ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

MASTER SEMESTER PROJECT - REPORT

PROJECT ADVISOR: ANNE-MARIE KERMARREC
PROJECT SUPERVISOR: OTHMANE SAFSAFI

RECONSTRUCTION OF A d-REGULAR DIGRAPH FROM ITS PAGERANK AND A SET OF EDGES

SCALABLE COMPUTING SYSTEM LABORATORY (SACS)

LUDMILA COURTILLAT--PIAZZA

11th June 2022

Abstract

In the following, we introduce the problem of reconstruction of a directed quasi-d-regular graph from its pageRank and a set of edges and we prove that this problem is NP-Complete. We also propose an algorithm based on back-tracking that solves the problem exactly and an efficient algorithm to approximate a solution, based on known algorithms of approximation for SUBSET-SUM problem. Finally, we present the performances of our algorithms on artificial and real graphs.

CHAPTER 1

INTRODUCTION

Introduced on 1999 by Google's founders, the pageRank algorithm Page et al. 1999 revolutionized online searches, offering results perceived by users as much more relevant than those offered by its competitors at the time and is today used by a large majority of the browsers. It is based on a vision of the web as a directed graph in which pages represent nodes and hyperlinks represent edges. The pageRank is then a metric that models *the average proportion of time spent by a random surfer on each node in a graph*. This metric therefore gives more importance to nodes in the graph with a large number of incoming edges. It provides us with information about the nodes themselves, but also about the overall structure of the graph.

What information does the pageRank contain exactly? This project is part of a larger effort to answer this question.

During this semester project we managed to reconstruct graphs for which we only know the pageRank of the nodes and a set of edges (possibly empty). As this problem looks hard in the case of really large graphs (like Youtube or the Web), we have chosen to approach the question in the form of ego-graphs, centered on a node called *root* or *entry-point*. Even in this case, that may look simpler, we prove that the problem is NP-hard.

Let us present our motivations and hypothesis for the problem we state.

1.1 MOTIVATIONS & HYPOTHESES

1.1.1 THE RECOMMENDATION SYSTEMS

In real life, the pageRank is not a public data. We can only compute it by extracting the graphs we are interested in and doing the computations our-self. In this context, deducing the graph topology from its pageRank doesn't seem to be an interesting question as we need to know the graph to compute the pageRank.

However, there exists in real life public metrics that can be good approximations of the pageRank. This is the case of the *number of views* in the context of a recommendation system. Indeed, if we assume that

- a recommendation system is a static graph in which the nodes are contents like videos, musics, posts, films and a directed edge $a \to b$ exists between a node a and a node b if and only if users are offered a link to b when they view content a.
- viewers can be approximated by a random surfer that choose a video/a film/a music among those proposed after viewing one,

in this case, the number of views measure the same information than the pageRank up to scaling.

The precedent assumptions are flawed, primarily because actual recommendation systems are not static. This problem can be solved by doing two snapshots on the number of views close enough in time to keep the recommendation system quasi static (and long enough to get a sufficient amount of views to approximate pageRank). The weak points of this hypotheses will be discussed more extensively in the section Limitations.

The application of *number of views in a recommendation system* motivate another assumption we made: the graphs we work with are supposed to have *quasi-d-regular out-degree*. This models the fact that a recommendation system will give the same number of recommendations after each content.

FOCUS ON EGO-GRAPHS

Today's recommendation systems (Youtube, Netflix) are too large to be processed entirely. This is why we chose to study partial graph in order to stay on realistic settings for this semester project.

The *ego-graph* perspective is the one we chose in order to extract a part of a graph to study without creating abrupt cuts of important edges or nodes in the structure we choose to keep.

The idea of ego-graph comes from social networks study and consists in focusing in the close surrounding of one node (one user in the context of social networks).

We adapt the notion of pageRank such that the random walks can jump to the central node of the graph (rather than to any node of the graph uniformly at random) at each step with some probability c. In the traditional notion of pageRank, this can be interpreted as the fact a user stops its navigation and a new user begin a navigation from an arbitrary point. In our ego-graph based model, each new navigation/random walk starts from the central/entry-point node. This can be assimilated to the home page of a website with recommendation system on contents like youtube. In this way we can choose to keep in our graphs only nodes with a sufficient probability to be reached from the entry-point and the cuts we make in the whole graph are unlikely to be damaging with respect to this metric (the ego-pageRank).

1.2 RELATED WORK

The graph reconstruction is an extensively studied subject, with in particular the recent work of C. Mathieu and H. Zhou (Mathieu and Zhou 2021 Kannan, Mathieu and Zhou 2018). But these articles propose reconstruction using distance queries.

Others authors use a notion of random walk that may be related with pageRank (Hoskins et al. 2018, He et al. 2014, Fontoura Costa and Travieso 2007, Wittmann et al. 2009).

In Wittmann et al. 2009, the authors reconstruct the adjacency matrix of a graph from the matrix of the expected average lengths of a random walk leading from one node to another, that is a notion of the distances between the nodes based on random walks.

In Hoskins et al. 2018, they base their reconstruction on a notion of *resistance* that also reflects the time spend by a random walker to go from one node to another.

In our approach we do not suppose any knowledge about the distances between the nodes, and this is why these works are really different from ours.

In Fontoura Costa and Travieso 2007, the authors use random walk to reconstruct a graph but they suppose they can do random walks themselves and search the the best way to cover the graph with this random walks.

In this semester project, we suppose that we cannot access the graph directly and therefore we cannot simply discover the edges by searching the graph.

In He et al. 2014, the authors do not suppose they can access the edges met during the random walks but they have the information of the intermediate 'pageRank' (called here *distribution*) of the graph after each one of M random walks. In contrast we suppose we only have one measure of the pageRank.

1.3 CONTRIBUTIONS

- We introduce the problem of reconstruction of a quasi-d-regular graph from its pageRank and a set of edges.
- We prove that this problem is NP-Complete with a reduction from the known to be NP-Complete SUBSETSUM problem (Cormen et al. 2007, Chapter 35.5, « The subset-sum problem »).
- We propose a exact deterministic algorithm based on back-tracking to solve the problem.
- Finally, we present an efficient approximation algorithm, based on known SUBSETSUM approximation algorithm, that we test on both artificial and real graphs.

1.4 CONTENTS

In chapter 2, we present preliminary definitions and state the problems we work with. In chapter 3, we prove the NP-completness of the problem of reconstruction we introduced. In chapter 4, we present our algorithms of reconstruction and we test it on both artificial and real graphs in chapter 5. In chapter 6, we discuss the limitations of our approach and future works on it. We conclude with section 7.

CHAPTER 2

PRELIMINARY NOTIONS

2.1 DEFINITIONS

We introduce here a notion of pageRank adapted with our use of ego-graphs that we call *ego-pageRank*.

Definition 1 (ego-pageRank)**.** *Let $G = (V, E)$ be a directed graph with $V = \{v_0, v_1, \cdots, v_{n-1}\}$ the set of vertices of G and E the set of edges of G. Let c be a* jump probability*. We assume v_0 to be the 'entry-point' (or* root*) of the graph. The ego-pageRank of (G, c) is a vector of probabilities $\{p_0, \cdots, p_{n-1}\}$ that associates to each node v_i of V the probability, for a random surfer beginning a random walk at vertex v_0 and jumping to node v_0 with probability c at each step, to pass by the node v_i before jumping to v_0. The ego-pageRank of G is the solution of*

$$p_i = \begin{cases} \frac{1-c}{d} \sum\limits_{j:(v_j,v_i)\in E} p_j & if \;\; i \neq 0 \\ 1 + \frac{1-c}{d} \sum\limits_{j:(v_j,v_i)\in E} p_j & if \;\; i = 0 \end{cases}$$

If $(v_j \to v_i) \in E$ the quantity $\frac{1-c}{d} p_j$ is called the contribution *of the node v_j to the ego-pageRank of the node v_i. The ego-pageRank of v_i is thus the sum of the contributions of its predecessors/ingoing edges. By extension, the contribution of the edge $(v_j \to v_i) \in E$ is also called the* ego-pageRank *of this edge.*

The probability c models the chance that a random surfer will stop exploring at the current node. It then starts a new exploration from the entry-point. Thus, $1/c$ is the average length of an exploration in this model. This could reflect a sequence of video viewing/ music listening following a recommendation system's suggestions (graph G) and starting from a home page (node v_0).

By the fixed-point theorem the ego-pageRank is well-defined and unique. In the following, we note $pageRank(G, c)$ this unique vector of probabilities.

In the following we refer to *ego-pageRank* as *pageRank* for short.

Remark. *The main difference between* ego-pageRank *and more traditional notions of pageRank consists in the fact that the random surfer always begins its random walk from the same entry-point. This point is necessary with ego-graphs so that the nodes furthest from the entry-point have the lowest pageRank.*

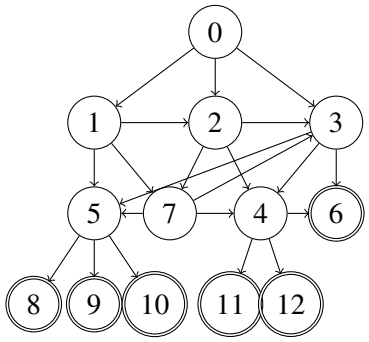

FIGURE 2.1
Example of *quasi-d-regular graph*

Definition 2 (Quasi-d-regular graph)**.** *A directed graph $G = (V, E)$ is said to be* quasi-d-regular *if there exists a subset of vertices L of V, that are called* leafs *such that each vertex in $V \setminus L$ admits exactly d outgoing edges and each vertex in L admits no outgoing edge (this is why it is called a leaf):*

- $\forall u \in V \setminus L,\ \#|\{v : (u, v) \in E\}| = d$
- $\forall u \in L,\ \#|\{v : (u, v) \in E\}| = 0$

The notion of *quasi-d-regular* reflects the concept of ego-graph in the context of d-regular graphs in that a quasi-d-regular graph is a d-regular graph in which we remove the nodes that are too far away from the entry-point (that would have been the successors of the leafs).

