DyGraph

Documentation

1.0

Table of Contents

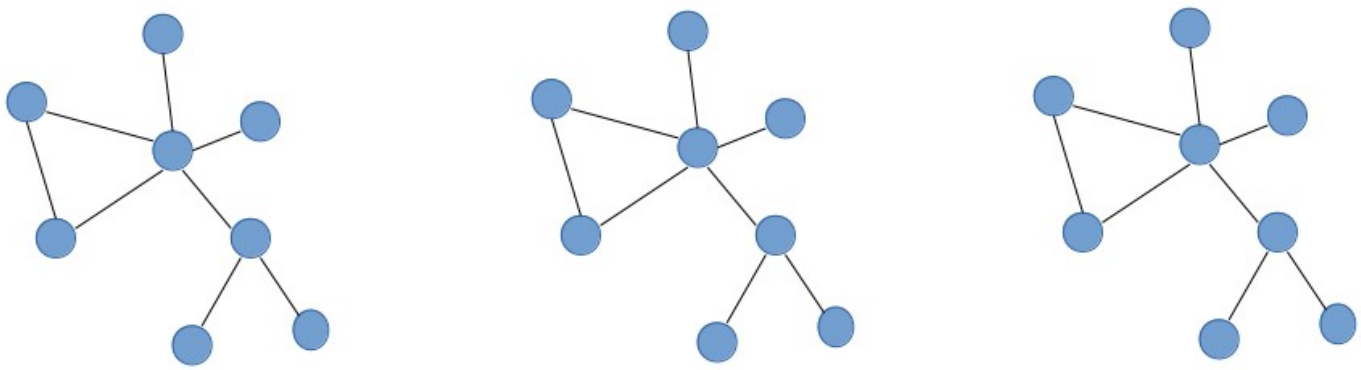
Introduction

This documentation intends to explain the rational and working principles of DyGraph, a dynamic graph generator that gives special emphasis to the degree distribution of dynamic graphs. The document also gives descriptions of the different parameters that are used in the tool by the various functions. It is organized in a way that there is a brief description of dynamic vs static graphs, followed by some of the properties that are used in the code, then a description of the functions in the tool. Finally, there is a section that shows some of the degree distributions of graphs generated with the tool as compared to the distribution of collected datasets.

What are graphs?

A graph is defined as a data structure representing a set of objects in which some of the objects are related in some way. The particular objects are represented as nodes while their relationship is represented with an edge. There are different classifications based on the type of classification that is used. In this document we cover the classification of graphs based on time.

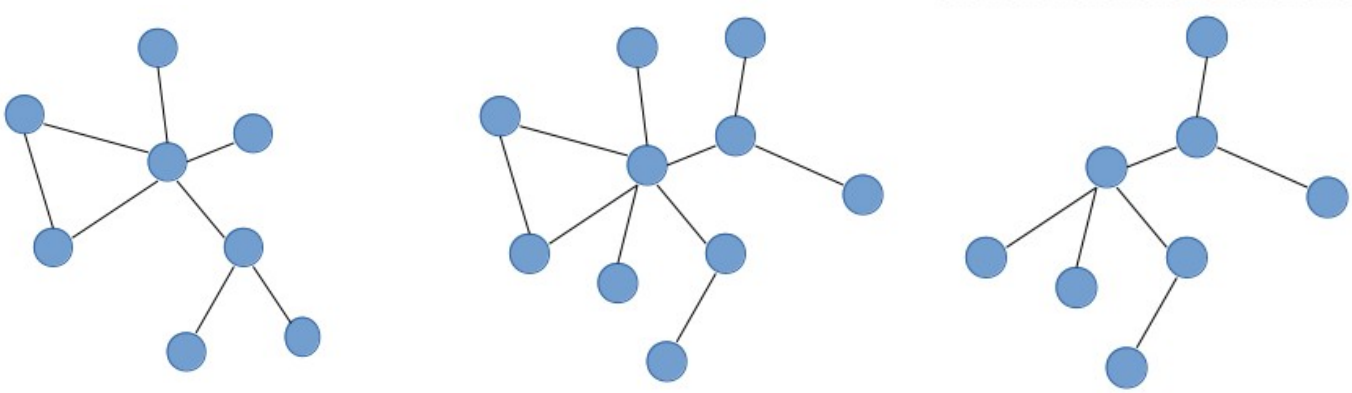
Static graphs are graphs that stay the same indefinitely. That means their vertices and edges do not change with time.



t=0 t=1 t=100

Figures 1.1,1.2, and 1.3. snapshots of a static graph at different time instances.

Dynamic graphs are graphs that change with time. In dynamic graphs, there is addition and removal of vertices and edges, which in turn effects change in the different properties of the graph.



t=0 t=1 t=100

Figures 2.1, 2.2, and 2.3. snapshots of a dynamic graph at different time instances.

Definitions

1. Degree: refers to the number of connections a node has.
2. Frequency: the number of nodes with a specific degree.
3. Degree distribution: a distribution showing degree vs frequency.
4. Degree sequence: is a monotonic non-increasing sequence of the degrees of the graphs.
5. Timestamp: is the time at which the snapshot is built.

Objectives behind the development of DyGraph

This project started with two major objectives:

- Having the ability to change the size of a graph: i.e., add and remove edges as time progresses.
- Replicate some of the properties of dynamic graphs that are observed in datasets, with special focus on the degree distribution of the graphs. Random and power law degree distributions are covered in DyGraph.

Features

Build a graph with power law distribution

- the build function will build a graph that has a power law distribution.
- the degree distribution is approximated using power law approximation. First the scale of the degree distribution is converted to log scale and linear regression is used in the log scale to find the approximation slopes.

  Freq=(10^B0)*Degree^B1

  log(Freq)=B0+B1*log(Degree)

  Parameters:
  - localMaximum: the maximum frequency in a given bin. Typical value for this parameter ranges from 10-1. This is due to the fact that the violation of strict power law occurs for values of frequencies less than 10.
  - Bin: a batch of degree and the associated frequencies in the degree distribution. For this parameter, the value is accommodated in the code, with a bin range of 10 for maximum degree less than 100 and a bin of range one-hundred for maximum degree greater than 100.
  - bin size: the number of degrees that are in a given bin. In a given range of degrees, a normal value of bin size depends on the index of the bin. For the first few bins, there is a large number of degrees in a given range as is the character of a AUC of an exponential decay. Typically, the first few bins have a size ranging from the one-hundred to fifty. As the curve smooths out towards the tail, the bins size decreases drastically, to a minimum value of one.
  - B0: is the slope used to build the power law model of the degree distribution. Since this is a slop for the additive component of the equation, it has a positive value with a typical range from 6.0-0.8 depending on the size of the graph.
  - B1: the slope used in to build the power law approximation of the degree distribution. This slope has a negative value as it is responsible for the concave curve of the power law distribution. Typical values range from -3.0 to -0.8.
  - maxdeg/ maxFreq: the maximum degree/frequency in the degree distribution. The value of these two parameters depends on the graph in question. Max

Frequency could go as high as tens of thousands while maximum degree usually stays in thousands of range.

The power law approximation is observed to show a strict power law only until a certain degree is reached. Usually, the degree up to which the degree distribution follows the power law is around the degree where the frequency is ten, i.e the distribution doesn't follow a power law distribution when the degree has a frequency less than ten. The remaining part of the distribution shows what looks like a discritized area-under-the-curve of an exponential decay. To accommodate for this, the degree distribution is divided into bins of certain size and the bin information is recorded to replicate the effects on the generated graphs. The figures that follow illustrate how the bins work.

