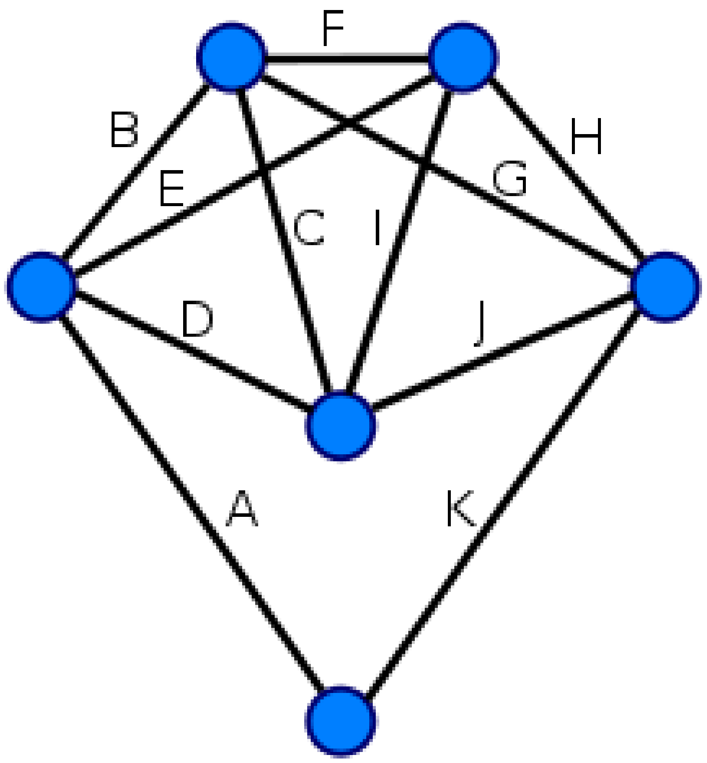


On sparse spanners of weighted graphs

Ben Gurion University

Shir Lupo |20302951
Elad Shamailov |308540202

Table Of Contents

Introduction

The algorithm we have chosen to implement was taken from the paper "On sparse spanner of weighted graphs" by Ingo Althofer, Gautman Das, David Dobkin, Deborah Joseph and Jose Soares.

This algorithm purpose is finding a t-spanner for weighted graph.
Let G = (V,E) be a connected n-vertex graph with arbitrary positive edge weights. A subgraph G' = (V,E') is a t-spanner if, between any pair of vertices, the distance in G' is at most t times longer than the distance in G. The value of t is the stretch factor associated with G' .

Sparsity is measured according to two criteria

1. Sparse in weight, which means the weight of the total edges is small.
2. Sparse in size, which means the number of edges is small.

The spanner problem appear in numerous applications e.g. biology in the process of reconstructing phylogenetic trees, underlying graph structure in distributed systems and communication networks constructions, etc.

This algorithm gives us a spanner which is sparse in size and also sparse in weight.

The Algorithm

Pseudo-code for the algorithm

```
SPANNER (G = (V,E), r)

G is a connected graph

r  is the stretch factor

1: Sort E by non-decreasing order

2: Create H= (V,E') where E' = { }

3: For each e = (u,v) in E do:

        Compute the shortest path P from u to v in H
        if $r * W(e) < W(P)$
            $E' \leftarrow E' \cup \{e\}$

4. Return  H
```

1. Let G = (V,E) be a connected graph, and r is the stretch factor.

2. The algorithm sorts the group of edges in a non decreasing order by it's weights.

3. First, in the algorithm constructs graph H with the same group of vertices as G, but with an empty group of edges E'.

4. On stage 3 in each loop, the next lightest edge of G is examined . the algorithm defines P as the shortest path between the two edges in H, the new graph. The algorithm compared the weight of the given edge with the current r and the weight of P.

5. Based on the result , the algorithm chooses If to add the examined edge of G to the edges in H.

6. The algorithm returns H, t-spanner.

Implementation of the algorithm

We chose to implement the algorithm in Python and we used "NetworkX" , Python package which has many standard graph algorithms, and where the nodes can hold arbitrary data like weights etc.

First, we wrote a method which created a connected graph as required in the algorithm, using a while loop. We used the nx.erdos_renyi_graph which creates random graph with n vertices and every edge exists in probability p. Only if the received graph is connected we return the new graph G.