Definition 3 (Depth)**.** *In a quasi-d-regular graph, the depth of node is the length of the shortest path from the entry-point to this node.*
By extension, the depth of an edge is the depth of node from witch it comes.

Example. In figure 2.1, the node 0 has depth 0, the nodes 1, 2, 3 has depth 1 and so on. The edge from node 5 to node 8 has depth 2 and the node 5 has depth 2.

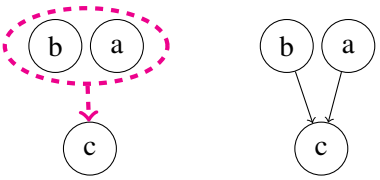

FIGURE 2.2
Example of a *meta-node* and a *meta-edge*

Definition 4 (Meta-node)**.** *A* meta-node *is a group of nodes that we choose to view as a unique one. An outgoing edge from a meta-node (i.e. a* meta-edge*) corresponds to an edge from each one of the nodes forming the meta-node. The pageRank of a meta-node is the sum of the pageRanks of its members. Hence, the equations on pageRank remains consistent when we consider meta-nodes and meta-edges.*

2.2 PROBLEMS

We present the problem we are interested in.

Problem 1: (d, c)-$graphFromPR^*$

parameters: $c \in]0,1[$, $d \in \mathbb{N}$; $d \geq 2$
input: a tuple (V, P, L, H) with $V = \{v_0, ..., v_{n-1}\}$ a set of vertices, $P = \{p_0^*, ..., p_{n-1}^*\}$ a pageRank vector, $L = \{l_0, ..., l_m\}$ a subset of V, H a set of directed edges V
output: a digraph $G = (V, E)$, a *quasi-d-regular* digraph such that, V is the set of vertices of G, the labeled graph $H \subset E$, L is the set of leafs of G and $PageRank(G, c) = P$.

The problem (d, c)-$graphFromPR$ consists in, given a set of nodes V among those we know which node should be a leaf or not (the leafs are in L), an associate measure of pageRank P for each node and a set of edges between this nodes, complete the edges of the graph to obtain a quasi-d-regular graph G, with pageRank P, nodes V and leafs L.

Remark: Note that the instances of (d, c)-$graphFromPR^*$ problem are supposed to admit at least one solution.

Theorem 1. *A solution for an instance of* (d, c)-$graphFromPR^*$ *is not necessarily unique.*

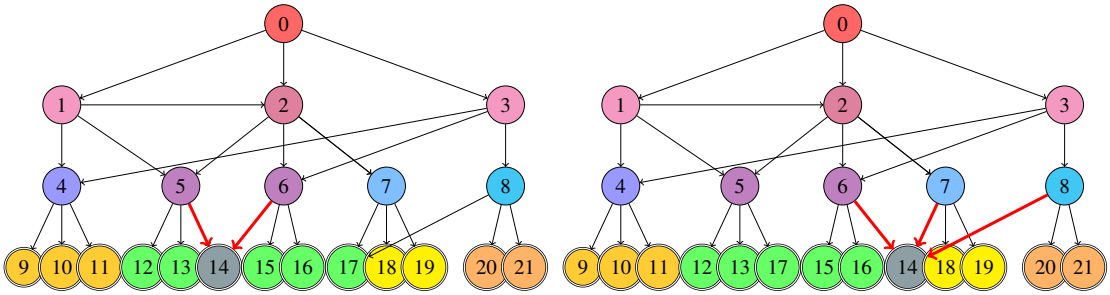

FIGURE 2.3
2 solutions for the same instance of (d, c)-$graphFromPR^*$: $V = [\![0; 21]\!], P = [\![0 : 1, 1 : 1/3, 2 : 4/9, 3 : 1/3, 4 : 2/9, 5 : 7/27, 6 : 7/27, 7 : 4/27, 8 : 1/9, 9 : 2/27, 10 : 2/27, 11 : 2/27, 12 : 7/54, 13 : 7/54, 14 : 14/54, 15 : 7/54, 16 : 7/54, 17 : 16/54, 18 : 4/54, 19 : 4/54, 20 : 1/27, 21 : 2/27]\!], L = [\![9; 21]\!], H = \{(0, 1), (2, 7), (8, 21)\}$

Proof. See a conter-exemple on figure 2.3. Note that, when two nodes have the same pageRank and are both leafs or both non-leafs nodes, we can always find a new solution from another one by exchanging the positions of the two nodes in the graph, because we have exactly the same information about the two nodes. But in the case we present in Figure 2.3, the structure of the two solutions are different, i.e. if we don't know the indices of the nodes, the two graphs are still note the same. Indeed, node 14, that is the only one to get pageRank $14/54$, has two parents with pageRank $7/27$ in one case and three parents with pageRank $1/9$, $4/27$ and $7/27$ in the other case. □

In order to study the complexity of this reconstruction problem, let us introduce the associated decision problem (d, c)-$graphFromPR$ that will be useful for the proofs.

Problem 2: (d, c)-$graphFromPR$

parameters: $c \in]0,1[$, $d \in \mathbb{N}$; $d \geq 2$
input: A tuple (V, P, L, H) with $V = \{v_0, ..., v_{n-1}\}$ a set of vertices, $P = \{p_0^*, ..., p_{n-1}^*\}$ a pageRank vector, $L = \{l_0, ..., l_m\}$ a subset of V, H a set of directed edges on V
output: Yes, if there exists a *quasi-d-regular* digraph $G = (V, E)$ such that, V is the set of vertices of G, the labeled graph $H \subset E$, L is the set of leafs of G and $PageRank(G, c) = P$, no otherwise.

Claim 1. *If (d, c)-$graphFromPR$ is NP-hard, (d, c)-$graphFromPR^*$ is as complex as (d, c)-$graphFromPR$ in time.*

Remark. *As (d, c)-$graphFromPR^*$ is not a decision problem, we cannot say that it is NP-hard, but the precedent claim roughly states that the NP-hardness of (d, c)-$graphFromPR$ is equivalent to (d, c)-$graphFromPR^*$ one.*

Proof. Suppose (d, c)-$graphFromPR$ is NP-hard. Let (V, P, L, H) be an instance of (d, c)-$graphFromPR$ (the decision problem). Suppose we have an algorithm A that solves(d, c)-$graphFromPR^*$ in time complexity $C(V, P, L, H)$. We describe an algorithm that solves the decision problem (d, c)-$graphFromPR$ using A.
Let m be the maximal number of steps taken by a Turing Machine to execute algorithm A on (V, P, L, H) when supposing it is a positive instance of (d, c)-$graphFromPR^*$. (Given the algorithm A, we can compute exactly its time complexity on instances that satisfy the assumptions made by the problem (d, c)-$graphFromPR^*$ it solves). Then we can run $A(V, P, L, H)$ on a Turing Machine and stop it after $m + 1$ steps if it has not terminated before. Then,

- if the execution has not terminated, we conclude (V, P, L, H) is not a positive instance of (d, c)-$graphFromPR$, otherwise $A(V, P, L, H)$ would have terminated before.
- if the execution terminated, it returned a digraph $G(V', E)$. Let $n = |V|$ and $\{p_0, ..., p_{n-1}\}$ be the set of pageRank of G. We can check in time polynomial in n that
  - $V' = V$
  - $L' = \{u \in V' \ \nexists v \in V' \mid (u, v) \in E\} = L$
  - $H \in E$
  - $\forall u \in V' \setminus L', \ \#\{v : (u, v) \in E\} = d$
  - $$\forall i \in [\![0, n-1]\!] \quad p_i = \begin{cases} \frac{1-c}{d} \sum_{j:(v_j,v_i)\in E} p_j & if \ \ i \neq 0 \\ 1 + \frac{1-c}{d} \sum_{j:(v_j,v_i)\in E} p_j & if \ \ i = 0 \end{cases}$$

    by unicity of the pageRank, it implies that $PageRank(G, c) = P$

  Then we can check in time $Q(n)$ polynomial in n that G is a certificate of (V, P, L, H) for the (d, c)-$graphFromPR$ problem.

The algorithm to solve the decision problem we just described has a time complexity of $C(V, P, L, H) + Q(n)$ in the worst case and solves an NP-hard problem. As $Q(n)$ is polynomial in (V, P, L, H) and $C(V, P, L, H) + Q(n)$ is the complexity of an NP-hard problem, $C(V, P, L, H)$ is as complex in time as an algorithm solving an NP-hard problem, and (d, c)-$graphFromPR^*$ is as complex in time as an NP-hard problem. □

Finally, let us recall the definition of the classical NP-hard problem SUBSET-SUM

Problem 3: SUBSET-SUM

input: $P = (p_1, ..., p_n)$ a set of integers, t a target integer
output: yes, if there exists a subset of P that sums up to t, no otherwise.

CHAPTER 3

PROOF OF NP-COMPLETNESS OF (d, c)-$graphFromPR$

We will prove that we can reduce SUBSET-SUM to (d, c)-$graphFromPR$. With claim 1, it implies that the reconstruction of a quasi-d-regular graph from its pageRank is a problem as complex in time as a NP-hard decision problem. In addition, as certificate verification is polynomial in the number of nodes, we conclude that (d, c)-$graphFromPR$ is NP-Complete.

3.1 REDUCTION OF SUBSET-SUM TO (d, c)-$graphFromPR$

In this section, we reduce an instance (S, t) of **SubsetSum** to an instance (V, L, P, H) of (d, c)-$graphFromPR$ and prove then the NP-hardness of (d, c)-$graphFromPR$.

