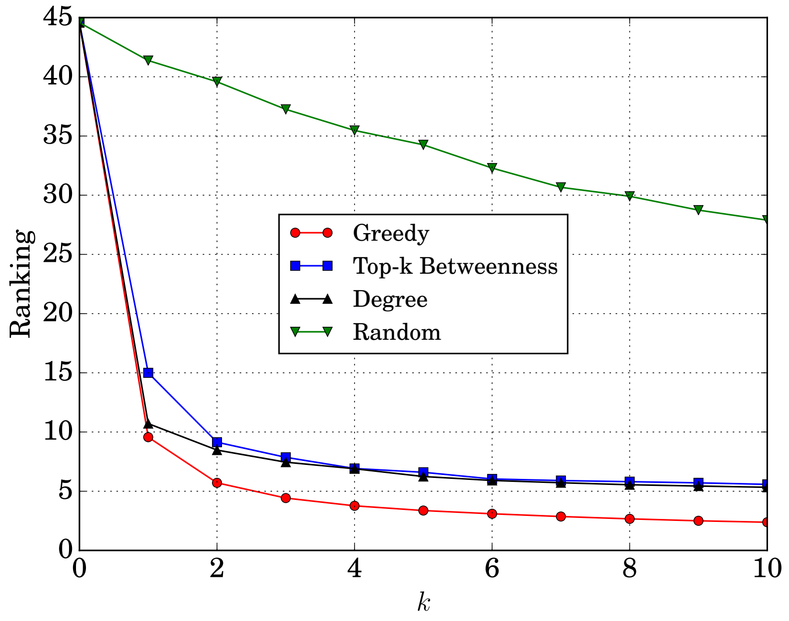
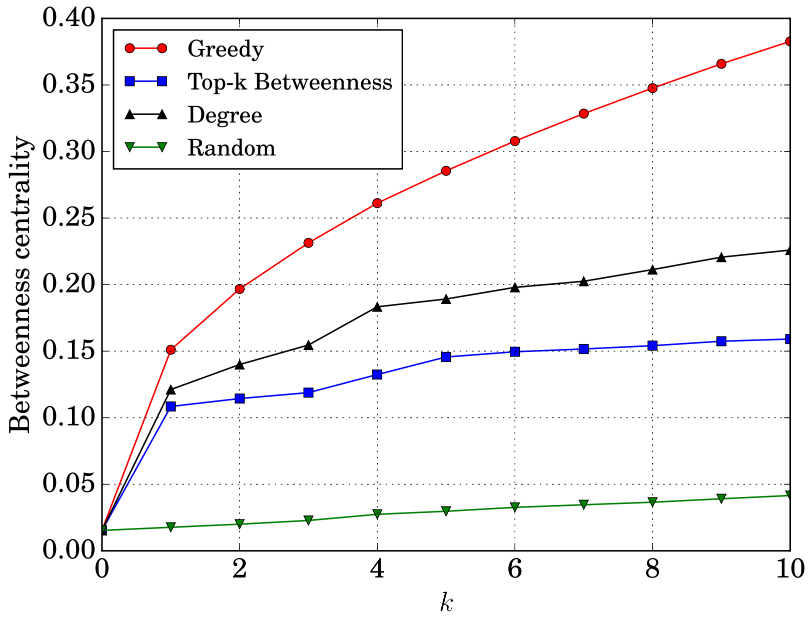


FIGURE 4.6: Percentage rank of the pivot as a function of the number k of inserted edges for the four heuristics. Top: results for the munmun-digg-reply graph. Bottom: results for the linux graph.

a graph with n nodes. This can be seen as the fraction of nodes with higher betweenness than the pivot. On munmun-digg-reply , the average initial rank is 2620 (about 43%). After 10 insertions, the rank obtained using GREEDYIMPROVEMENT is 476 (about 7%), whereas the one obtained by the other approaches is never lower than 1188 (about 19%). It is interesting to notice that 3 edge insertions with GREEDYIMPROVEMENT yield a rank of 1011, which is better than the one obtained by the other approaches after 10