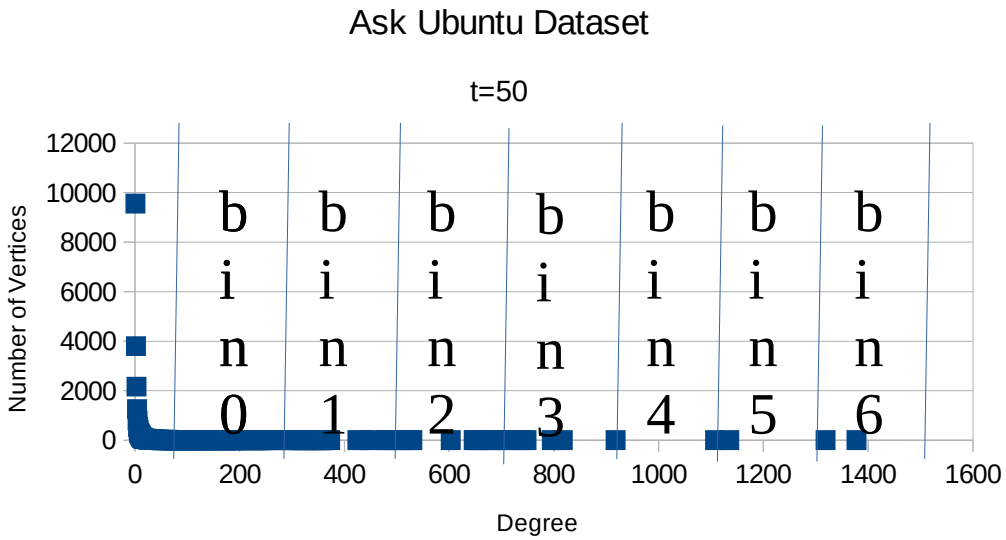


Figure 3. Classifying the degree distribution into 6 bins with an interval of 200.

In Figure 3, it is seen the 0th bin is from degree 100 to 300. In between this range, there will be X number of degrees (not all degrees in the range exist in the graph). The bin size refers to the number of degrees that are between the range 100 and 300.

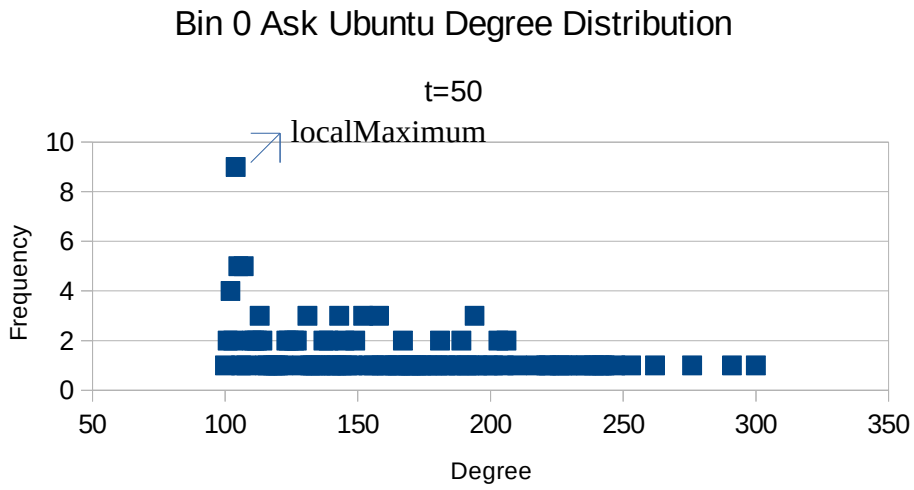


Figure 4. A closer look at the 0th bin of the distribution in Figure 3.

Looking at the 0^{th} bin more closely, we see the local maximum for the bin, which is the maximum frequency in the bin is 9. The tool will take this information and follow the following steps to build the degree distribution of the graph.

1. Build an approximate of a power law distribution using the maxDeg, B0 and B1.
2. When a degree with frequency less than ten is reached, go into the bin information and for each bin
    1. Select (binSize) number of degrees from the range of the degree
    2. For each of the degrees randomly select a frequency, with the local maximum as the ceiling of selection.
    3. Add the degree-frequency pair into the distribution.

Add Functions

Add Vertices: the tool allows users to add vertices to the graph at each timestamp.
-add vertices by assigning them IDs in the data structure.

Add Edges: the tool allows users to add edges to the graph at each timestamp.

1. Add random edges
   -adds random edges to the graph by selecting two random vertices and adding an edge between them if the edge does not already exist. If it does, the tool will attempt to select another pair of vertices and add edges between them.
2. add edges that sustain the power law
   -add edges following the power law distribution of the graph. Ensures the sustaining of the degree distribution as time progresses.
   -follows similar steps as the build function, but in this case, the following additional steps are taken:
    1. Convert the degree distribution of the previous timestamp to a degree sequence.
    2. Convert the degree distribution of the current timestamp to a degree sequence.
    3. Take the difference of the two degree sequences and shuffle it randomly.
    4. Add edges by selecting two vertices at a time from the degree sequence.

Remove Functions

Remove vertices: the tool allows you to remove vertices from the graph randomly as well as to remove vertices that do not have a connection.

Remove Edges: currently the tool allows the removal of edges randomly.

Get Information about the graph

-the GET command will run an approximation function on the distribution of the grph found in the analysis folder. It will generate a file called graph_distribution_info.txt which is a file that shows the values of B0, B1, localMaximum, bin information and so forth. It also generates a commands.txt file in the root directory, which can be used to build a graph by using the build from file functionality of the tool.

-the PRINT command prints the edge list and change log of the current timestamp.

How to use the tool

Go into the root directory and run the command “make debug”. This will create a binary called graph_tool_debug. Run the corresponding binary.
The tool has three options:

1. Console
2. File
3. Build from dataset

Building from console

When you choose the console option, the tool will take you into a console that allows you to enter the commands that are described in the read me file.
Note: the first command used in the console should be CREATE, as it calls the graph constructor and failure to do so might result in a seg fault.

Build from File

in the second option, the tool will read commands from the commands.txt file in the root directory. An example of a command file is present in the git repo.
Note: functions that take parameters such as BUILD and ADD EDGE POWERLAW, take the parameters in line when they are called in the commands.txt file.

Build from dataset

In this case, the tool will read files that are found in the real folder, which contain elists of graphs that are collected, and builds elist, clog and dist files in the output and analysis folders. Those files can letter be used to extract information about dataset graphs using the get command.