3.1.1 IDEA OF THE PROOF.

In the problems of reconstruction we are interested in, we want to reconstruct a graph from its pageRank and a set of edges. First, note that the pageRank of a node is the sum of the *contributions*(1) given by each of its predecessors edges. As we consider quasi-d-regular graphs, the contribution of each predecessor j to the pageRank of a node can be computed from the pageRank of this predecessor (this is $(1 - c)p_j/d$). In other words, we can associate to each outgoing edge of a node a *contribution* (or flow) $(1 - c)p_j/d$ it will give to its tail node. As each successor of a node has the same probability to be joined by a step from this one, and as we do not allow multiple-edges, the notion of contribution of a predecessor and contribution of a given edge from this predecessor are the same.

Then, in order to obtain the right pageRank for a given node i, we need to choose during the reconstruction a set of predecessors $v_1^i, ..., v_r^i$ for i from $V \setminus L$ such that their contributions $(1 - c)p_j/c$ sums up to p_i (and include the ingoing edges of i in H). Thus, we can see this question as a SUBSET-SUM problem with contributions of the nodes in $V \setminus L$ as the set and the pageRank p_i as the target.

Following this idea we reduce the problem SUBSET-SUM to (d, c)-$graphFromPR$ by generating in polynomial time an instance of (d, c)-$graphFromPR$ with a specific node which pageRank equals the desired target and each possible predecessor presents a contribution that equals one of the elements of the set of weights. Another node will be a well for the unused edges.

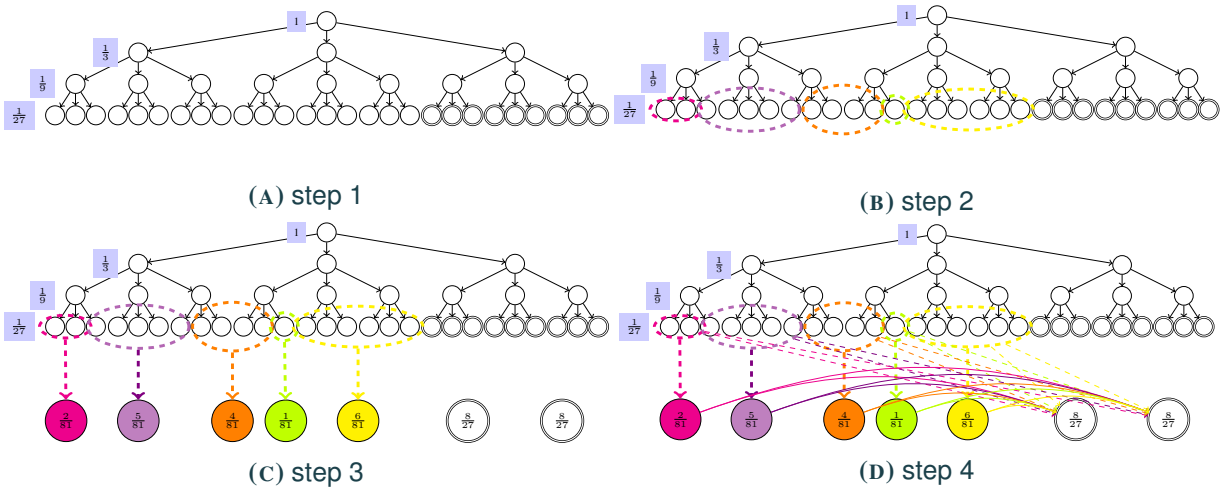

FIGURE 3.1
Example of reduction for $S = \{2, 5, 4, 1, 6\}$, $t = 8$. See figure 3.2 for more readability

THE CONSTRUCTION.

Let us consider an instance of SUBSET-SUM (S, t) with $S = \{s_1, ..., s_n\}$ a set of integers and t a target integer. We will construct an instance of (d, c)-$graphFromPR$ (V, P, L, H) such that (S, t) admits a certificate if and only if (V, P, L, H) admits one.

- Let $m = \sum_{s_i \in S} s_i$ and $a = \lceil (\log_d(m)) \rceil$.
- step 1: T is the the complete d-regular tree of height a. It has $d^a > m$ leafs. The root of the tree will be the entry-point of our graph, then it has a pageRank of 1 and at each depth i, the nodes of the tree of depth i have a pageRank of $((1-c)/d)^i$.
  V_1 and P_1 are the set of nodes of T and associated pageRanks. $L_1 = \emptyset$, $H_1 = T$.
- step 2: We classify the d^p leafs of T in $n+1$ meta-nodes $w_1, ..., w_{n+1}$ such that $w_1, ..., w_n$ are respectively equivalents to $s_1, ..., s_n$ leafs of T and w_{n+1} is equivalent to the $d^p - m$ remaining nodes. For $i = 1$ to n, the total pageRank of each meta-node w_i is given by $s_i((1-c)/d)^a$.
  $V_2 = V_1, P_2 = P_1, L_2 = \{w_{n+1}\}, H_2 = H_1$.
- step 3: We create n new nodes $v_1, ..., v_n$ such that each one has exactly all the members of a meta-node as predecessors. In other words, for all i in $[\![1, n]\!]$, v_i has for unique ingoing edge a meta-edge from w_i, that is an edge from each node in the meta-node w_i. The pageRank of v_i is $s_i((1-c)/d)^{a+1}$ as each one of the s_i leafs of T in w_i has an associate pageRank $(1 - c/d)^a$.
  $V_3 = V_2 \cup \{v_1, ..., v_n\}$, $P_3 = P_2 \cup \{s_1((1-c)/d)^{a+1}, ..., s_n((1-c)/d)^{a+1}\}$, $L_3 = L_2$,
  $H_3 = H_2 \cup \{(w_i, v_i)\ \forall i \in [\![1, n]\!]\}$
- step 4: We create $d-1$ nodes $t_1, ..., t_{d-1}$ that receive as parents all nodes that didn't complete their number of successors up to d, i.e. $w_1, ..., w_n$ and $v_1, ..., v_n$. This node has a pageRank of $m((1-c)/d)^{a+1} + m((1-c)/d)^{a+2}$
  $V_4 = V_3 \cup \{t_1, ..., t_{d-1}\}$,
  $P_4 = P_3 \cup \{m((1-c)/d)^{a+1} + m((1-c)/d)^{a+2}\ \ n\ times\}$,
  $L_4 = L_3 \cup \{t_1, ..., t_{d-1}\}$,
  $H_4 = H_3 \cup \{(w_i, t_j)\ \forall i \in [\![1, n]\!], j \in [\![1, d-1]\!]\} \cup \{(v_i, t_j)\ \forall i \in [\![1, n]\!], j \in [\![1, d-1]\!]\}$
- step 5: It now only remains one outgoing edge to each of the n nodes $v_1, ..., v_n$ with pageRank $s_i((1-c)/d)^{a+1}$ to be completed, we create two nodes v_{target} and v_{well} that will be leafs of our

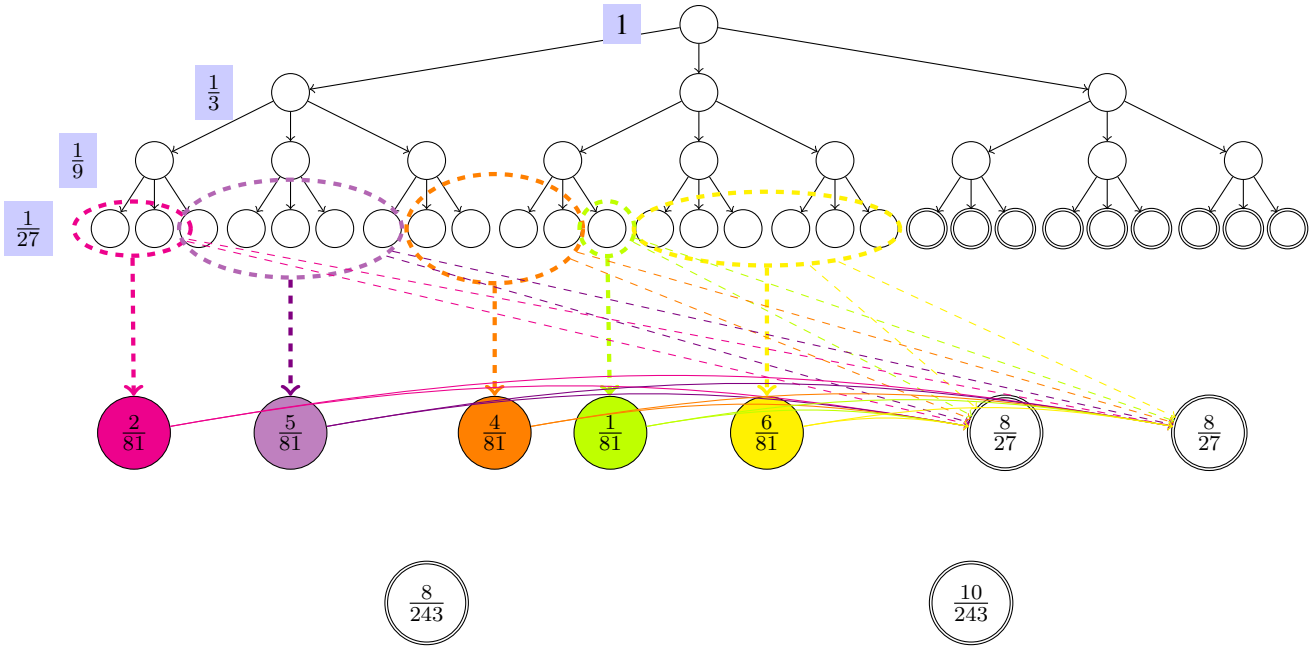

FIGURE 3.2
Complete instance of (d, c)-$graphFromPR^*$ corresponding to the instance of SUBSET-SUM instance $S = \{2, 5, 4, 1, 6\}$, $t = 8$

graph. v_{target} has PageRank $t((1-c)/d)^{a+2}$ and v_{well} has a pageRank $(m-t)((1-c)/d)^{a+2}$.
$V_5 = V_4 \cup \{v_{target}, v_{well}\}$,
$P_5 = P_4 \cup \{t((1-c)/d)^{a+2}, (m-t)((1-c)/d)^{a+2}\}$,
$L_5 = L_5 \cup \{v_{target}, v_{well}\}$

Finally, $V, P, L, H = V_5, P_5, L_5, H_5$.