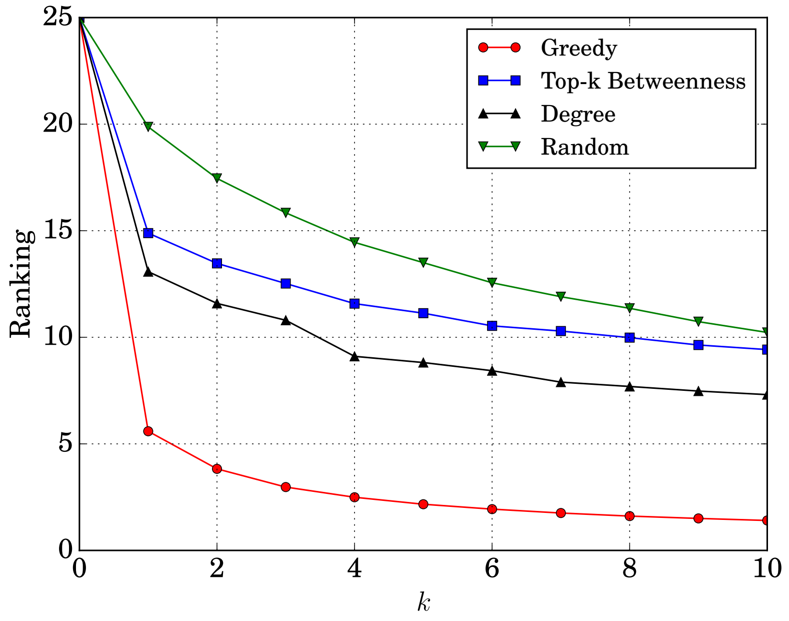


FIGURE 4.7: Average results over all directed networks. On the top, average percentage betweenness of the pivots as a function of k. On the bottom, average percentage rank of the pivots.

insertions. Similarly, also on the linux graph, 3 iterations of GREEDYIMPROVEMENT are enough to bring down the rank from 2498 (45.6%) to 247 (4.4%), whereas the other approaches cannot go below 299 (5.3%) with 10 iterations. Similar results can be observed on the other tested (directed) instances. Figure 4.7 reports the average results over all directed networks, both in terms of betweenness (top) and rank (bottom) improvement. The initial average betweenness of the sample pivots is 0.015%. GREEDYIMPROVEMENT is by far the best approach, with an average final percentage betweenness (after 10 iterations) of 0.38% and an average final percentage rank of 1.4%. As a comparison, the best alternative approach (DEGREE) yields a percentage betweenness of 0.22% and a percentage rank of 7.3%. Not surprisingly, the worst approach is RANDOM, which in 10 iterations yields a final percentage betweenness of 0.04% and an average percentage rank of 10.2%. On average, a single iteration of GREEDYIMPROVEMENT is sufficient for a percentage rank of 5.5%, better than the one obtained by all other approaches in 10 iterations. Also, it is interesting to notice that in our experiments DEGREE performs significantly better than TOP-K. This means that, for the betweenness of a node in a directed graph, it is more important to have incoming edges from nodes with high out-degree than with high betweenness. We will see in the following that our results show a different behaviour for undirected graphs.

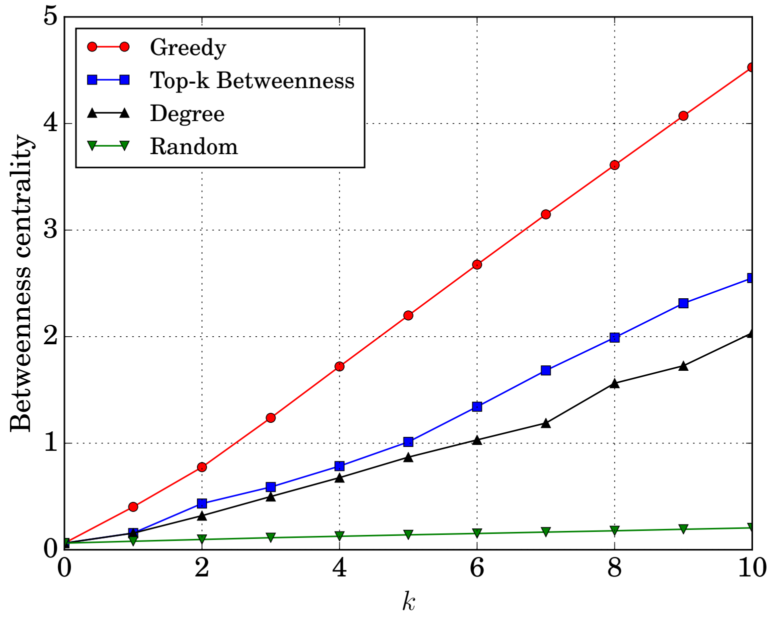
Also, notice that, although the percentage betweenness scores are quite low, the improvement using GREEDYIMPROVEMENT is still large: with 10 insertions, on average the scores change from an initial 0.015% to 0.38%, which is about 25 times the initial value.