After, we wrote a method "Sort_edges" who sort the edges by non-decreasing order with the function 'sorted' in python.
Then - we build the subgraph H. We create new graph with the same vertices as G with the function 'add nodes' from nx library.
than, for every edge in G we check if there's a path in H. If not – than the shortest path is assigned to be infinity, if there is a path we assign a new variable 'shortest_path_weight' to represent us the weight of the shortest path which has been found by the famous algorithm of Dijkstra
To implement Dijkstra we use nx special implementation of Dijkstra.
Then like said in the algorithm, if $r*W(e) < W(P)$ where's P is the shortest path we found using Dijkstra we add edge e to graph H.
Finally, we return the graph H.

Experiment 1 – test the relationship between the stretch factor and the number of edges in the spanner

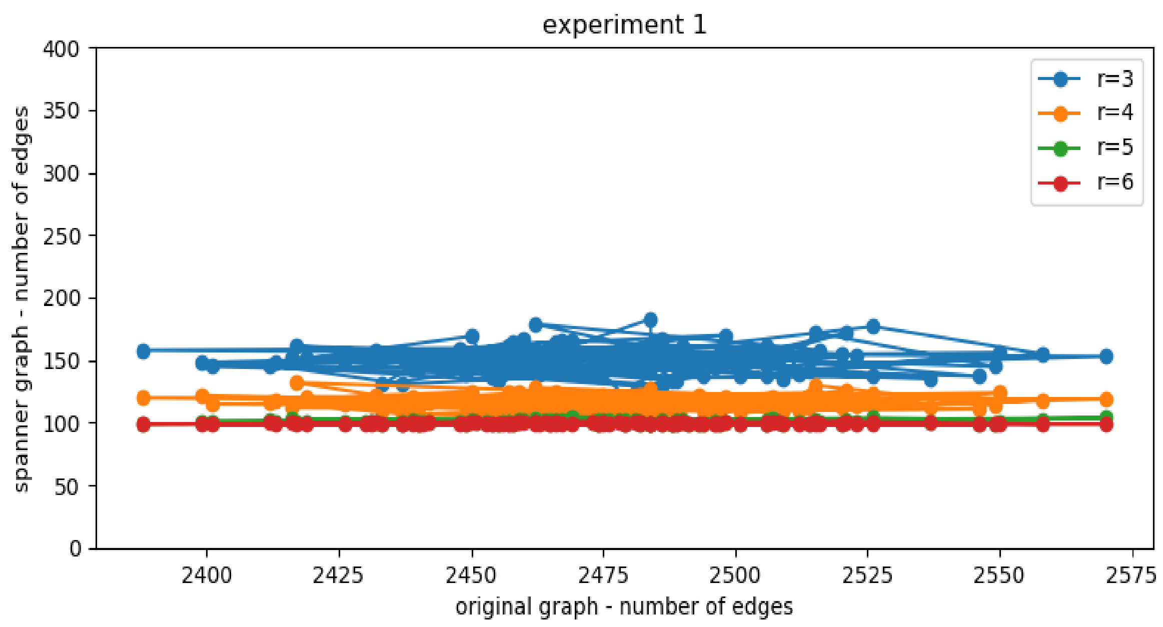
In this experiment, we wanted to test the relationship between the stretch factor and the number of edges in the spanner graph. In addition, we wanted to determine what is the main factor for the number of edges in the spanner graph: Is it the number of the edges in the original graph? Is it the stretch factor? Or maybe neither of them?

In order to conduct the experiment, we ran the algorithm on 100 random graphs with the following parameters:

- Number of verticals: 100
- p = 0.5 (probability for an edge)
- constant weight for an edge - for each e that belongs to E , w(e) = 1

we ran the algorithm 100 times with: r=3 , r=4 , r=5 and r=6

In the following chart, we can see the results of the experiment:



It's easy to see from the graph that the number of the edges in the spanner graph does not depend on the number of edges in the original graph, it depends on the stretch factor. (Each graph had random number of edges, but the reason that the lines of the number of edges in spanner graph goes down is the stretch factors that are represented each one by different color in the graph).

We expected to see results like this, because according to the paper:

$$Size(G') < n\left\lceil n^{\left(\frac{2}{r-1}\right)}\right\rceil$$

And from the graph we immediately see that the size of G changes when we change r. when the stretch factor is bigger, then the number of edges in the spanner is smaller.