Notation. *In the following we denote by $p[v]$ the pageRank associated to a node $v \in V$ in (V, P, L, H).*

CORRECTNESS.

In this section, we consider (S, t), an instance of SUBSET-SUM and (V, P, L, H), the associated instance of (d, c)-$graphFromPR$. we just describe the construction and we prove the following theorem.

Theorem 2. *(S, t) admits a certificate if and only if (V, P, L, H) admits a certificate.*

Proof. If there exist a subset $S' = \{s'_1, ..., s'_r\} \subset S$ that sums up to t, then calling $v'_1, ..., v'_r$ the associated nodes in V, $\sum_{i=1}^{r} p[v'_i] = \sum_{i=1}^{r} s'_i((1-c)/d)^{a+2} = t((1-c)/d)^{a+2}$, i.e. the pageRank of v_{target}.

In addition, the remaining edges, that comes from $\{v_1, ..., v_n\}/\{v'_1, ..., v'_r\}$ sum up to

$$\begin{aligned}\sum_{i=1}^{n} p[v_i] - \sum_{j=1}^{r} p[v'_i] &= \sum_{i=1}^{n} s_i \left(\frac{1-c}{d}\right)^{a+2} - \sum_{j=1}^{r} s_j \left(\frac{1-c}{d}\right)^{a+2} \\ &= \left(\frac{1-c}{d}\right)^{a+2} \left(\sum_{i=1}^{n} s_i - \sum_{j=1}^{n} s'_j\right) \\ &= (m-t)\left(\frac{1-c}{d}\right)^{a+2}\end{aligned}$$

that is the pageRank of v_{well}. Thus, $H \cup \{(v'_i, v_{target}),\ i \in [\![1, r]\!]\} \cup \{(v'', v_{well}),\ i \in \{v_1, ..., v_n\}/\{v'_1, ..., v'_r\}\}$ is a certificate of (d, c)-$graphFromPR$.

If there exist G a certificate of (V, P, L, H), then

Lemma 1. *There exists a subset $V' \in \{v_1, ..., v_n\}$ such that*

$$G = H \cup \{(v', v_{target}),\ v' \in V'\} \cup \{(v'', v_{well}),\ v'' \in \{v_1, ..., v_n\}/V'\}$$

Proof. (of the lemma)

(i) v_{target} and v_{well} are the only nodes that did not complete their pageRank with H, such that all the other nodes satisfies the equation of pageRank

$$p_i = \begin{cases} \dfrac{1-c}{d} \displaystyle\sum_{j:(v_j,v_i)\in E} p_j & if\ \ i \neq 0 \\ 1 + \dfrac{1-c}{d} \displaystyle\sum_{j:(v_j,v_i)\in E} p_j & if\ \ i = 0 \end{cases}$$

- the *root* v_0 has no ingoing edge and has pageRank 1.
- at each depth $i \geq 1$ of the tree, each node of depth i has exactly one parent of pageRank $((1-c)/d)^{i-1}$ and its pageRank is $((1-c)/d)^i = ((1-c)/d)((1-c)/d)^{i-1}$
- for $i \in [\![1, n]\!]$, the node v_i has exactly one meta-predecessor of pageRank $s_i((1-c)/d)^{a+1}$ (i.e. s_i predecessors of pageRank $((1-c)/d)^a$) and its pageRank is $s_i((1-c)/d)^{a+1} = ((1-c)/d) \sum_{j=1}^{s_i}((1-c)/d)^a = ((1-c)/d)p[w_i]$
- the nodes $t_1, ..., t_{d-1}$ have $v_1, ..., v_n$ and $w_1, ..., w_n$ as parents. For all i in $[\![1, n]\!]$, v_i has a pageRank $s_1((1-c)/d)^{a+1}$ and w_1 has a pageRank $s_i((1-c)/d)^a$. The nodes $t_1, ..., t_d$ have a pageRank

$$\begin{aligned} & m\left(\frac{1-c}{d}\right)^{a+1} + m\left(\frac{1-c}{d}\right)^{a+2} \\ &= \left(\frac{1-c}{d}\right) \sum_{i=1}^{n} s_i \left(\left(\frac{1-c}{d}\right)^{a} + \left(\frac{1-c}{d}\right)^{a+1}\right) \\ &= \left(\frac{1-c}{d}\right) \sum_{i=1}^{n} (p[w_i] + p[v_i]) \end{aligned}$$

- Finally, v_{target} and v_{well} have a non zero pageRank and have no ingoing edge in H, then it need to complete its paegRank with new edges

Then, v_{target} and v_{well} are the only nodes that did not completed their pageRank with H.

(ii) It only remains one outgoing edge from each one of the n nodes $v_1, ..., v_n$ to add to H to obtain a quasi-d-regular graph with set of nodes V and set of leafs L, in other words, all the nodes in $V \setminus L$ have exactly d outgoing edges in H, except $v_1, ..., v_n$ that have $d-1$ outgoing edges in H. Indeed,

- In T, that is the complete d-regular tree of depth a, all the nodes have exactly d outgoing edges, by definition of a d-regular tree, except the leafs, that forms the meta-nodes $w_1, ..., w_{n+1}$.
- For i in $[\![1, n]\!]$, the meta-node w_i has one outgoing edge to v_i and one outgoing edge to each one of the $d-1$ nodes $t_1, ..., t_{d-1}$ for a total of d outgoing edges.
- For i in $[\![1, n]\!]$, the node v_i has one outgoing edge to each one of the $d-1$ nodes $t_1, ..., t_{d-1}$ for a total of $d-1$ outgoing edges.
- The remaining nodes $w_{n+1}, t_1, ..., t_{d-1}, v_{target}, v_{well}$ are all in L. So they have no edges in H and they should not have any at all.

Hence, it only remains one outgoing edge from each one of the n nodes $v_1, ..., v_n$ to add to H to obtain a quasi-d-regular graph with set of nodes V and set of leafs L.

We conclude lemma (1) by (i) and (ii).

□

By lemma 1, There exists a subset $V' = \{v'_1, \cdots, v'_r\} \subset \{v_1, \cdots, v_n\}$ such that

$$G = H \cup \{(v', v_{target}),\ v' \in V'\} \cup \{(v'', v_{well}),\ v'' \in \{v_1, \cdots, v_n\}/V'\}$$

The nodes in $\{v_1, ..., v_n\}$ have pageRank $\{s_1((1-c)/d)^{a+1}, \cdots, s_n((1-c)/d)^{a+1}\}$. Then, by denoting $p[v'_1] = s'_1((1-c)/d)^{a+1}, \cdots, p[v'_r] = s'_r((1-c)/d)^{a+1}$, $s'_1, \cdots, s'_r$ are the elements of S associated with the nodes in V' and

$$\begin{aligned} p[v_{target}] &= \left(\frac{1-c}{d}\right) \sum_{i=1}^{r} p[v'_i] \\ t\left(\frac{1-c}{d}\right)^{a+2} &= \left(\frac{1-c}{d}\right) \sum_{i=1}^{r} s'_i \left(\frac{1-c}{d}\right)^{a+1} \\ t &= \sum_{i=1}^{r} s'_i \end{aligned}$$

Then $(s'_1, \cdots, s'_n)$ is a certificate for the SUBSET-SUM instance (S, t). □

COMPLEXITY OF THIS REDUCTION.

In this subsection we discuss the complexity of the reduction we propose in order to ensure it is polynomial. Indeed, while the generated instance (V, P, L, H) presents an amount of nodes exponential in the size n of the set S, the time complexity of the construction is already polynomial.

- a can be written with an amount of bits linear in the number of bits required to encode S

- Step 1 take a time $\Theta(a)$ because describing the d-regular tree T of depth a is $\Theta(log(a) + log(d)) = \Theta(log(a))$, as d is a constant and because we only need to describe the tree (not to construct it explicitely). It takes a time linear in a to attribute the pageRank of the nodes at each depth.
- Step 2 is polynomial in n. Since all the leafs of the tree are identical, the construction of a meta-node w_i is $\Theta(log(s_i))$ (again, we only need to describe it) and computing the pageRank of w_i is just a multiplication.
- Step 3 is linear in n.
- Step 4 has a complexity of $\Theta(d) + \Theta(n) = \Theta(n)$ as d is a constant.
- Step 5 is constant in time.

Then our construction is polynomial in the size of S.

Remark. *If $t > m$, we can trivially conclude that (S, t) admits no certificate.*

This concludes the proof of NP-Hardness of (d, c)-$graphFromPR$.

3.2 NP-COMPLETNESS

As we have seen in the proof of claim 1, given an instance of (d, c)-$graphFromPR$ (V, P, L, H) and a graph $G = (V, E)$, it can be checked in time polynomial in (V, P, L, H) that it is a certificate. Then, (d, c)-$graphFromPR$ is in the complexity class NP. As it is NP-Hard, (d, c)-$graphFromPR$ **is NP-complete**.

The problem (d, c)−graphFromPR is NP-hard. This is why, in the next chapter, we will see an algorithm that solves it in time exponential. However, we will also introduce an efficient approximation algorithm, inspired by the same idea as the reduction from from SUBSETSUM we have seen.

CHAPTER 4

ALGORITHMS

In this chapter, we first introduce a basic principle that is important to understand to approach our reconstruction algorithms. Then, we introduce a simple algorithm that solves exactly the problem (d, c)-$graphFromPR$ in time possibly exponential and finally we present an efficient algorithm that approximately solve the problem.