4.5.2 Undirected graphs

Although it was proven [30] that GREEDYIMPROVEMENT has an unbounded approximation ratio for undirected graphs, it is still not clear how it actually performs in practice. Therefore, we compare GREEDYIMPROVEMENT with TOP-K BETWEENNESS, TOP-K DEGREE and RANDOM also on several undirected real-world networks, listed in Table 4.3.

Figure 4.8 shows the percentage betweenness and ranking, averaged over the undirected networks of Table 4.3. Also in this case, GREEDYIMPROVEMENT outperforms the other heuristics. In particular, the average initial betweenness of the pivots in the different graphs is 0.05%. After 10 iterations, the betweenness goes up to 3.7% with GREEDYIMPROVEMENT, 1.6% with DEGREE, 2.1% with TOP-K and only 0.17% with RANDOM. The average initial rank is 45%. GREEDYIMPROVEMENT brings it down to 0.7% with ten iterations and below 5% already with two. Using the other approaches, the average rank is always worse than 10% for TOP-K BETWEENNESS, 15% for DEGREE and 20% for RANDOM. As mentioned before, differently from directed graphs, TOP-K performs significantly better than DEGREE in undirected graphs.

Also, notice that in undirected graphs the percentage betweenness scores of the nodes in the examined graphs are significantly larger than those in the directed graphs. This could be due to the fact that many node pairs have an infinite distance in the examined directed graphs, meaning that these pairs do not contribute to the betweenness of any



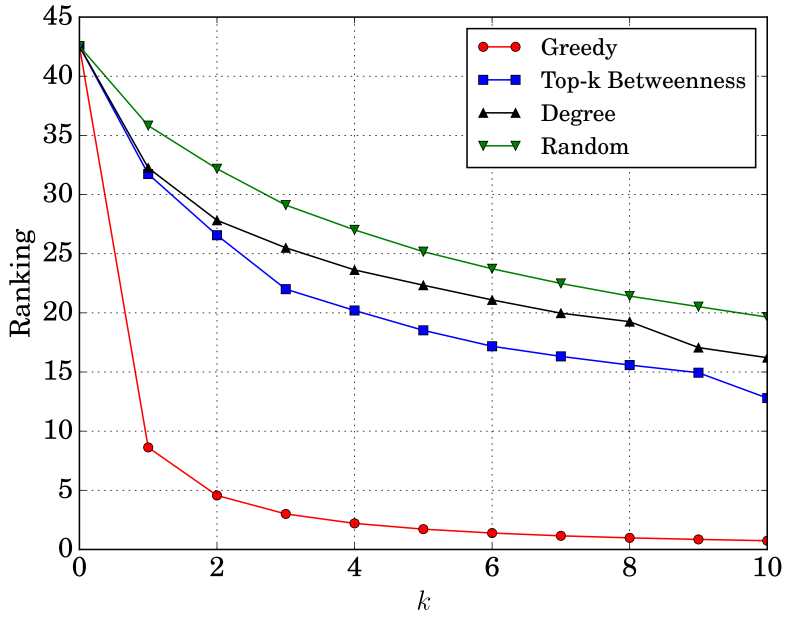


FIGURE 4.8: Average results over all undirected networks. On the top, average percentage betweenness of the pivots as a function of k. On the bottom, average percentage rank of the pivots.

node. Also, say we want to increase the betweenness of x by adding edge (v, x). The pairs (s, t) for which we can have a shortcut (leading to an increase in the betweenness of x) are limited to the ones such that s can reach v and such that t is reachable from x, which might be a small fraction of the total number of pairs. On the contrary, most undirected graphs have a giant connected component containing the greatest majority of the nodes. Therefore, it is very likely that a pivot belongs to the giant component or

Table 4.3: Running times of the betweenness algorithms on undirected real-world graphs using the incremental algorithm for the betweenness of all nodes described in [12].

