Theoretical Computer Science exam 4BIM

Sergio Peignier and Théotime Grohens

11th January 2020

1 Exercise 1

- Draw graphs that have:
  - A large average clustering coefficient and a large average node distance.
  - A low average clustering coefficient and a low average node distance.
  - A low average clustering coefficient and a large average node distance.
  - A large average clustering coefficient and a low average node distance.
- Show that a tree, that has a node with degree d, has at least d leaves.

2 Exercise 2

This exercise aims at implementing a label propagation algorithm and apply it to a real world example. The label propagation algorithm is a semi-supervised algorithm that assigns a label (class) to unlabeled nodes from a network, by propagating the labels of a small subset of labeled nodes. Let us describe the main steps of this algorithm.

Label propagation algorithm

- Let $G = \langle V, E \rangle$ be a graph with set of nodes V and set of edges E. Let $U \subset V$ a set of labeled nodes, s.t. for each node $u \in U$, c_u is the label of u.
- Initialize the labels c_v of each node $v \in V$ in the network (unlabeled nodes are marked as "unknown").
- Let $X = (v \in V \mid c_v = "unkwown")$ the list of unlabeled nodes, arranged in a random order.
- For each node $x \in X$, let $N_x = \{n \mid (n, x) \in E \vee (x, n) \in E\}$ be the set of neighbors of x.

- For each label c, count the frequency $f(c, x)$ of labels among neighbors N_x of node x, $f(c, x) = |\{n \in N_x \mid c_n = c \wedge c_n \neq "unkwown"\}|$
- The label of node x is set as $c_x = argmax_c f(c, x)$, i.e., node x takes as label the majority label among its neighbors. If two or more labels have the same frequency, choose one label randomly.
- The algorithm stops when labels of unclassified nodes do not change.

Application We will apply this algorithm to label a network of Twitter users (journalists and politicians). In this network, there is a link between users A and B, if user A re-tweets user B. The weight of edge (A, B) is equal to number of times A re-tweets B. Only a subset of nodes are labeled according to their political affiliation (0 or 1).

File "network.csv" contains the list of edges. Each row has three elements, the first one represents the user that retweets (source node), and the second one the user that is retweeted (target node), and the last column the number of retweets (weight).

File "classes.txt" has two columns, for each row the first element represents a user (node) and the second one represents its label. Notice that this file does not provide a label for all nodes.

Questions

1 Program the label propagation algorithm. Your function should take as an inputs a networkx graph, and a pandas Series or a dictionary containing the labels (values) of some nodes (keys).

2 Load the classes (pandas Series or dictionary).

3 Load the list of edges as a pandas DataFrame, and convert the DataFrame into a networkx graph.

4 Compute the sum of retweets for each node, sort the nodes according to the sum of retweets and print the 10 users more retweeted.

4 Apply the label propagation algorithm to the real world dataset.

5 Compute the number of nodes belonging to each political parties (number of nodes sharing the same label).

6 Plot the network (the colors of the nodes should represent their labels).

7 Describe the network briefly.

3 Exercise 3: the knapsack problem

3.1 Description of the problem

The knapsack problem is a famous NP-complete computer science problem, illustrated in figure 1. Given a bag of capacity M and a list of N items with weights w_i and values x_i, the goal of the problem is to choose items to put in the bag in a way that maximises the total value of the items in the bag, while their total weight remains less than the capacity of the bag.

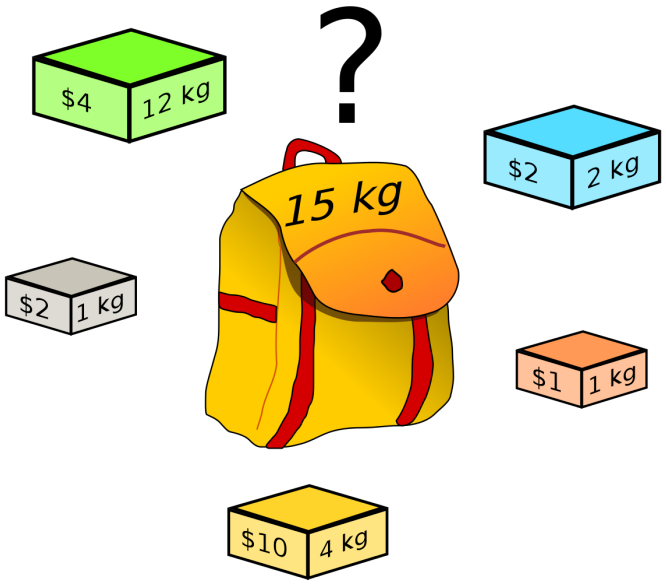

Figure 1: An illustration of the knapsack problem (from Wikipedia).

In this exercice, we will study a version of the problem in which each item can be used at most P times. The constraints of the problem are therefore to find a set of numbers k_i such that:

$$\begin{cases} \forall i \leq N, 0 \leq k_i \leq P \\ \sum_{i=1}^{N} k_i w_i \leq M \\ \sum_{i=1}^{N} k_i x_i \text{ is the largest possible} \end{cases}$$

3.2 Theoretical complexity class

Questions

1. Write the knapsack problem as an abstract optimisation problem, and give the corresponding instance set and solution set.

2. Give an equivalent abstract decision problem, with its associated instance set and solution set.

3. Give a corresponding concrete decision problem, by giving a binary encoding of instances in the instance set.

4. Give a polynomial-time algorithm (either in Python or in **detailed** pseudocode) that checks a certificate for the decision problem. This proves that the knapsack problem is NP-easy (we won't prove that it is NP-hard in this exercise).

3.3 Solving the problem

Questions

1. Give a dynamic programming algorithm that solves the knapsack problem (either in Python or in pseudocode).

2. What is its time complexity? Give a formal proof.

3. Does this algorithm really run in polynomial time? If not, prove why.