Claim 2. *In a partially reconstructed graph H' with associated pageRank $\{p_0, ..., p_{n-1}\}$, for which we know the expected pageRank $\{p_0^*, ..., p_{n-1}^*\}$, we can say, for a node, if it is still missing incoming edges and if it is still missing outgoing edges.*

Proof.

- **For ingoing edges**, given node v_i that is not the root, if

$$p_i^* = \frac{1-c}{d} \sum_{j:(v_j,v_i)\in E} p_j^*$$

  v_i has completed its pageRank. Indeed, it remains to complete the pageRank of its predecessors to get its pageRank equal to its expected pageRank. On the other hand, if

$$p_i^* \leq \frac{1-c}{d} \sum_{j:(v_j,v_i)\in E} p_j^*$$

  v_i needs additional ingoing edges. The reasoning is the same for the root node, with the equation $p_0^* = 1 + \frac{1-c}{d} \sum_{j:(v_j,v_0)\in E} p_j^*$, v_i.
- **for outgoing edges**, given a node i, if $i \in L$, it does not need outgoing edges. Else, it needs outgoing edges if and only if it has strictly less than d edges.

□

The following algorithms will use this observations (that does not depend on the current pageRank in the graph, but only on the edges and the expected pageRank to gradually rebuild a graph by completing the in and outgoing edges of each node.

4.1 EXACT ALGORITHM BASED ON BACK-TRACKING

In this section we present an exact algorithm to solve the (d, c)-$graphFromPR$ problem. This algorithm is presented in order to show we can resolve the problem exactly but it does not run in reasonable time. Its presentation may also be useful in order to understand the approximation algorithm we present in section 4.2.

The exact algorithm based on backtracking is presented through algorithms 1 and2.

Idea of the algorithm. Starting with the node with the highest pageRank requiring successors (usually the root), the number of successors for each node not yet processed is increased up to d. Nodes are processed in order of decreasing pageRank and successors can only be chosen if they have sufficiently high pageRank and still have few enough parents to satisfy the pageRank equation. The last choices are cancelled and replaced by others if the reconstruction becomes impossible.

More precisely, during the reconstruction, for each node j which still lacks predecessors, we memorize p'_j the part of its label which is still not attributed to a parent. Thus, if $PRED'_j$ is the set of predecessors already assigned to j,

$$p'_j = p_j - \frac{1}{k} \sum_{q \in PRED'_j} p_q$$

The reconstruction algorithm is based on the following data structures:

- the set *need_pred* of nodes i that lack predecessors, together with the share of their label p'_i that has not yet been assigned to a parent
- the set *need_succ* of nodes which have already been integrated to the connected component of node 0 in the graph and whose successors have not yet been chosen
- the partially reconstructed graph consisted of the edges provided at the beginning and the other ones already chosen

If H includes m outgoing edges for the root, the recursive function d, c-GRAPHSEARCH chooses $d - m$ successors among the possible successors of the root and recursively calls on the previous graph to which we have added a link between the root and these chosen children. If $m = d$, we does not begin with the entry-point but with the node of connected component of the entry-point of highest pageRank which has not yet completed all its outgoing edges. If there are no solutions with this choice of successors for the first node, other choices of $d - m$ possible successors are explored until a solution is found. The procedure is the same for the maximum pageRank successor of the first node and for all the other nodes that are not leafs (that are not in L).

When a graph is partially reconstructed, potential successors of a node i are nodes j such that:

- j still lacks predecessors, in other words $p'_j = p_j - \frac{1}{k} \sum_{q \in PRED'_j} p_q > 0$
- $p'_j > p_i/k$: what remains of j's label to be attributed is sufficient for j to be a successor of i, respecting the definition of pageRank

Algorithm 1 function d, c-EXACTRECONSTRUCTION(V, P, L, H)

1: $need\_pred \leftarrow [v \in V : P^*_{bis}[v] \geq \epsilon]$ ▷ nodes that need ingoing edges
2: $need\_succ \leftarrow [v \in V \setminus L$ that has less than d outgoing edges] ▷ nodes that need outgoing edges
3: $G = H$
4: **return** GraphSearch(P, *need_pred*, *need_succ*, G)

Algorithm 2 function d, c-GRAPHSEARCH(P, *need_pred*, *need_succ*, G)

if *need_succ* = $\emptyset$ **then return** G
end if
$i \leftarrow$ node with maximal pageRank in $need\_pred$
possible_succs = { j such that $(j, p'_j) \in$ *need_pred* $\wedge$ $p'_j > p_i/k$ }
if $|$*possibles_succs*$| < k$ **then return** 'No solution'
else
    while it remains not explorated choices of k elements in *succs_possibles* **do**
        succ = choose k successors among *possible_succs*
        nw_graph = *graph* $\cup$ {$(i \rightarrow j)$ such that $j \in$ succ}
        nw_need_pred = *need_pred* / {(j, p'_j) such that $j \in$ succ} $\cup$ {$(j, p'_j - p_i/k)$ for $j \in succ$ such that $p'_j > p_i/k$}
        nw_need_succ = *need_succ* / {i}
        if $i \notin L$ and i has less than d successors **then**
            nw_need_succ = *nw_need_succ* $\cup\{i\}$
        end if
        s = GraphSearch(P, d, c, *nw_need_pred*, *nw_need_succ*, *nw_graph*)
        if $s \neq$ 'No solution' **then return** s
        end if
    end while
    return 'No solution'
end if

4.2 APPROXIMATION ALGORITHM

Principle of the algorithm. We enumerate each node that still need ingoing edges (that we know thanks to the principle of claim 2 in order of *increasing pageRank* and complete the ingoing edges of each one using a subsetSum approximation algorithm.

This is a probabilistic algorithm using deterministic approximation algorithm. As the output of subsetSum approximation algorithm depends on the order of the input elements, we randomly shuffle the input at each iteration of our probabilistic algorithm. In our implementation that we present in chapter 5, we use an approximation algorithm from Ibarra and Kim 1975.

The data structure structures $need\_pred$ and $need\_succ$ play the same role as in the exact algorithm.

However, there are additional parameters in the approximation algorithm:

- ϵ is the precision used to do the computations. As the subsetSum problem is about integer, we work with integer normalized version of the pageRank: all the pageRank are multiplied by $\lceil \frac{1}{\epsilon} \rceil$ and ceiled. Thus, if $\epsilon = 10^{-6}$, a pageRank measure of 0.0123456789 will be translated in 12345 for the computations.
- subsetSum-ϵ is a parameter of the approximation algorithm SUBSETSUMAPPROX that measures the maximal error in the output of SUBSETSUMAPPROX with respect to the best solution possible
- nb_it is the number of iteration we want our probabilistic reconstruction to run. In the end, it return the best solution found according to the function BEST.

The function BEST correspond to a measure of proximity with the expected pageRank. In our implementation we choose to use the metric *score* presented in chapter 5.

This approximation algorithm seems to be really efficient and precise as we will see in the next chapter,

Algorithm 3 d, c-APPROXRECONSTRUCTION(V, P, L, H, ϵ, subsetSum-ϵ, nb_it)

▷ We will be working with integers for using subsetSum approximation
normFactor $\leftarrow \lceil 1/\epsilon \rceil$
$\epsilon_{bis} \leftarrow \epsilon \times normFactor$ ▷ $\epsilon_{bis} \simeq 1$
$P_{bis} \leftarrow \begin{cases} V & \rightarrow & \mathbb{N} \\ v & \mapsto & \lceil P[v] \times normFactor \rceil \end{cases}$
$G_{best} \leftarrow H$
for $k = 0$ to nbIt **do**
    $G_i \leftarrow H$
    ▷ Computing remaining PageRank to be completed
    $P^*_{bis} : v \mapsto P_{bis}[v] - \sum_{u \in V:(u,v) \in H} \frac{1-c}{d} P_{bis}[u]$
    $need\_pred \leftarrow [v \in V : P^*_{bis}[v] \geq \epsilon]$ ▷ nodes that need ingoing edges
    $need\_succ \leftarrow [v \in V \setminus L$ that has less than d outgoing edges] ▷ nodes that need outgoing edges
    while $need\_pred$ is not empty **do**
        Shuffle $need\_succ$
        $i \leftarrow$ node with minimal pageRank in $need\_pred$
        ▷ We collect possible ingoing edges for node i and eliminate the too big ones
        $candidates \leftarrow \left\{ \frac{1-c}{d} P^*_{bis}[v],\ v \in need\_succ\ s.t.\ \frac{1-c}{d} P^*_{bis}[v] < 2 \times P^*_{bis}[i] \right\}$
        $nodes$ = SUBSETSUMAPPROX(candidates, $P^*_{bis}[i]$, subsetSum-ϵ)
        update H, $need\_pred$, $need\_succ$ with the edges $\{u \rightarrow i,\ u \in nodes\}$
    end while
    $G_{best} \leftarrow$ BEST(G_{best}, G_i)
end for
return G_{best}

where we present our results.

CHAPTER 5

EXPERIMENTS

In this section, we present the results of our approximation algorithm on both real and artificial graphs.

5.1 EVALUATION METRICS

Before presenting the results, let us explain the different measures we use to evaluate the performances of our algorithms.

Definition 5 (Score)**.**

$$score = \frac{1}{\sum_{v_i \in V} p_i^*} \sum_{v_i \in V} p_i^* \left(1 - \frac{|p_i^* - p_i|}{p_i^*}\right)$$

with p_i^* the expected pageRank of node v_i and p_i the pageRank of v_i in the reconstruction graph.

$err = \frac{|p_i^* - p_i|}{p_i^*}$ is a classical measure of error with respect to a target value and $1 - err$ is in contrast a measure of precision. With the *score* we use here, we take in account this notion of precision for each node proportionally to its expected pageRank, this measures its importance.