Network	$n = \|V\|$	$m = \|E\|$	Time ($k = 10$) [s]
Mus-musculus	4 610	5 747	575.3
HC-BIOGRID	4 039	10 321	760.0
Caenor-eleg	4 723	9 842	557.3
ca-GrQc	5 241	14 484	310.8
advogato	7 418	42 892	190.7
hprd-pp	9 465	37 039	1097.7
ca-HepTh	9 877	25 973	2933.3
dr-melanog	10 625	40 781	1410.2
oregon1	11 174	23 409	596.7
oregon2	11 461	32 730	430.2
Homo-sapiens	13 690	61 130	2259.6
GoogleNw	15 763	148 585	691.9
CA-CondMat	21 363	91 342	28832

that it will after the first edge insertion.

It is interesting to notice that, despite the unbounded approximation ratio, the improvement achieved by GreedyImprovement on undirected graphs is even larger than for the directed ones: on average 74 times the initial score.

Chapter 5

Conclusions

One of the main issues in complex networks analysis is to identify what are the most important nodes within a network. To mathematically capture this concept, several centrality metrics are defined in literature. In this thesis we studied how to increase the centrality value of a node as much as possible. To this aim we defined the optimization problem of finding a limited amount of edges to be added in a network in order to maximize the centrality of a given node. We tackled the Centrality Maximization (CM) problem for two of the most important centrality measures namely harmonic centrality (CM-H) and betweenness centrality (CM-B). Regarding CM-H, we designed a greedy $(1 - 1/e)$-approximation algorithm for both directed and undirected graphs. We experimentally evaluated the quality of the solution and the performance on both synthetic and real networks. We also developed a dynamic algorithm to speed up the computation and analyse large networks. Instead, regarding CM-B, we proposed a greedy $(1 - 1/e)$-approximation algorithm ratio only for directed graphs. Despite we proved that the same algorithm can have an unbounded approximation ratio on undirected graphs, it exhibits experimentally good performance in terms of value and ranking. We showed that CM-H cannot be approximated in polynomial-time within a factor $1 - \frac{1}{3e}$ in directed graphs ($1 - \frac{1}{15e}$ in undirected graphs), unless $P = NP$. On the other hand, we proved that CM-B cannot be approximated in polynomial time within a factor $1 - \frac{1}{2e}$ in both directed and undirected graphs, unless $P = NP$. We also show that the problems of improving the ranking according to harmonic (CRM-H) and betweenness (CRM-B) centralities adding a limited amount of edges incident do not admit a polynomial time approximation algorithm, unless $P = NP$. Our experiments show that the solution achieved by the greedy algorithms is far better in practice than the theoretical approximation factor and they perform better in terms of quality of the solutions than several simple natural algorithms. Despite we proved that the CRM problem for harmonic and betweenness centralities does not admit a polynomial time approximation algorithm, we notice that the greedy algorithms allow to reach the top positions in the ranking with few edges addition. Moreover, our heuristics allow to solve the problems on large graphs with millions of nodes and edges adding any amount of edges in reasonable time and this means that our methods can be applied in real world systems where having a high harmonic centrality or betweenness centrality can have a positive impact on the node itself.

5.1 Open problems and future research directions

There are a lot of open problems related to CM that deserve further investigation. In the following we list several possible research directions.

- First of all, it would be interesting to close open cases pointed out in Table 1.1 and to close the gaps between approximation and inapproximability results. In particular, it would be nice to design an approximation algorithm for CM-B on undirected graphs, because we shown in Section 4.4.2 that the greedy algorithm can have an unbounded approximation ratio. Moreover, it would be interesting to study the CM problem with other centrality indices.
- Sometimes in real networks it can be easier to create a link to a closer node (e.g a neighbour of your neighbours) instead of a farther one. It would be nice to define and study CM problem with edge cost based on this property.
- It is easy to imagine a scenario in which two or more nodes try to increase their centrality by adding new edges. In such context, the best strategy that each node should adopt might be different from the greedy one. It would then be interesting to study this scenario from a game theoretic perspective.
- To capture the centrality of a group or a particular class of nodes, the notion of centrality index of a node can be extended to a set of nodes in the graph [37]. Informally, given a centrality index c and a subset of nodes $U \subseteq V$, the *group centrality index* c on U is the centrality computed without taking into account the edges with endpoints in U (we can consider U as a single "virtual" node). It would be worth to study the CM problem with the aim to maximize the group centrality of a certain group of vertices.
- The dynamic algorithms described in Chapter 3 and in [12] are based on shortest path updating. In order to speed up the computation and analyse larger networks, it would nice to include distance estimators like the sketches described in [24, 25].
- It would be interesting to study the "dual" problem of CM in which we want to minimize the centrality of a given node by deleting edges incident to that node.