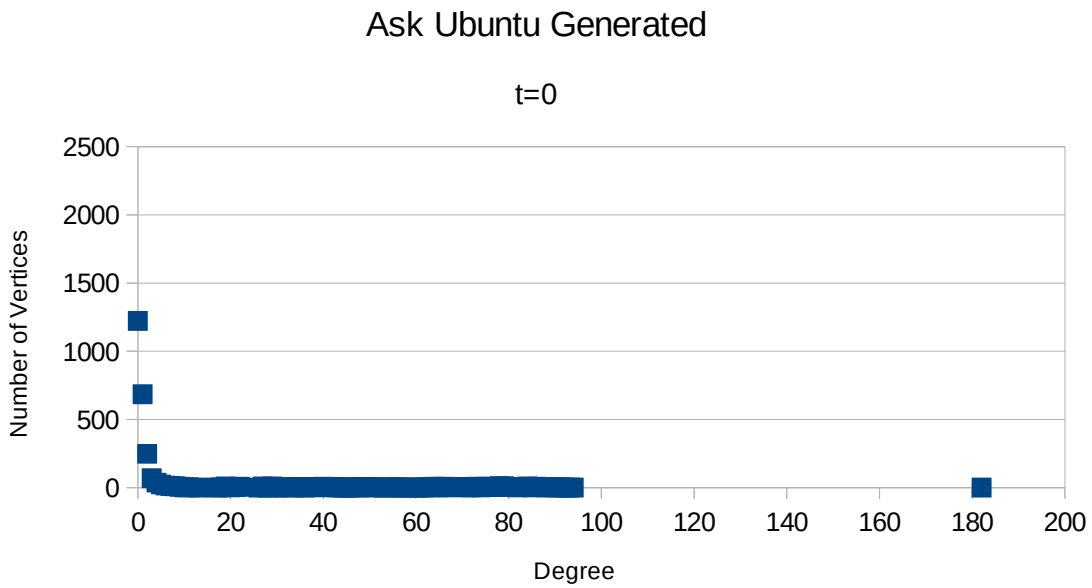


Figure 5.2 Degree Distribution of generated Ask Ubuntu Dataset at time 0 in linear scale.

Logarithmic scale

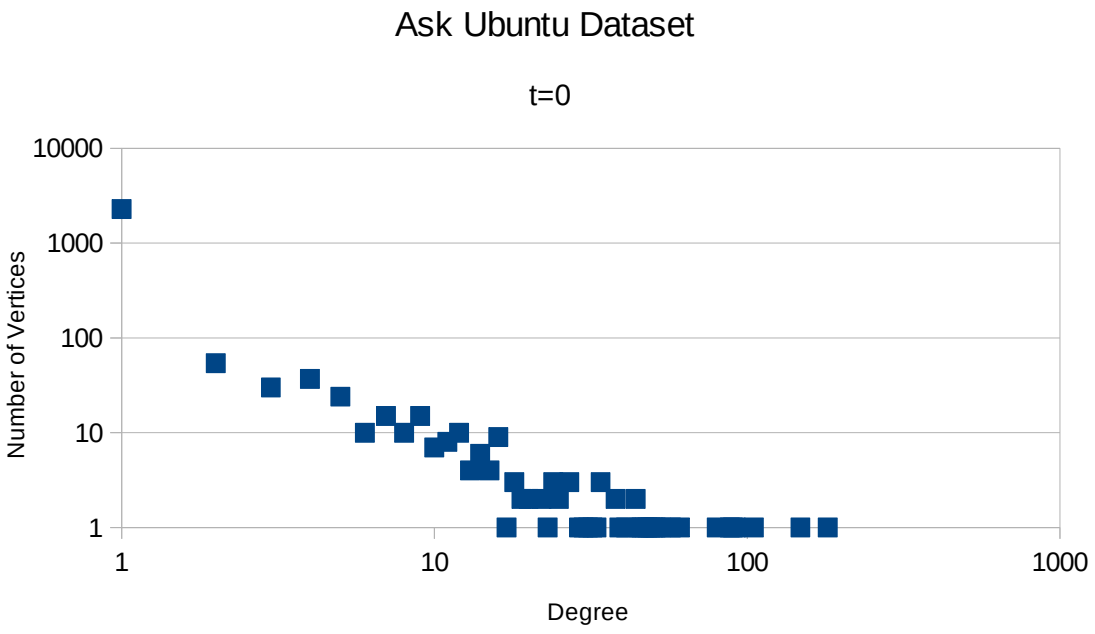


Figure 5.3 Degree Distribution of collected Ask Ubuntu Dataset at time 0 in logarithmic scale.

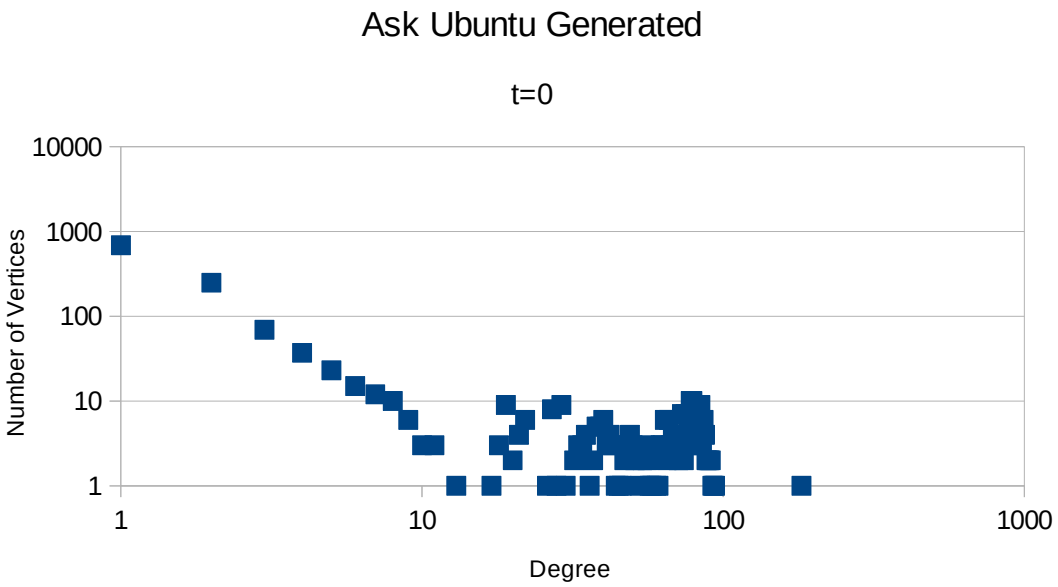


Figure 5.4 Degree Distribution of generated Ask Ubuntu Dataset at time 0 in logarithmic scale.

Time 50

Linear Scale

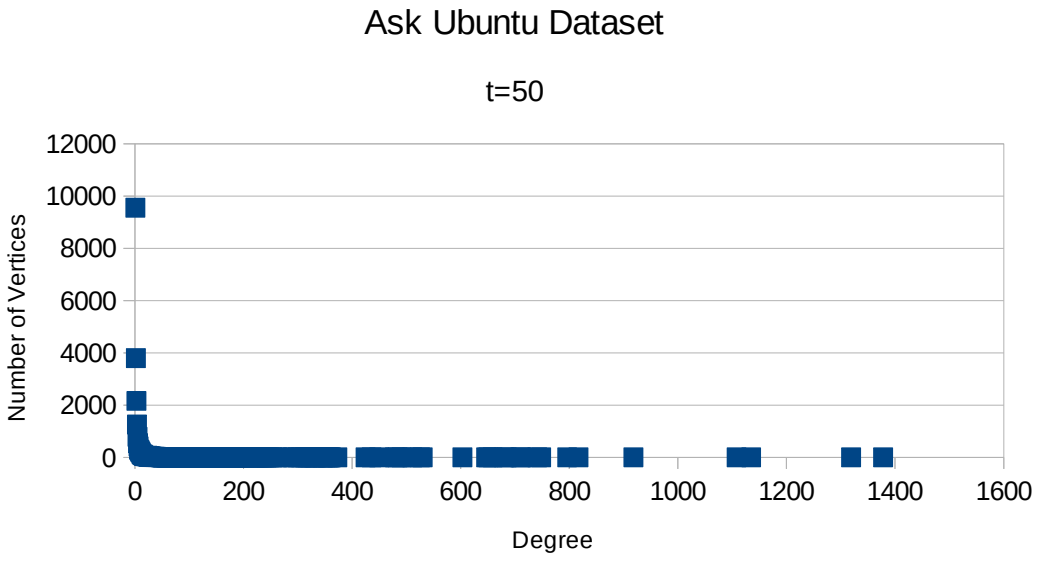


Figure 6.1 Degree Distribution of collected Ask Ubuntu Dataset at time 50 in linear scale.