However, at the beginning of the reconstruction, each node has often pageRank 0 except the node *root* that has pageRank 1, and given the fact that expected pageRank are often approximately close to these values (generally close to 1 for node $root$ and lower than $1/d$ for the other nodes), an algorithm of reconstruction that does not add any edge will often obtain a good score with the precedent metric. In order to evaluate the real performances of our algorithm, we use a normalized version of the score.

Definition 6 (Normalized score)**.**

$$normalized\ score = \frac{score - score_0}{1 - score_0}$$

where $score_0$ is the score it would have reached without adding any edge.

Definition 7 (Error)**.** *Average error:* $error = \frac{1}{|V|} \sum_{v_i \in V} \frac{|p_i^* - p_i|}{p_i^*}$

Definition 8 (Completeness)**.** *We call* completeness *the ratio of the expected number of edges we actually added to the graph. If the completeness equals 1 it means that there is $|V \setminus L| \times d$ edges in the graph, because all the nodes in $V \setminus L$ must have exactly d edges and the nodes in L must have 0 edges.*

Remark. *Because of the construction of our algorithms, the completeness cannot exceed 1, i.e., the number of edges in the reconstruction cannot be greater than the expected one. This is because we do*

the reconstruction with a set of available edges ($need\_succ$[1]) from which we take all the new edges we construct. When the set is empty, the reconstruction stops.

Definition 9 (Similarity)**.** *We call* similarity *the ratio of edges present in the original graph we find in the reconstruction. If we denote by G the original graph and by G' the reconstruction:*

$$similarity = \frac{\{|(u \to v)\ s.t.\ (u \to v) \in G\ \wedge\ (u \to v) \in G'\}|}{|G'|}$$

Definition 10 (weighted similarity)**.** *The weighted similarity weight each edge by the pageRank of the node from witch it begin:*

$$weighted\ similarity = \frac{1}{\sum_{(v_i \to v_j) \in G'} p_i} \sum_{(v_i \to v_j) \in G} p_i 1_{\{(v_i \to v_j) \in G'\}}$$

Definition 11 (Clustering ratio)**.** *Ratio between the clustering coefficient of the reconstruction and the one of the original graph.*

Remark. *The* similarity*, the* weighted similarity *and the* clustering ratio *are measures of the difference between the graph we reconstruct and an "original graph" we know to have exactly the expected pageRank. As shown with the theorem 1, there may exists different graphs that match exactly to a pageRank distribution and a set of edges. This is why we do not expect the similarity and the clustering ratio to equal 1. This metrics are there only as information about the distance between our reconstruction and another graph with close pageRank and a set of common edges.*

5.2 RESULTS ON ARTIFICIAL GRAPHS

5.2.1 GENERATION OF THE ARTIFICIAL GRAPHS

We generated artificial quasi-d-regular graphs to test the performances of our approximation algorithm.

The generation process. The generation process begins from the entry-point and create d successors to it. Then, for each new node, it creates d new edges from it. Each edge goes to an existing node with probability $1/2$ and creates a new node with probability $1/2$. When the current pageRank of a node is lower than a chosen threshold ϵ_{leaf}, it is a leaf and it has no successors. This process terminates because, when it is created, a node has a pageRank lower than $\frac{1}{d}$ times the pageRank of its parent. The limit of the expected average pageRank at each depth is zero.

Here, we present results on a family of artificial graphs generated with this method with approximately 1,500 nodes. These graphs were generated with

- an out-degree $d = 7$,
- a probability of jump to the entry-point $c = 1/5$,
- nodes with pageRank lower than $\epsilon_{leaf} = d \times 0.0001$ chosen as leafs.

On these graphs, in order to get a precision sufficient to do the computations on nodes with the lowest measures of pageRank, the algorithm will always be run with precision $\epsilon = 0.00001$. This has no influence on the complexity of the reconstruction.

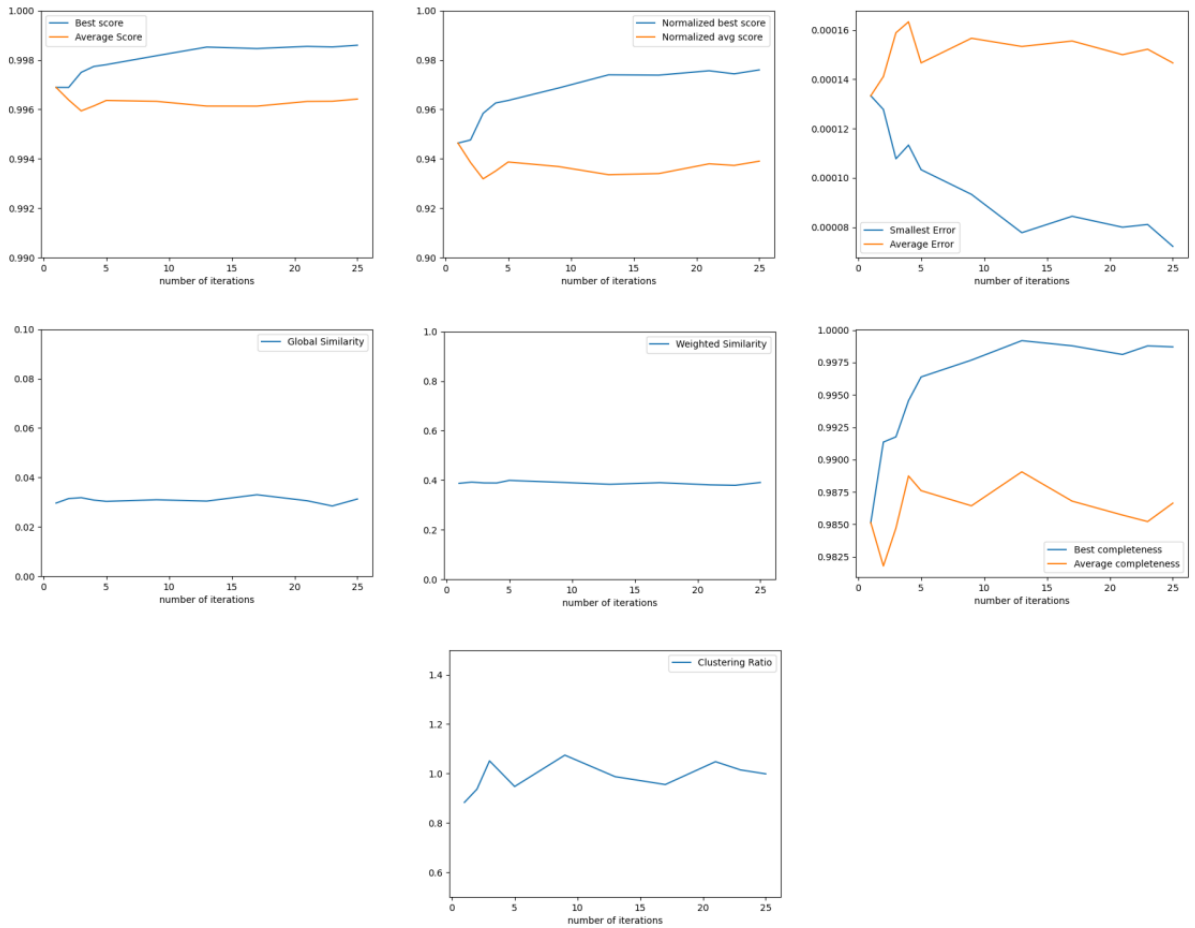

FIGURE 5.1
Evolution of different metrics in function of the number of iteration we choose to run the approximation algorithm with. Average results obtained on 9 different graphs with approximately 1,500 nodes each, subsetSum-ϵ=1/4, 1% of edges in input.

5.2.2 INFLUENCE OF THE NUMBER OF ITERATIONS

Figure 5.1 presents the evolution of the performances of our algorithm in function of the number of times we chose to run the probabilistic reconstruction process. The results have been obtained by averaging the results on 9 graphs with approximately 1,500 nodes each, subsetSum-ϵ=1/4 and each edge of the graph present in input with probability 0.01.

First, note that **the result has a pageRank really close to the expected one, even with only one iteration**: the score measure is greater than 0.996, the normalized score is better than 0.93 and the error is lower than 0.0002.

As expected, increasing the number of iterations makes us more likely to increase our best score. The best score, best normalized score and best completeness have similar shape in that they increase quickly when the number of iteration is between 1 and 5. Then, they increase more slowly with 10 up to 15 iterations and tend to stabilize after 15 iterations. This means that it is useless to do more than 15 iterations. One iteration is already good to get a approximate reconstruction with pageRank pretty close to the expected one. $1 - error$ acts as the precedent metrics.

The average measures of score, normalized score, error and completeness are a bit fluctuating when the

[1]This is actually a set of node, but as we remove a node of the set when it has d edges,so this is equivalent to a set of edges.

number of iterations is low and stabilize when it increases, this is also an expected effect.

The similarity is always close to 3% and the weighted similarity to 40%. The clustering ratio fluctuates between 0.9 and 1.1. These 3 measures do not seem to be influenced by the number of iterations. This means that, how close the pageRank of our reconstruction is to the expected one, it does not influence its similarities to the "original graph".

We said that, despite really close measures of pageRank, the original graph and the reconstruction has only 3% of common edges. How can we explain such a low similarity ? The 40% of weighted similarity gives us a clue: there is really higher similarity among the big edges (edges with high pageRank) than among the small ones. It can be explained by the fact that there are much more edges associated with a small pageRank than edges associated with big ones. Among the edges with small pageRank, many share the same pageRank or really close ones. These edges can be exchanged without changing the pageRank of the nodes of the graph and the result is as close to the expected one as the precedent. This observation will be confirmed with figures 5.3 and 5.5 in section 5.2.4.

Finally, we note that the clustering ratio is close to 1, meaning that that the original graph and the reconstruction have close clustering coefficient.