Examples are applications in which one wants to reduce the traffic flow in nodes of a road or a communication networks or reduce the spread of disease in epidemic and social networks.

- Finally, it would be challenging to study the problem where it is possible to add edges incident to other vertices or weight changes to the existing edges.

Bibliography

[1] Arnetminer. http://arnetminer.org. Accessed: 2015-01-15.

[2] GLPK – GNU Linear Programming Kit. http://www.gnu.org/software/glpk.

[3] R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

[4] K. Avrachenkov and N. Litvak. The effect of new links on google pagerank. *Stochastic Models*, 22(2):319–331, 2006.

[5] L. Backstrom and J. Leskovec. Supervised random walks: Predicting and recommending links in social networks. In *Proceedings of the 4th ACM International Conference on Web Search and Data Mining*, pages 635–644. ACM, 2011.

[6] Z. Bar-Yossef and L.-T. Mashiach. Local approximation of pagerank and reverse pagerank. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 279–288. ACM, 2008.

[7] A.-L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[8] R. Bauer, G. D'Angelo, D. Delling, A. Schumm, and D. Wagner. The shortcut problem - complexity and algorithms. *J. Graph Algorithms Appl.*, 16(2):447–481, 2012.

[9] A. Bavelas. A mathematical model for group structures. *Human organization*, 7(3):16–30, 1948.

[10] P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Wächter. Branching and bounds tightening techniques for non-convex minlp. *Optimization Methods and Software*, 24(4-5):597–634, 2009.

[11] E. A. Bender and E. Canfield. The asymptotic number of labeled graphs with given degree sequences. *Journal of Combinatorial Theory, Series A*, 24(3):296 – 307, 1978.

[12] E. Bergamini, P. Crescenzi, G. D'Angelo, H. Meyerhenke, L. Severini, and Y. Velaj. Improving the betweenness centrality of a node by adding links. *arXiv preprint arXiv:1702.05284*, 2017.

[13] K. Bharat, A. Broder, M. Henzinger, P. Kumar, and S. Venkatasubramanian. The connectivity server: Fast access to linkage information on the web. *Computer networks and ISDN Systems*, 30(1):469–477, 1998.

[14] D. Bilò, L. Gualà, and G. Proietti. Improved approximability and non-approximability results for graph diameter decreasing problems. *Theoretical Computer Science*, 417:12–22, 2012.

[15] P. Boldi, M. Rosa, and S. Vigna. Hyperanf: Approximating the neighbourhood function of very large graphs on a budget. In *Proceedings of the 20th international conference on World wide web*, pages 625–634. ACM, 2011.

[16] P. Boldi and S. Vigna. Four degrees of separation, really. In *Advances in Social Networks Analysis and Mining (ASONAM), 2012 IEEE/ACM International Conference on*, pages 1222–1227. IEEE, 2012.

[17] P. Boldi and S. Vigna. Axioms for centrality. *Internet Mathematics*, 10(3–4):222–262, 2014.

[18] B. Bollobás, C. Borgs, J. Chayes, and O. Riordan. Directed scale-free graphs. In *Proceedings of the 14th annual ACM-SIAM symposium on Discrete algorithms*, pages 132–139. SIAM, 2003.

[19] M. Borassi and E. Natale. KADABRA is an ADaptive Algorithm for Betweenness via Random Approximation. In *Proceedings of the 24th Annual European Symposium on Algorithms*, volume 57, pages 20:1–20:18. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016.

[20] U. Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25(2):163–177, 2001.

[21] A. Z. Broder, R. Lempel, F. Maghoul, and J. Pedersen. Efficient pagerank approximation via graph aggregation. *Information Retrieval*, 9(2):123–138, 2006.

[22] Y.-Y. Chen, Q. Gan, and T. Suel. Local methods for estimating pagerank values. In *Proceedings of the 13th ACM international conference on Information and knowledge management*, pages 381–389. ACM, 2004.