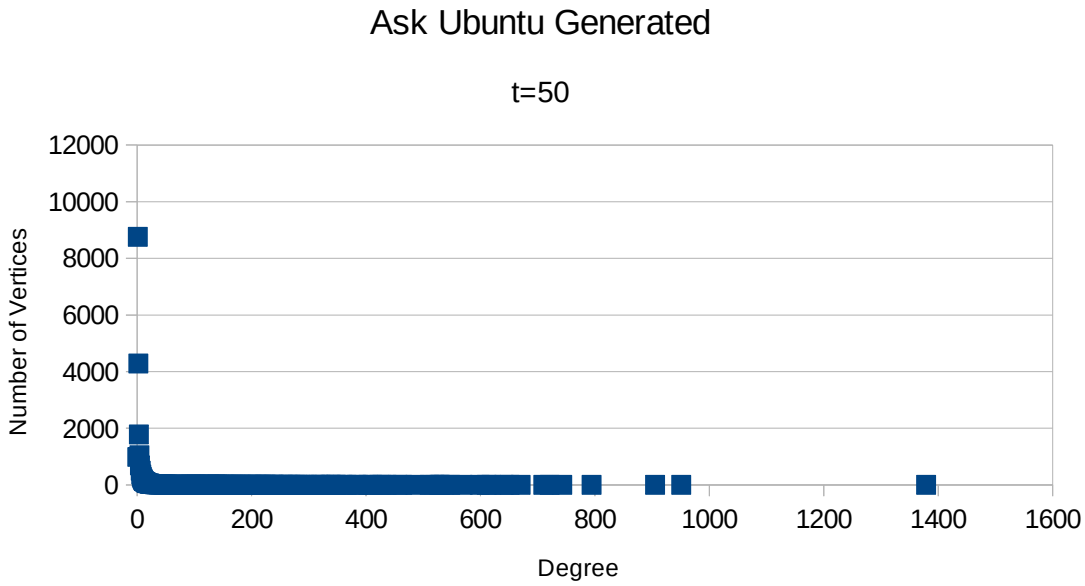


Figure 6.2 Degree Distribution of generated Ask Ubuntu Dataset at time 50 in linear scale.

Logarithmic Scale

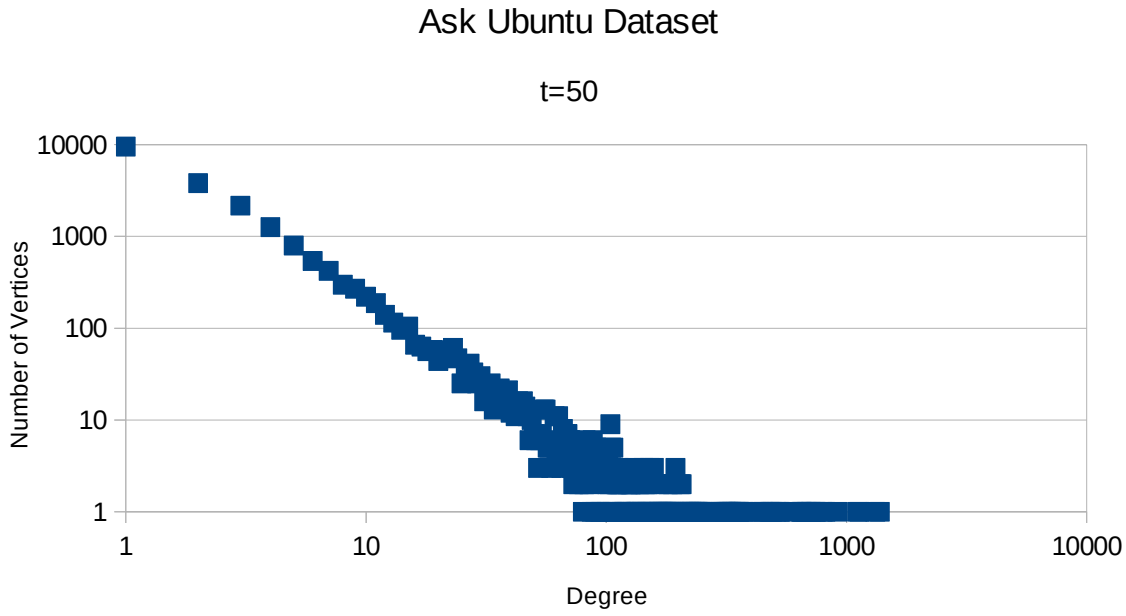


Figure 6.3 Degree Distribution of collected Ask-Ubuntu Dataset at time 50 in logarithmic scale.

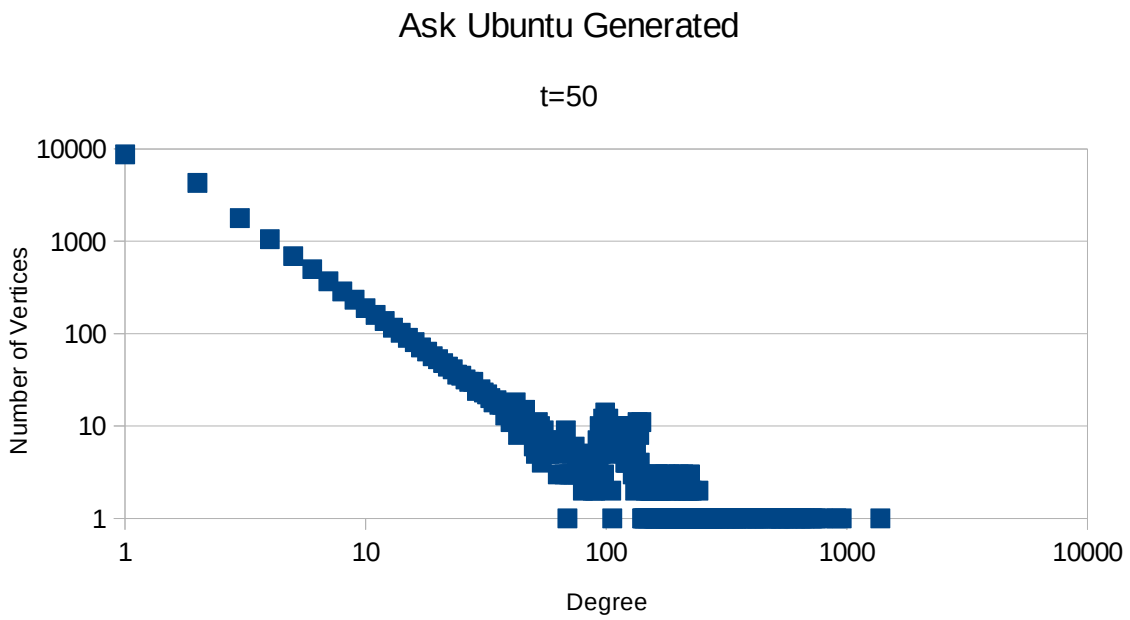


Figure 6.4 Degree Distribution of generated Ask-Ubuntu Dataset at time 50 in logarithmic scale.