5.2.3 INFLUENCE OF THE PARAMETER ϵ-SUBSETSUM

Figure 5.2 shows the precedent metrics in function of subsetSum-ϵ. This experiment was made on 9 different graphs with approximately 1,500 nodes each and run with 5 iterations by execution. Each edge is known at the beginning with probability 1%.

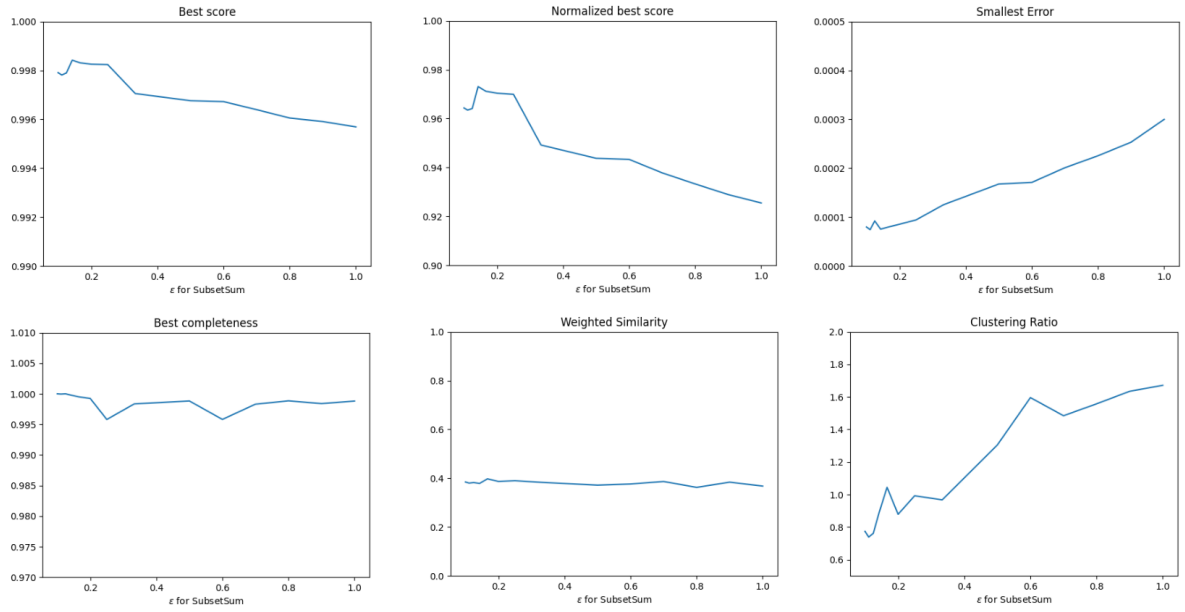

FIGURE 5.2
Evolution of different metrics in function of the value of subsetSum-ϵ we choose to run the approximation algorithm with. Average results obtained on 9 different graphs with approximately 1,500 nodes each, and 5 iterations by execution, 1% of edges in input.

We varied the parameter subsetSum-ϵ between $1/10$ and 1. In the original article Ibarra and Kim 1975 from witch we have taken the algorithm we use in our implementation, the authors prove the error of the result of the algorithm with respect to the optimal solution to be slower than subsetSum-ϵ. In practice, in our implementation we noted errors hundred times lower than this bound. This is why we keep such

"high" values of subsetSum-ϵ.

First note that, apart from the clustering ratio, the results here are really close to the precedent ones, even if the axes of the figures are chosen in order to observe variations: the score is greater than 99.5 %, the normalized score is between 93% and 98%, the error is lower than 0.0003, the completeness is close to 1, the weighed similarity is close to 40% and the similarity is close to 3%.

The general shapes of the *score*, *normalized score* and *error* we obtain show that, as expected, the higher subsetSum-ϵ is, the less precise is the reconstruction.

However, we obtain surprising results for subsetSum-ϵ lower than $1/7$: higher precision on the approximation seems to imply less at the end for the reconstruction. A possible explanation can be that it is noise on the data we work with.

On another hand, the completeness, the similarity (not shown here but really similar to the one presented in 5.1) and the weighted similarity do not seem to be influenced by subsetSum-ϵ. Concerning the completeness it means that the fact of succeeding in constructing the number of edges we wanted is not really impacted by subsetSum-ϵ, that is the precision of the choice of the edges.

In contrast, the clustering ratio moves away from 1, when we decrease subsetSum-ϵ. Good precision seems to guarantee a clustering ratio close to the one of the original graph.

5.2.4 PERFORMANCES ACCORDING TO THE SIZE OF THE GRAPH AND THE NUMBER OF KNOWN EDGES

In the table 5.1 and the figures 5.4 and 5.5, we compare the performances of our approximation algorithm on graphs with approximately 1,500 nodes and 10,000 nodes. As the graphs with 1,500 nodes, the graphs with 10,000 nodes are generated with the same process as described in section 5.2.1 with parameters $d = 10, c = 1/5, eps_{leafs} = 0.00001$. For this graphs, all the computations are made with precision $\epsilon = 0.000001$.

We also did a reconstruction on a graph with approximately 100.000 with one iteration, that ran in 70 minutes and obtain a score of 99.8 %.

These results and those presented in table 5.1 show that our approximation algorithm is efficient enough to face the reconstruction of large ego-graphs.

% of known edges	Graphs 1500	Graphs 10,000
0	17.397s	10m10.609s
0.001	17.374s	10m1.713s
0.01	17.996s	10m15.792s
0.1	16.073s	8m53.863s
0.5	9.154s	8m53.863s
0.9	2.619s	0m45.815s
1	2.173s	0m21.937s

TABLE 5.1
Time of 9 executions with 1 iteration each for graphs with 1500 nodes approximately and graphs with 10,000 nodes approximately, ϵ=1/4

Figure 5.3 shows the influence of the depth on the similarity and the population. Each color figures a proportion of the total number of expected edges that is given in input. For example, the orange bar has been obtained with a reconstruction process such that each edge of the original graph had probability 0.001 to be present in the input set of edges given at the beginning of the reconstruction. The x-axis of

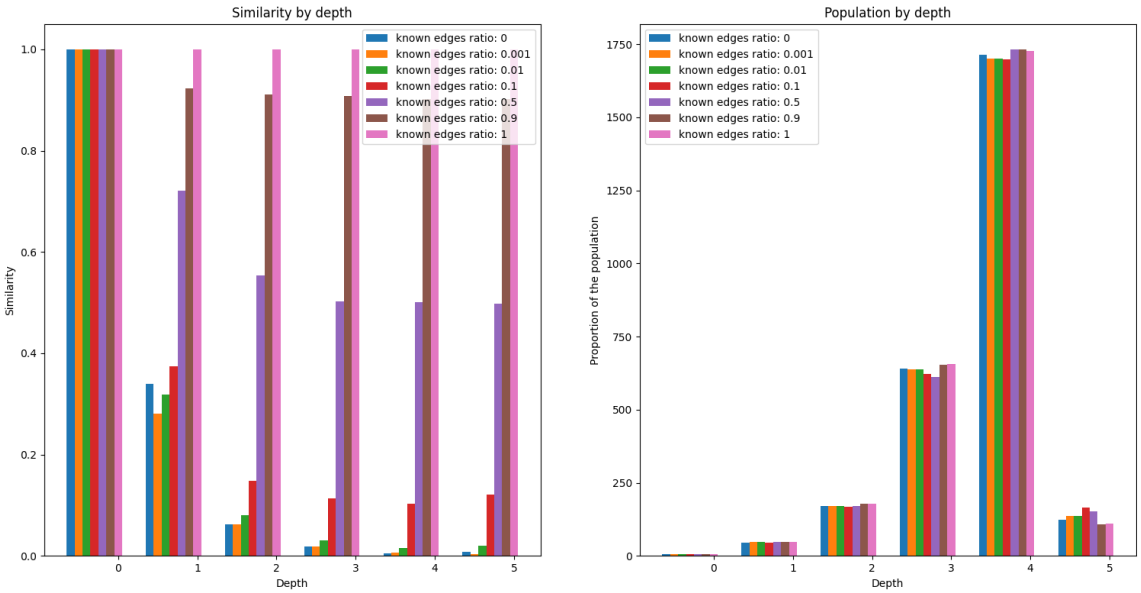

FIGURE 5.3
Similarity and Population by the depth in function of different proportions of the total number of expected edges given in input of the reconstruction. Average results obtained on 9 different graphs with approximately 1,500 nodes each, ϵ=1/4 and 5 iterations by execution.

the figure corresponds to the depth(3) of the edges. The first figure shows the similarity at each depth in function of the proportion of known edges at the beginning and the second figure shows the population of edges at each depth.

Note that the direct successors of the entry-point are always the same in the reconstruction and in the original graph, whatever the proportion of known edges. Then, the similarity declines exponentially with the depth and this is true for all the proportions of known edges at the beginning. This confirm the assumption made in section 5.2.2 about the difference between the similarity and the weighted similarity: the deeper the edges are, the more numerous they are (this is what we see with population) and the more likely they are to be exchanged without changing the pageRank of the concerned nodes. The last depth is an exception to the exponential increase of population because there have predecessors with pageRanks close to be the ones of a leaf. While being less numerous, these edges have a pageRank so low that they are even more likely to be exchanged without consequences than the other ones.

In figure 5.4 we focus on the different metrics previously studied on the graphs with 1,500 and 10,000 nodes according to the proportion of the total number of edges given in input. Whatever the number of known edges, we note that the *score*, the *normalized score*, the *completeness* and the *error* are better on the graphs with 10,000 nodes: the algorithms seems to be better at reaching its goals for a larger graph. In contrast, the measures of similarity that are the *similarity*, the *weighted similarity* and the *clustering ratio* are higher for the graphs with 1,500 nodes. Indeed, as the clustering ratio is really close to 1 for the graphs with 1,500 nodes, it reaches 1.8 with the graphs with 10,000 nodes. Then, when the graph is bigger, the reconstruction is closer to the expected pageRank but farther away from the original graph in term of common edges and clustering coefficient.