[23] F. Chierichetti, R. Kumar, S. Lattanzi, A. Panconesi, and P. Raghavan. Models for the compressible web. In *Proceedings of the 50th Annual Symposium on Foundations of Computer Science*, pages 331–340. IEEE, 2009.

[24] E. Cohen. All-distances sketches, revisited: Hip estimators for massive graphs analysis. *IEEE Transactions on Knowledge and Data Engineering*, 27(9):2320–2334, 2015.

[25] E. Cohen, D. Delling, T. Pajor, and R. F. Werneck. Computing classic closeness centrality, at scale. In *Proceedings of the 2nd ACM conference on Online social networks*, pages 37–50. ACM, 2014.

[26] T. H. Cormen. *Introduction to algorithms.* MIT press, 2009.

[27] P. Crescenzi, G. D'Angelo, L. Severini, and Y. Velaj. Greedily improving our own centrality in a network. In *Proceedings of the 14th International Symposium on Experimental Algorithms*, volume 9125 of *Lecture Notes in Computer Science*, pages 43–55. Springer, 2015.

[28] P. Crescenzi, G. D'Angelo, L. Severini, and Y. Velaj. Greedily improving our own closeness centrality in a network. *ACM Transactions on Knowledge Discovery from Data*, 11(1):9:1–9:32, 2016.

[29] P. Crescenzi, R. Grossi, M. Habib, L. Lanzi, and A. Marino. On computing the diameter of real-world undirected graphs. *Theoretical Computer Science*, 514:84–95, 2013.

[30] G. D'Angelo, L. Severini, and Y. Velaj. On the maximum betweenness improvement problem. *Electronic Notes in Theoretical Computer Science*, 322:153 – 168, 2016. Proceedings of the 16th Italian Conference on Theoretical Computer Science (ICTCS15).

[31] S. Dehghani, M. A. Fazli, J. Habibi, and S. Yazdanbod. Using shortcut edges to maximize the number of triangles in graphs. *Operations Research Letters*, 43(6), 2015.

[32] E. D. Demaine and M. Zadimoghaddam. Minimizing the diameter of a network using shortcut edges. In *Proceedings of the 12th Scandinavian Symposium and Workshops on Algorithm Theory*, volume 6139 of *Lecture Notes in Computer Science*, pages 420–431. Springer, 2010.

[33] I. Dinur and D. Steurer. Analytical approach to parallel repetition. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 624–633. ACM, 2014.

[34] D. Eppstein and J. Wang. Fast approximation of centrality. In *Proceedings of the 12th annual ACM-SIAM symposium on Discrete algorithms*, pages 228–229. Society for Industrial and Applied Mathematics, 2001.

[35] P. Erdős and A. Rényi. On random graphs I. *Publicationes Mathematicae*, 6:290–297, 1959.

[36] D. Erdős, V. Ishakian, A. Bestavros, and E. Terzi. A divide-and-conquer algorithm for betweenness centrality. In *Proceedings of the 2015 SIAM International Conference on Data Mining*. SIAM, 2015.

[37] M. G. Everett and S. P. Borgatti. The centrality of groups and classes. *The Journal of Mathematical Sociology*, 23(3):181–201, 1999.

[38] M. G. Everett and T. W. Valente. Bridging, brokerage and betweenness. *Social Networks*, 44:202–208, 2016.

[39] U. Feige. A threshold of ln n for approximating set cover. *Journal of the ACM*, 45(4), 1998.

[40] F. Frati, S. Gaspers, J. Gudmundsson, and L. Mathieson. Augmenting graphs to minimize the diameter. *Algorithmica*, 72(4):995–1010, 2015.

[41] L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41, 1977.

[42] M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, New York, 1979.

[43] D. Gleich, L. Zhukov, and P. Berkhin. Fast parallel pagerank: A linear system approach. *Yahoo! Research Technical Report YRL-2004-038, available via http://research. yahoo. com/publication/YRL-2004-038. pdf*, 13:22, 2004.

[44] IMDB. http://www.imdb.com. Accessed: 2015-01-15.

[45] V. Ishakian, D. Erdös, E. Terzi, and A. Bestavros. A framework for the evaluation and management of network centrality. In *Proceedings of the 2012 SIAM International Conference on Data Mining*, pages 427–438. SIAM, 2012.