Time 100

Linear Scale

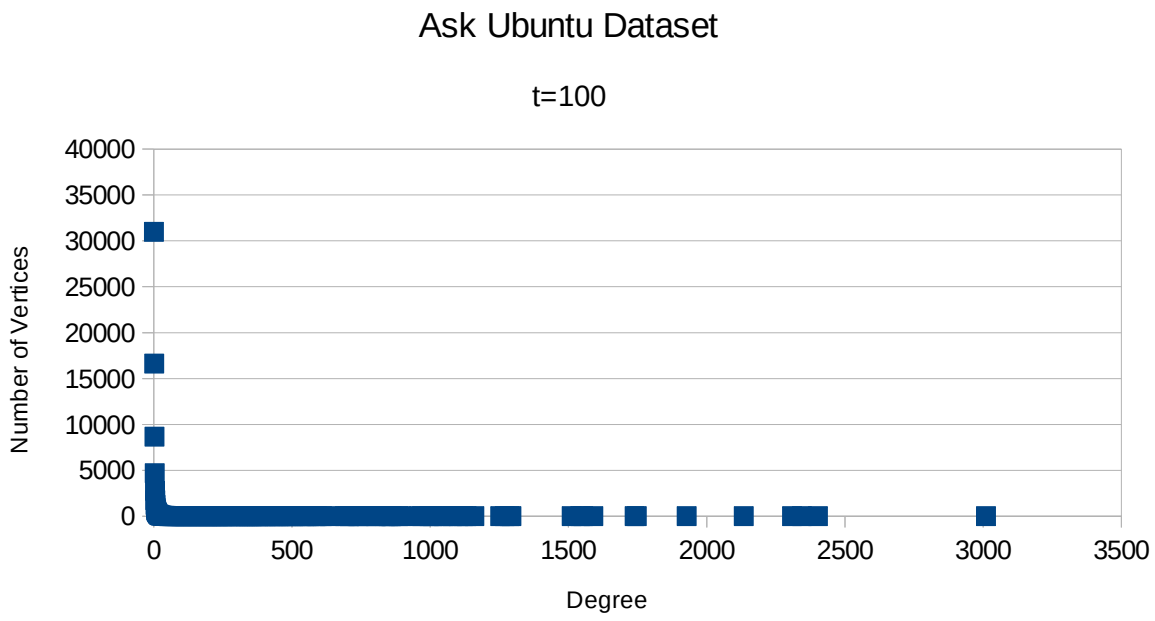


Figure 7.1 Degree Distribution of collected Ask-Ubuntu Dataset at time 100 in linear scale.

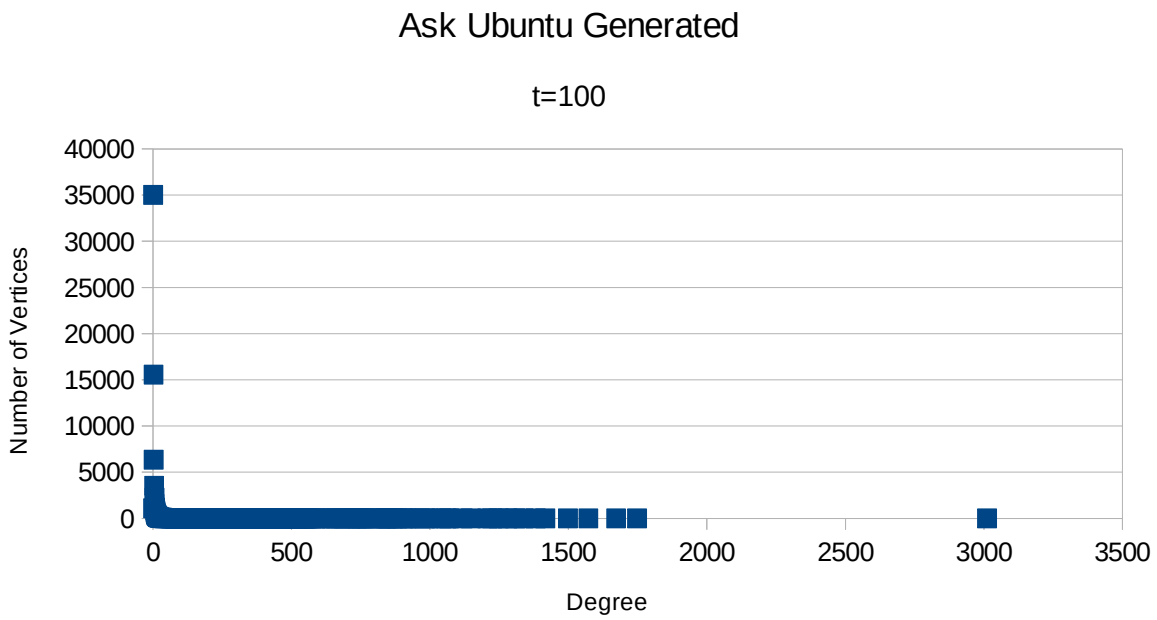


Figure 7.2 Degree Distribution of generated Ask-Ubuntu Dataset at time 100 in linear scale.

Logarithmic Scale

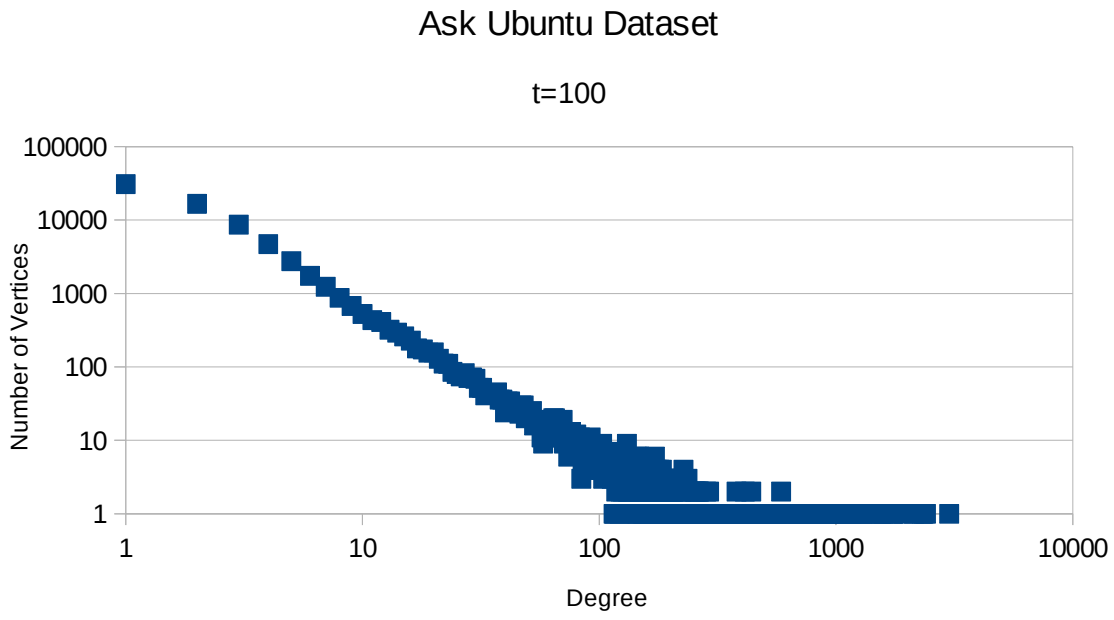


Figure 7.3 Degree Distribution of collected Ask-Ubuntu Dataset at time 100 in logarithmic scale.