Experiment 2 – test with stretch factor 1 and constant weight

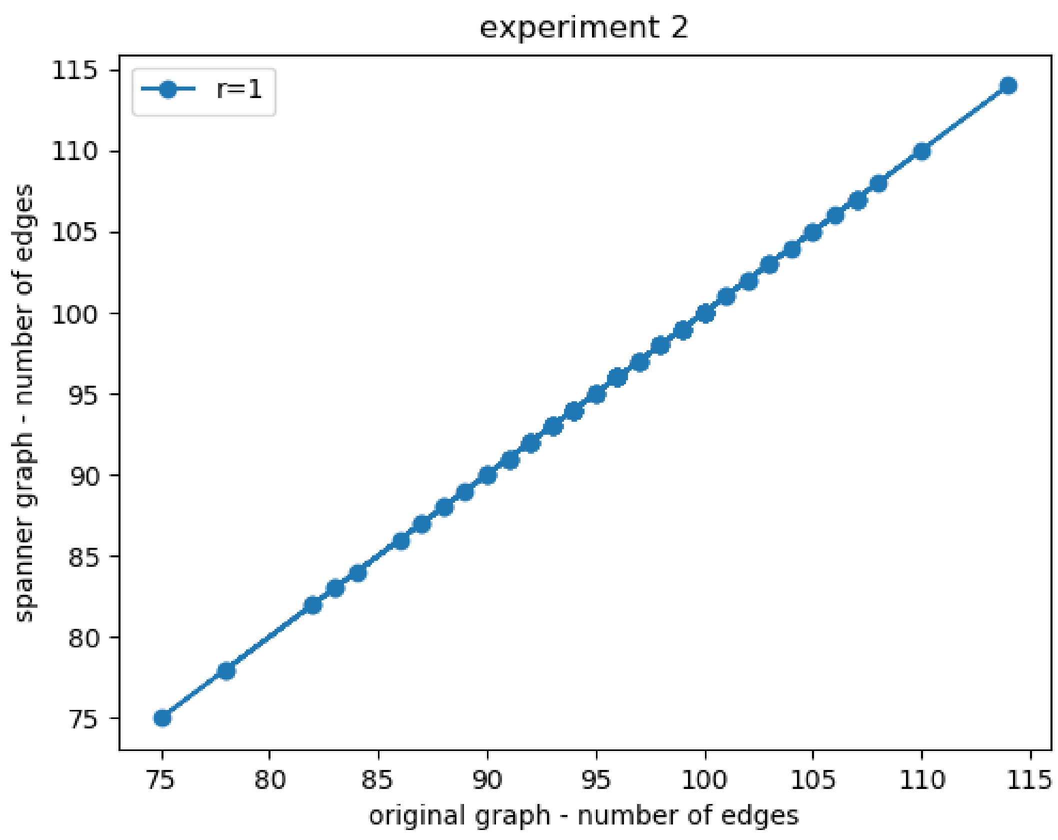
We wanted to test what happens when we set w(e) = c for every e that belongs to E and r = 1.

We expect by theory and by the paper, that this stretch factor will has no meaning and that the graph will remain the same.

In order to test it, We ran the algorithm on 100 random graphs with the following parameters:

- Number of verticals: 20
- p = 0.5 (probability for an edge)
- constant weight for an edge - for each e that belongs to E , w(e) = 1
- r = 1

In the following chart, we can see the results of the experiment:



We can see that we got linear graph, where's for every (x,y) , x=y. hence , we got that in each of the 100 graphs , when we set r = 1 we get spanner graph that is the same as the original graph with the same number of edges.