[46] R. Jacob, D. Koschützki, K. A. Lehmann, L. Peeters, and D. Tenfelde-Podehl. Algorithms for centrality indices. In *Network Analysis*, pages 62–82. Springer, 2005.

[47] S. Kamvar, T. Haveliwala, and G. Golub. Adaptive methods for the computation of pagerank. *Linear Algebra and its Applications*, 386:51–65, 2004.

[48] U. Kang, C. E. Tsourakakis, A. P. Appel, C. Faloutsos, and J. Leskovec. Hadi: Mining radii of large graphs. *ACM Trans. Knowl. Discov. Data*, 5(2):8:1–8:24, Feb. 2011.

[49] D. Kempe, J. Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. *Theory of Computing*, 11(4):105–147, 2015.

[50] J. Kleinberg. The small-world phenomenon: An algorithmic perspective. In *Proceedings of the 32nd annual ACM symposium on Theory of computing*, pages 163–170. ACM, 2000.

[51] C. Kohlschütter, P.-A. Chirita, and W. Nejdl. Efficient parallel computation of pagerank. In *Proceedings of the 28th European Conference on Information Retrieval*, pages 241–252. Springer, 2006.

[52] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. Stochastic models for the web graph. In *Proceedings ot the 41st Annual Symposium on Foundations of Computer Science*, pages 57–65. IEEE, 2000.

[53] J. Kunegis. KONECT - The Koblenz network collection. In *Proceedings of the 1st International Web Observatory Workshop*, pages 1343–1350, 2013.

[54] Laboratory for Web Algorithmics. http://law.di.unimi.it/index.php. Accessed: 2015-01-15.

[55] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data*, 1(1), Mar. 2007.

[56] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 420–429. ACM, 2007.

[57] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

[58] M. Ley. DBLP. http://dblp.uni-trier.de/. Accessed: 2015-01-15.

[59] R. Li and J. X. Yu. Triangle minimization in large networks. *Knowledge and Information Systems*, 45(3):617–643, 2015.

[60] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *Proceedings of the 12th International Conference on Information and Knowledge Management*, pages 556–559. ACM, 2003.

[61] B. Macdonald, P. Shakarian, N. Howard, and G. Moores. Spreaders in the network SIR model: An empirical study. *CoRR*, abs/1208.4269, 2012.

[62] P. Malighetti, G. Martini, S. Paleari, and R. Redondi. The impacts of airport centrality in the EU network and inter-airport competition on airport efficiency. Technical Report MPRA-7673, 2009.

[63] M. Marchiori and V. Latora. Harmony in the small-world. *Physica A: Statistical Mechanics and its Applications*, 285(3):539–546, 2000.

[64] F. McSherry. A uniform approach to accelerated pagerank computation. In *Proceedings of the 14th international conference on World Wide Web*, pages 575–582. ACM, 2005.

[65] A. Meyerson and B. Tagiku. Minimizing average shortest path distances via shortcut edge addition. In *Proceedings of the 12th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, volume 5687 of *Lecture Notes in Computer Science*, pages 272–285. Springer, 2009.

[66] S. Milgram. The small world problem. *Psychology today*, 2(1):60–67, 1967.

[67] M. Minoux. Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization Techniques*, pages 234–243. Springer, 1978.

[68] M. Molloy and B. Reed. A critical point for random graphs with a given degree sequence. *Random Structures & Algorithms*, 6(2–3):161–180, 1995.

[69] G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of approximations for maximizing submodular set functions–I. *Mathematical Programming*, 14(1):265–294, 1978.

[70] M. E. Newman. The structure and function of complex networks. *SIAM review*, 45(2):167–256, 2003.

[71] M. Olsen and A. Viglas. On the approximability of the link building problem. *Theoretical Computer Science*, 518:96–116, 2014.

[72] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: bringing order to the web. 1999.

[73] C. R. Palmer, P. B. Gibbons, and C. Faloutsos. Anf: A fast and scalable tool for data mining in massive graphs. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 81–90. ACM, 2002.

[74] C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43(3):425–440, 1991.

[75] M. Papagelis. Refining social graph connectivity via shortcut edge addition. *ACM Transactions on Knowledge Discovery from Data*, 10(2):12, 2015.