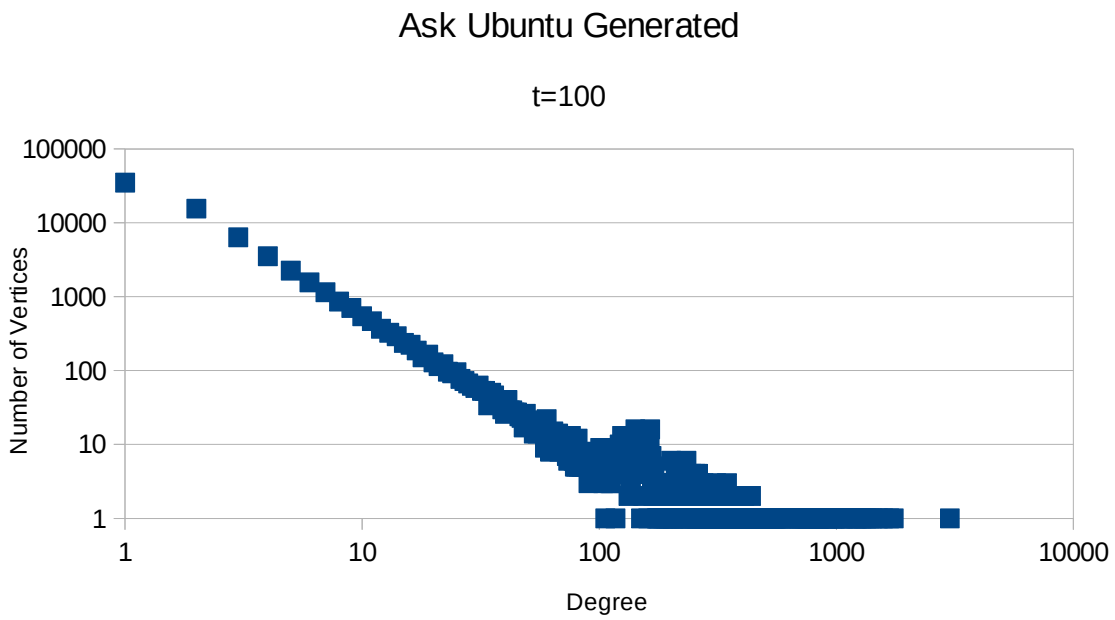


Figure 7.4 Degree Distribution of generated Ask-Ubuntu Dataset at time 100 in logarithmic scale.

Time 150

Linear Scale

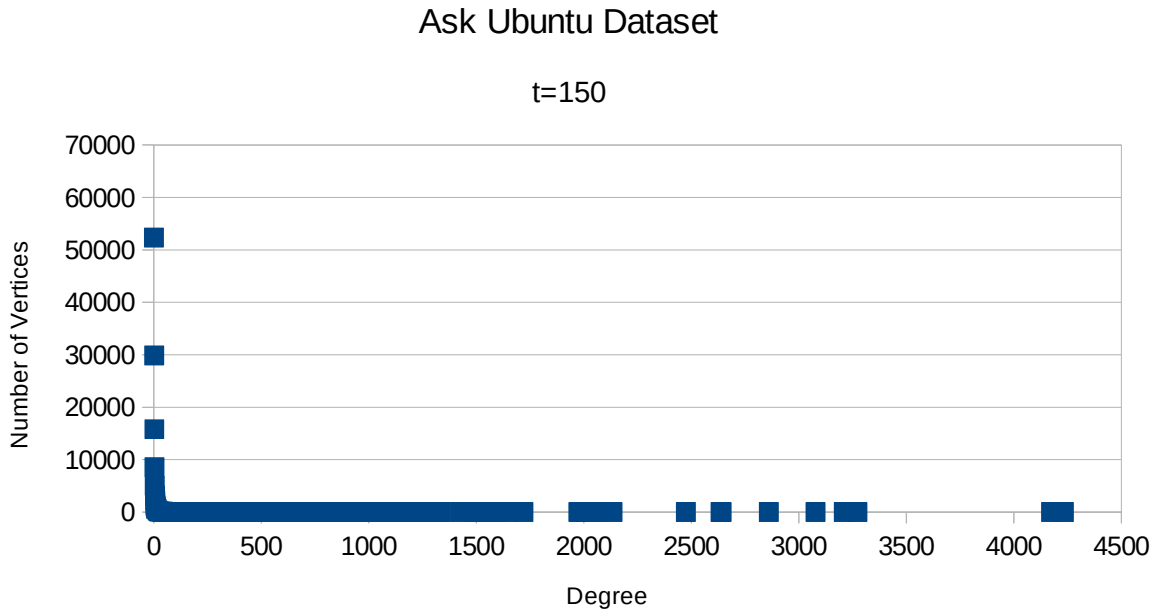


Figure 8.1 Degree Distribution of collected Ask-Ubuntu Dataset at time 150 in linear scale.

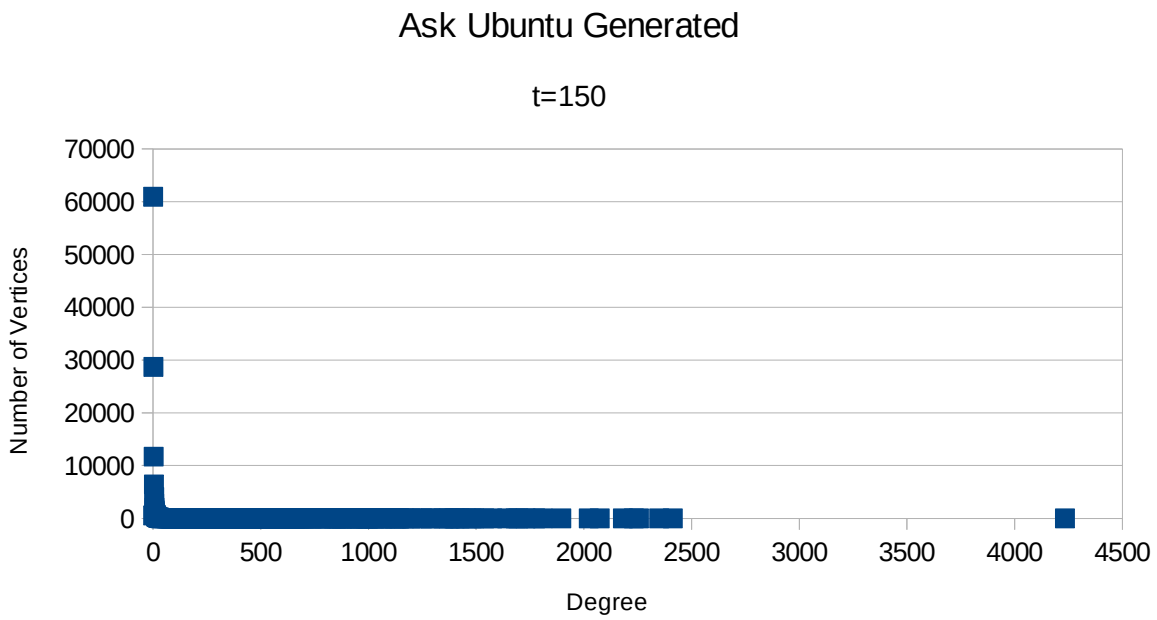


Figure 8.2 Degree Distribution of generated Ask-Ubuntu Dataset at time 150 in linear scale.