This effect of lower similarity for the graph with 10,000 nodes is also shown by figure 5.5, that shows that a higher proportion of the edges have high depth, what we identify as a cause of low similarity.

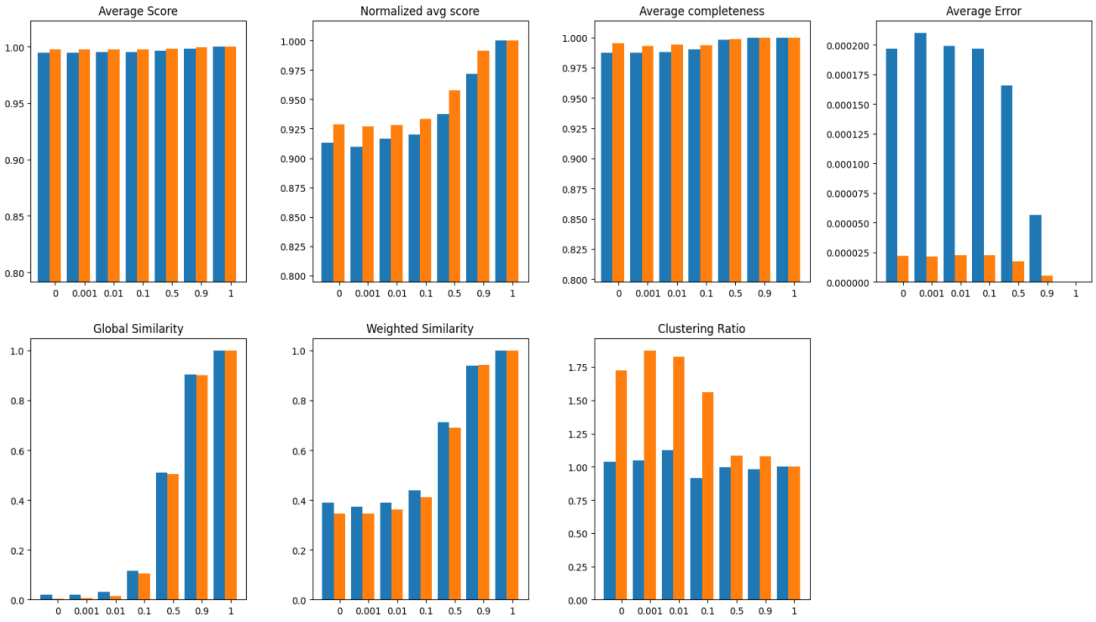

FIGURE 5.4
Comparison of different metrics on graphs with approximately 1,500 nodes and 10,000 nodes each in function of the proportion of edges in input. Average results obtained on 9 different graphs of each type with subsetSum-ϵ=1/4 and 5 iterations by execution.

5.3 RESULTS ON REALS GRAPHS

We tested our approximation algorithm on an ego-graph from Youtube with out-degree 19 and maximal depth 5. This graph has $11,820$ nodes and $32,413$ edges.

We run the algorithm on this graph with subsetSum-$\epsilon = 1/4$ and 5 iterations. The results are presented in tables 5.3 and 5.2. and are really similar to the results on artificial graphs.

best score	0.9990005
avg score	0.9982323
normalized best score	0.9533408
normalized avg score	0.9174768
best completeness	0.9761325
avg completeness	0.8984882
best precision	0.9999896
avg precision	0.9999831
common edges	0.0103315
pond common edges	0.3266173
clustering ratio	0.1995812

TABLE 5.2

depth	population	similarity
0	17	1.0
1	322	0.1273292
2	3207	0.0349236
3	20201	0.0180189
4	47889	0.0043643

TABLE 5.3

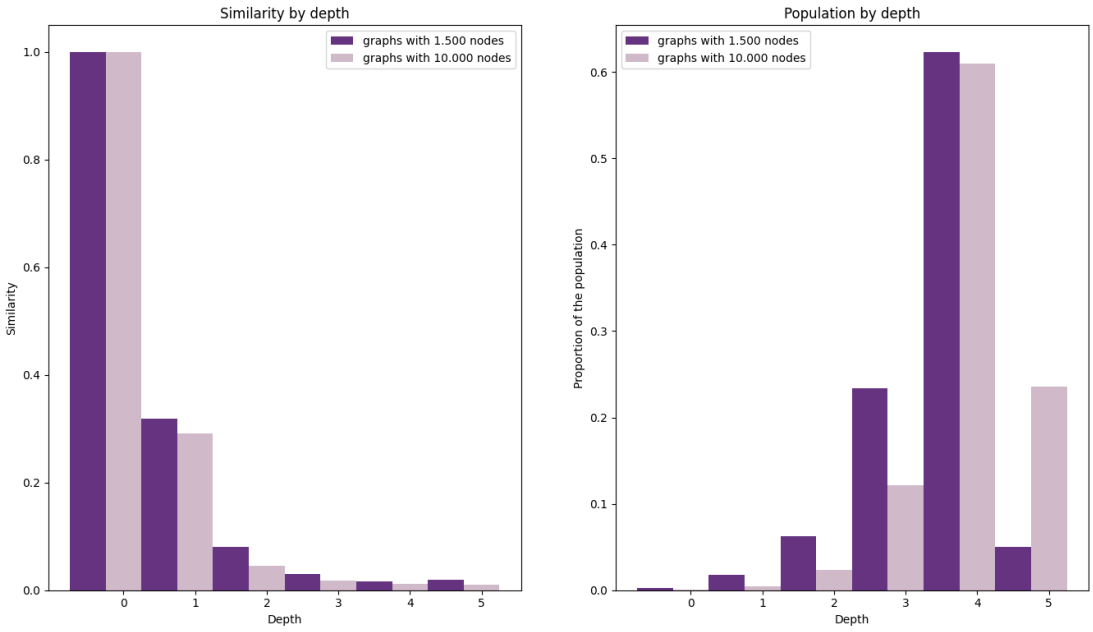

FIGURE 5.5

Similarity according to the proportion of known edges at each depth and proportion of the population at each depth for graphs with approximately 1500 nodes (dark colors) and 10,000 nodes (light colors). In this figure, the columns on the x-axis corresponds to the depth, i.e. the length of the shortest path from the entry-point. The colors figure the ratio of known edges. The data concerning the graphs 1500 is shown in dark color and the one concerning the graphs 10,000 is shown in light color.

CHAPTER 6

LIMITATIONS AND FUTURE WORK

Our problem statement is not close to our motivation case of *number of views* for several reasons:

- The users acts no as random surfer. The mechanisms in choosing recommendation are more complex. However, the majority of the traffic on youtube is actually generated by the recommendations, according to youtube it-self (Goodrow n.d.).
- We supposed the graph of the recommendation system to be static, that is not the case in the majority of the modern recommendation system. However, we can do two snapshots of the number of views in the graph and subtract the first number of views to the second in order to observe the views generated between the two snapshots. The snapshots must be close enough in time so that the system has not evolved too much in the meantime, but far enough in time to get a number of view sufficient to approximate the pageRank. With this solution, we suppose that the number views increase really faster than the recommendation system evolves.
- The main problem we have with our problem is that the recommendation graph is often personalized according to each user. Then, the number of views we observe does not reflect one graph but a myriad of.
- The ego-graph approach suppose a common home page for all the users.

Nevertheless, this work was a first step to answer the two large and complex questions: *'What can we learn about a recommendation system from number of views ?'* and *'What information does the pageRank bring with itself ?'*

Finally, even if it proves its efficiency on the experiments, our approximation algorithm already need to get its approximation ratio bounded.

CHAPTER 7

CONCLUSION

During this semester project, we introduce the introduce the problem of reconstruction of a directed quasi-d-regular graph from its pageRank and a set of edges. This problem was motivated by the concrete case of recovering a recommendation system from the number of views on each content. We prove that the problem to be NP-Complete. We then propose an algorithm based on back-tracking that solves the problem exactly but in time exponential and an efficient algorithm to approximate a solution, based on known algorithms of approximation for SUBSET-SUM problem. We presented the performances of our approximation algorithm on artificial and real graphs. This algorithm is efficient in practice but we already need a theoretical bound for its precision.

BIBLIOGRAPHY

Page, Lawrence et al. (1999). *The PageRank citation ranking: Bringing order to the web.* Tech. rep. Stanford InfoLab.

Mathieu, Claire and Hang Zhou (2021). 'A Simple Algorithm for Graph Reconstruction'. In: *arXiv preprint arXiv:2112.04549*.

Kannan, Sampath, Claire Mathieu and Hang Zhou (2018). 'Graph reconstruction and verification'. In: *ACM Transactions on Algorithms (TALG)* 14.4, pp. 1–30.

Hoskins, Jeremy et al. (2018). 'Inferring networks from random walk-based node similarities'. In: *Advances in Neural Information Processing Systems* 31.

He, Zhe et al. (2014). 'Network reconstruction by the stationary distribution of random walk process'. In: *arXiv preprint arXiv:1410.4120*.

Fontoura Costa, Luciano da and Gonzalo Travieso (2007). 'Exploring complex networks through random walks'. In: *Physical Review E* 75.1, p. 016102.

Wittmann, Dominik M et al. (2009). 'Reconstruction of graphs based on random walks'. In: *Theoretical Computer Science* 410.38-40, pp. 3826–3838.

Cormen, Thomas H. et al. (2007). *Introduction to algorithms*. MIT Press.

Ibarra, Oscar H and Chul E Kim (1975). 'Fast approximation algorithms for the knapsack and sum of subset problems'. In: *Journal of the ACM (JACM)* 22.4, pp. 463–468.

Goodrow, Cristos (n.d.). *Inside Youtube - On Youtube recommendation system https://blog.youtube/inside-youtube/on-youtubes-recommendation-system/*.