[76] M. Papagelis, F. Bonchi, and A. Gionis. Suggesting ghost edges for a smaller world. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, pages 2305–2308. ACM, 2011.

[77] N. Parotsidis, E. Pitoura, and P. Tsaparas. Selecting shortcuts for a smaller world. In *Proceedings of the 2015 SIAM International Conference on Data Mining*, pages 28–36. SIAM, 2015.

[78] N. Parotsidis, E. Pitoura, and P. Tsaparas. Centrality-aware link recommendations. In *Proceedings of the 9th ACM International Conference on Web Search and Data Mining*, pages 503–512. ACM, 2016.

[79] S. Perumal, P. Basu, and Z. Guan. Minimizing eccentricity in composite networks via constrained edge additions. In *Proceedings of the 32th IEEE Military Communications Conference*, pages 1894–1899. IEEE, 2013.

[80] A. Popescul and L. H. Ungar. Statistical relational learning for link prediction. In *IJCAI workshop on learning statistical models from relational data*, 2003.

[81] M. Riondato and E. M. Kornaropoulos. Fast approximation of betweenness centrality through sampling. *Data Mining and Knowledge Discovery*, 30(2):438–475, 2016.

[82] M. Riondato and E. Upfal. ABRA: approximating betweenness centrality in static and dynamic graphs with rademacher averages. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1145–1154. ACM, 2016.

[83] S. Saha, A. Adiga, B. A. Prakash, and A. K. S. Vullikanti. Approximation algorithms for reducing the spectral radius to control epidemic spread. In *Proceedings of the 2015 SIAM International Conference on Data Mining*, pages 568–576. SIAM, 2015.

[84] A. E. Sariyüce, K. Kaya, E. Saule, and Ü. V. Çatalyürek. Incremental algorithms for closeness centrality. In *Proceedings of the 2013 IEEE International Conference on Big Data*, pages 487–492. IEEE, 2013.

[85] A. E. Sariyüce, E. Saule, K. Kaya, and Ü. V. Çatalyürek. Shattering and compressing networks for betweenness centrality. In *Proceedings of the 2013 SIAM International Conference on Data Mining*. SIAM, 2013.

[86] A. Slobbe, E. Bergamini, and H. Meyerhenke. Faster incremental all-pairs shortest paths. *Karlsruhe Reports in Informatics*, 2016.

[87] C. L. Staudt, A. Sazonovs, and H. Meyerhenke. Networkit: An interactive tool suite for high-performance network analysis. *arXiv preprint arXiv:1403.3005*, 2014.

[88] F. W. Takes and W. A. Kosters. Computing the eccentricity distribution of large graphs. *Algorithms*, 6(1):100–118, 2013.

[89] H. Tong, B. A. Prakash, T. Eliassi-Rad, M. Faloutsos, and C. Faloutsos. Gelling, and melting, large graphs by edge manipulation. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pages 245–254. ACM, 2012.

[90] Uri alonlab. http://www.weizmann.ac.il/mcb/UriAlon/. Accessed: 2015-01-15.

[91] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. In *Measures of Complexity*, pages 11–30. Springer, 2015.

[92] F. Vella, G. Carbone, and M. Bernaschi. Algorithms and heuristics for scalable betweenness centrality computation on multi-gpu systems. *arXiv preprint arXiv:1602.00963*, 2016.

[93] S. Wasserman and K. Faust. *Social network analysis: Methods and applications*, volume 8. Cambridge university press, 1994.

[94] S. Wasserman and K. Faust. *Social network analysis: Methods and applications*, volume 8. Cambridge university press, 1994.

[95] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998.

[96] V. V. Williams. Multiplying matrices faster than coppersmith-winograd. In *Proceedings of the 44th annual ACM symposium on Theory of computing*, pages 887–898. ACM, 2012.

[97] D. Williamson and D. Shmoys. *The Design of Approximation Algorithms.* Cambridge University Press, 2011.

[98] E. Yan and Y. Ding. Applying centrality measures to impact analysis: A coauthorship network analysis. *Journal of the Association for Information Science and Technology*, 60(10):2107–2118, 2009.

[99] Z. Yin, M. Gupta, T. Weninger, and J. Han. A unified framework for link recommendation using random walks. In *Proceedings of the 2010 International Conference on Advances in Social Networks Analysis and Mining*, pages 152–159. IEEE Computer Society, 2010.