Logarithmic Scale

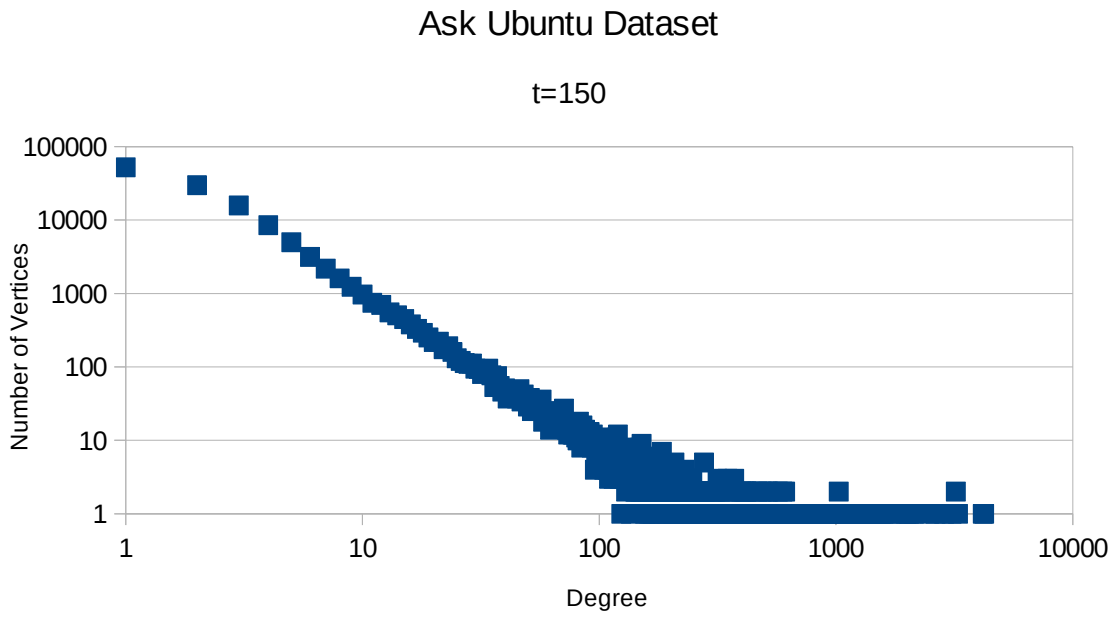


Figure 8.3 Degree Distribution of collected Ask-Ubuntu Dataset at time 150 in logarithmic scale.

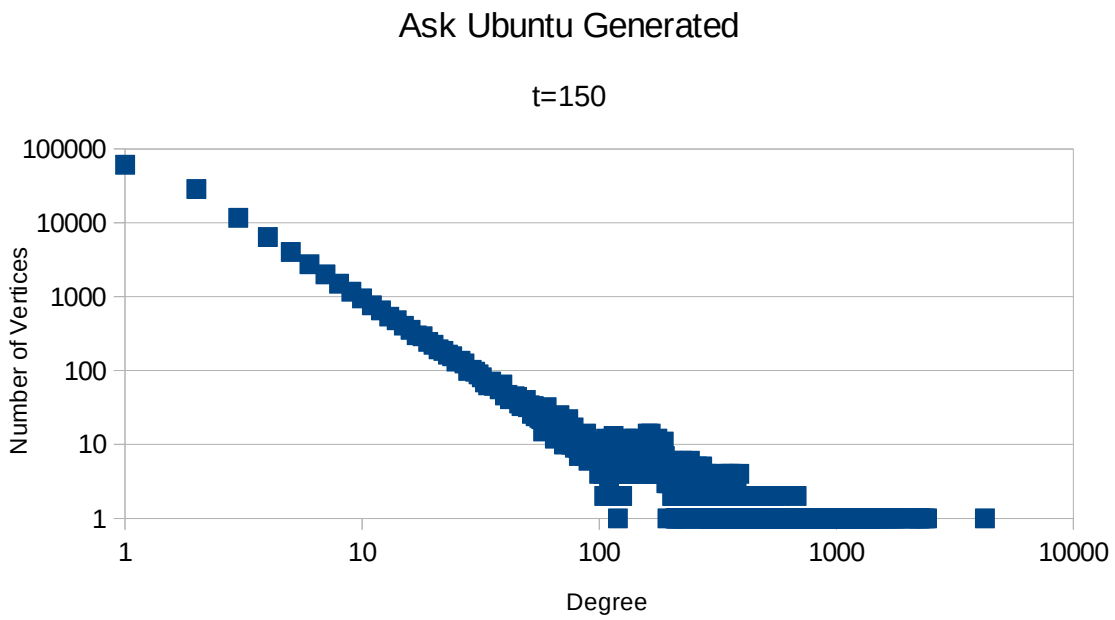


Figure 8.4 Degree Distribution of generated Ask-Ubuntu Dataset at time 150 in logarithmic scale.

Time 177

Linear Scale

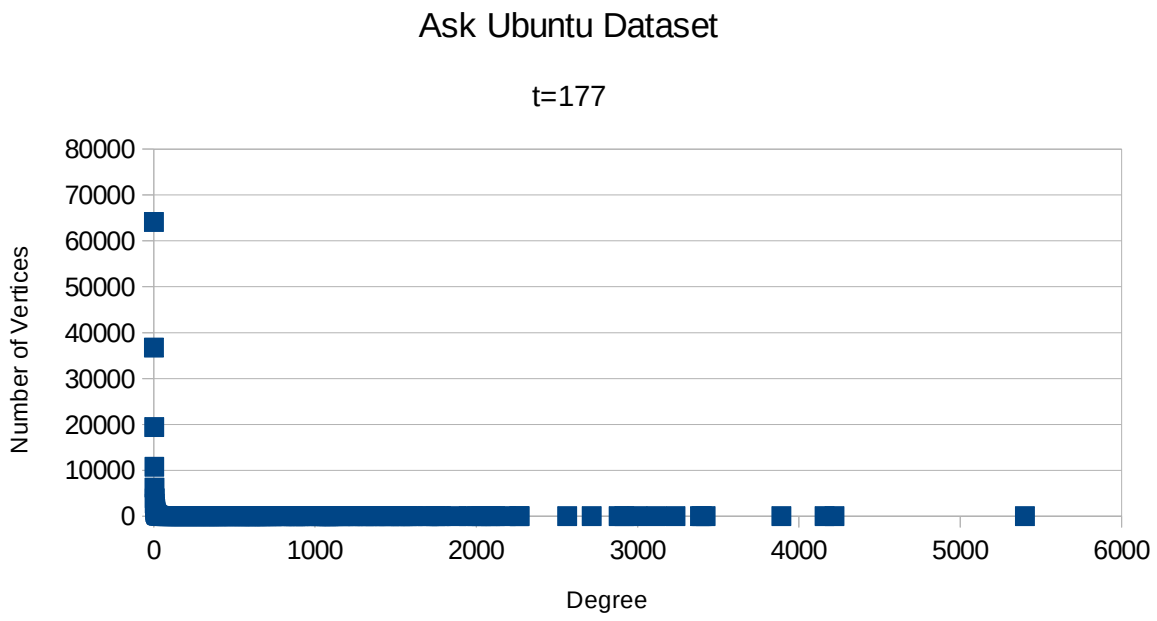


Figure 9.1 Degree Distribution of collected Ask-Ubuntu Dataset at time 177 in linear scale.

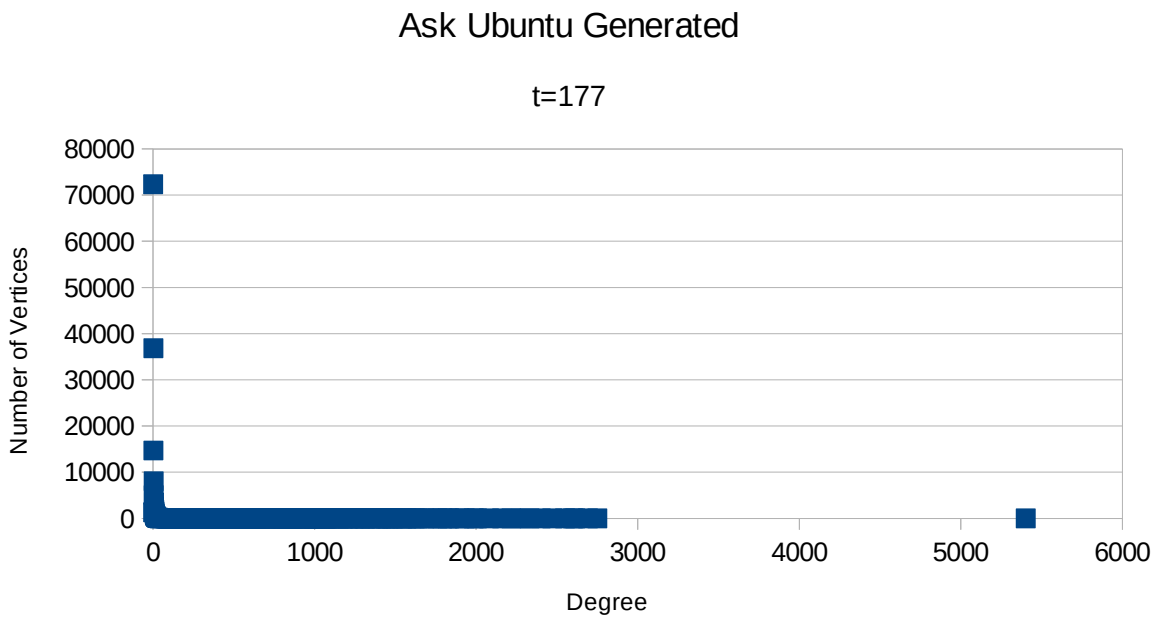


Figure 9.2 Degree Distribution of generated Ask-Ubuntu Dataset at time 177 in linear scale.

Logarithmic Scale

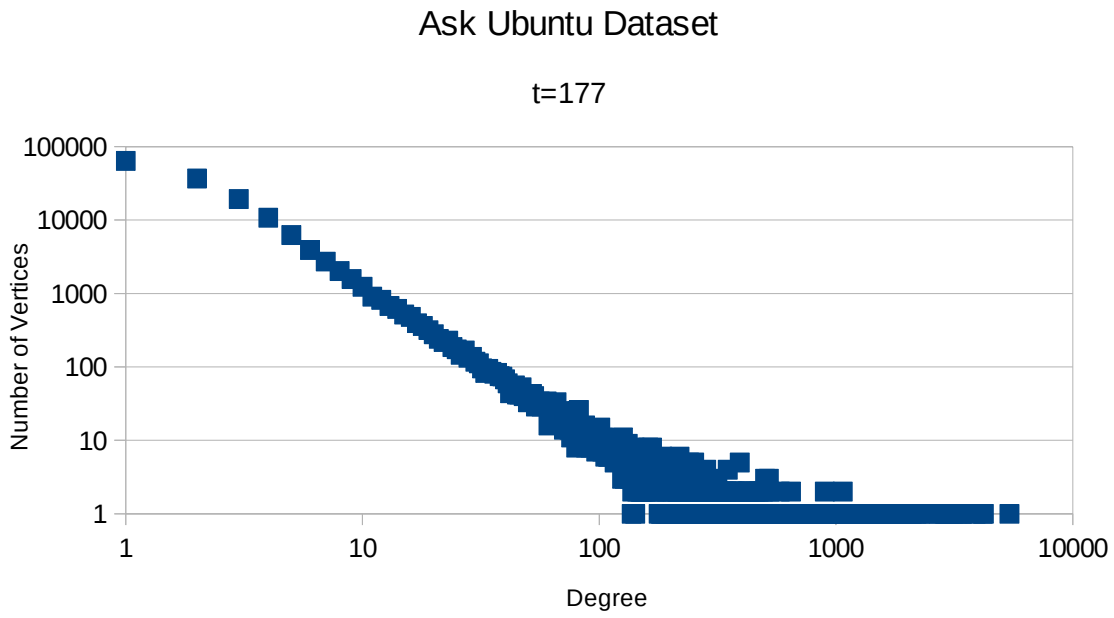


Figure 9.3 Degree Distribution of collected Ask-Ubuntu Dataset at time 177 in logarithmic scale.

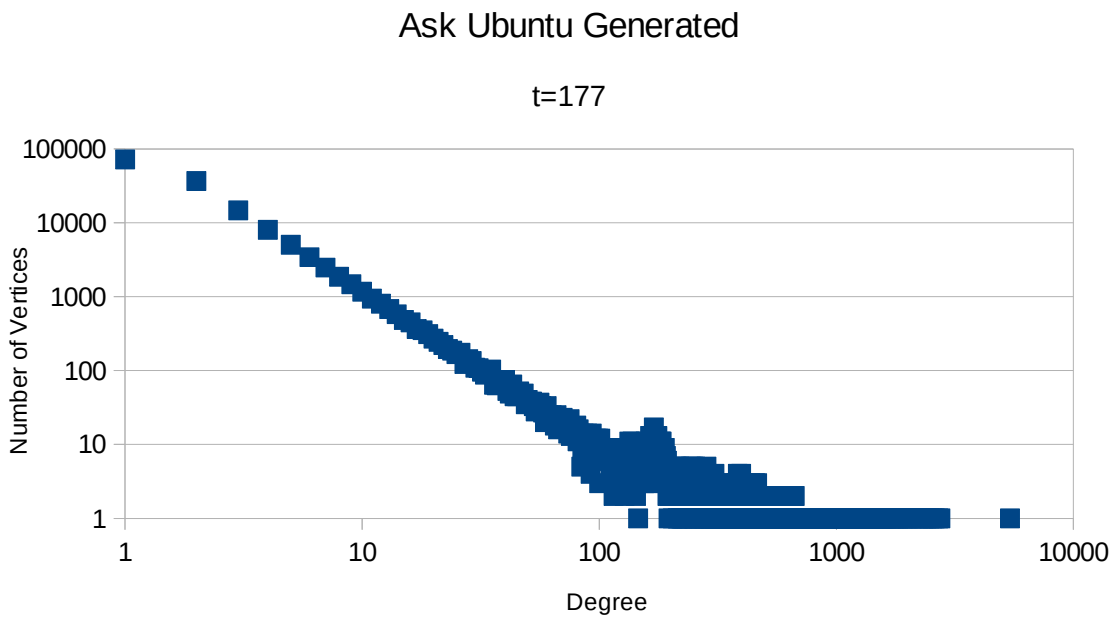


Figure 9.4 Degree Distribution of generated Ask-Ubuntu Dataset at time 177 in logarithmic scale.

As observed from the above degree distributions, the generated dynamic graph mimics the collected ones' degree distribution as time progresses. It is observed that the distribution maintains power law in a similar manner to that of the collected dataset. It can be concluded that the tool works more effectively for graphs with higher number of edges, vertices and maximum degree.