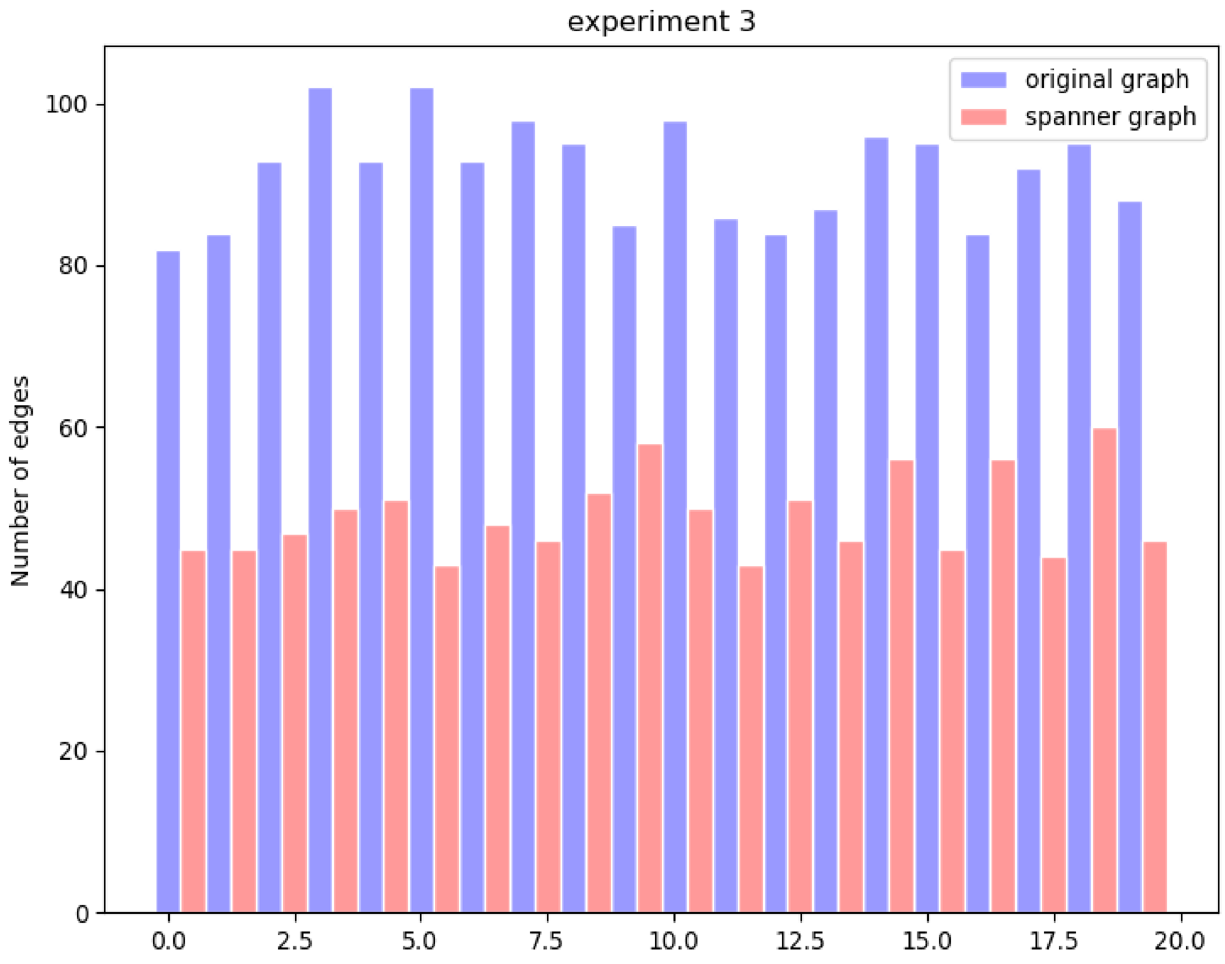
Experiment 3 – test with stretch factor 1 and random weight

In experiment 2 we saw that when we set constant weight for all the edges in the graph, the spanner graph is the same as the original graph. In this experiment, we want to see what happens when we set random weight for each edge in the graph.

In order to test it, We ran the algorithm on 20 random graphs with the following parameters:

- Number of verticals: 20
- p = 0.5 (probability for an edge)
- w = random (1,5) , random weight for each edge
- r = 1

In the following chart, we can see the results of the experiment:

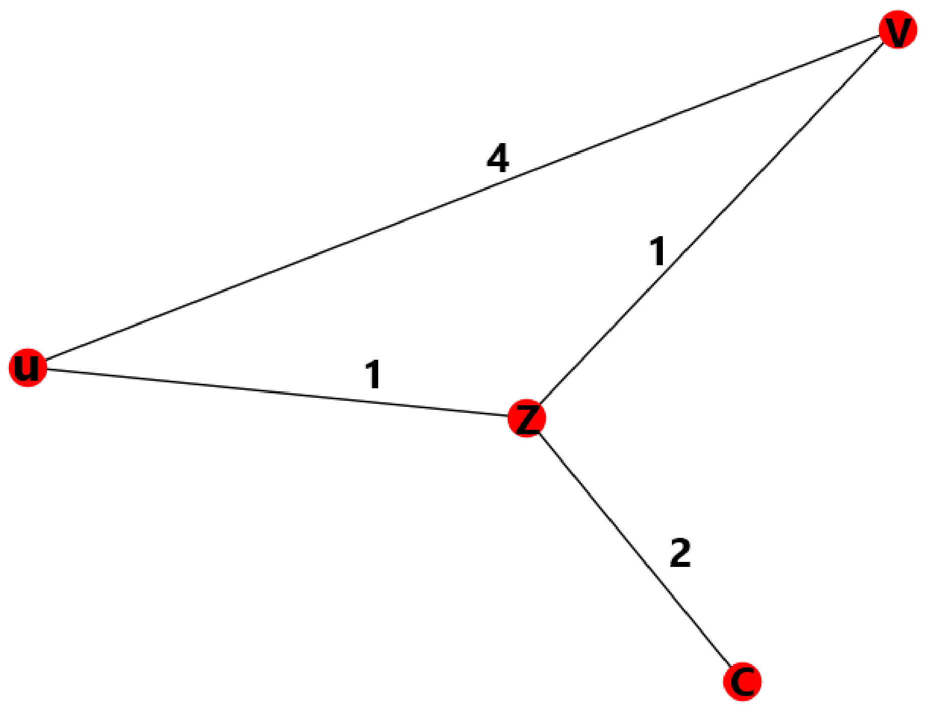


We can see from the chart that in contradiction to experiment 2, if the weight is not constant – the spanner graph is not the same as the original graph.

We see in the experiment that even for r = 1, if the weight is random so the number of edges changes in the spanner graph.

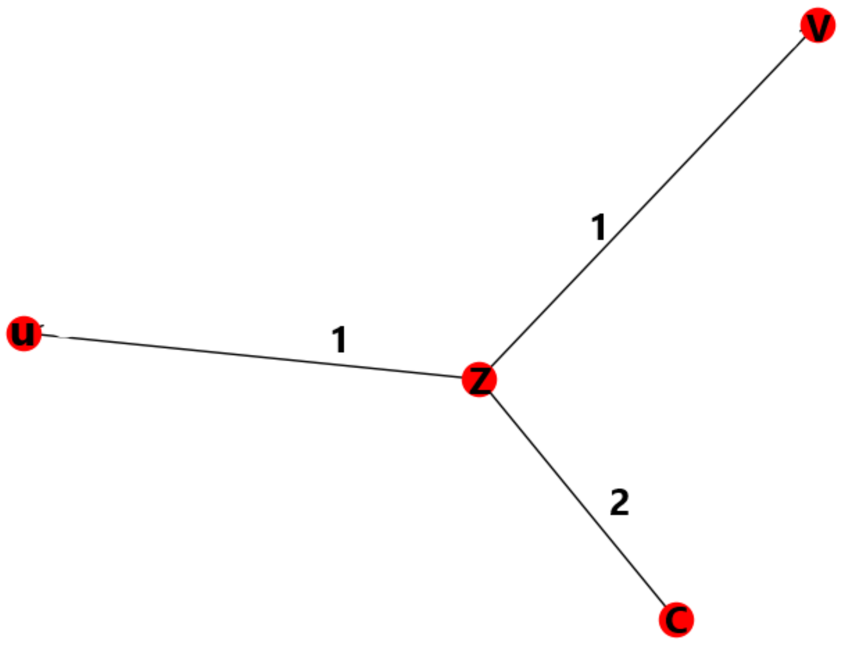
The reason we didn't get identical number of edges is because when we build the spanner graph, and not all the weights are the same, we have situations where we examine edge (u,v) from the original graph, but in the spanner graph we already have a lighter path from u to v.

For example, let's look at this graph:



We know from the spanner algorithm, that we order the edges in non-decreasing order. Hence, when we will examine the edge (u,v) with weight 4 , we will already have a lighter path from u to v , this path is u,z,v with the weight of 2.

So the spanner graph will have 3 edges instead of 4, and will look like this:



Experiment 4 – test with changing density

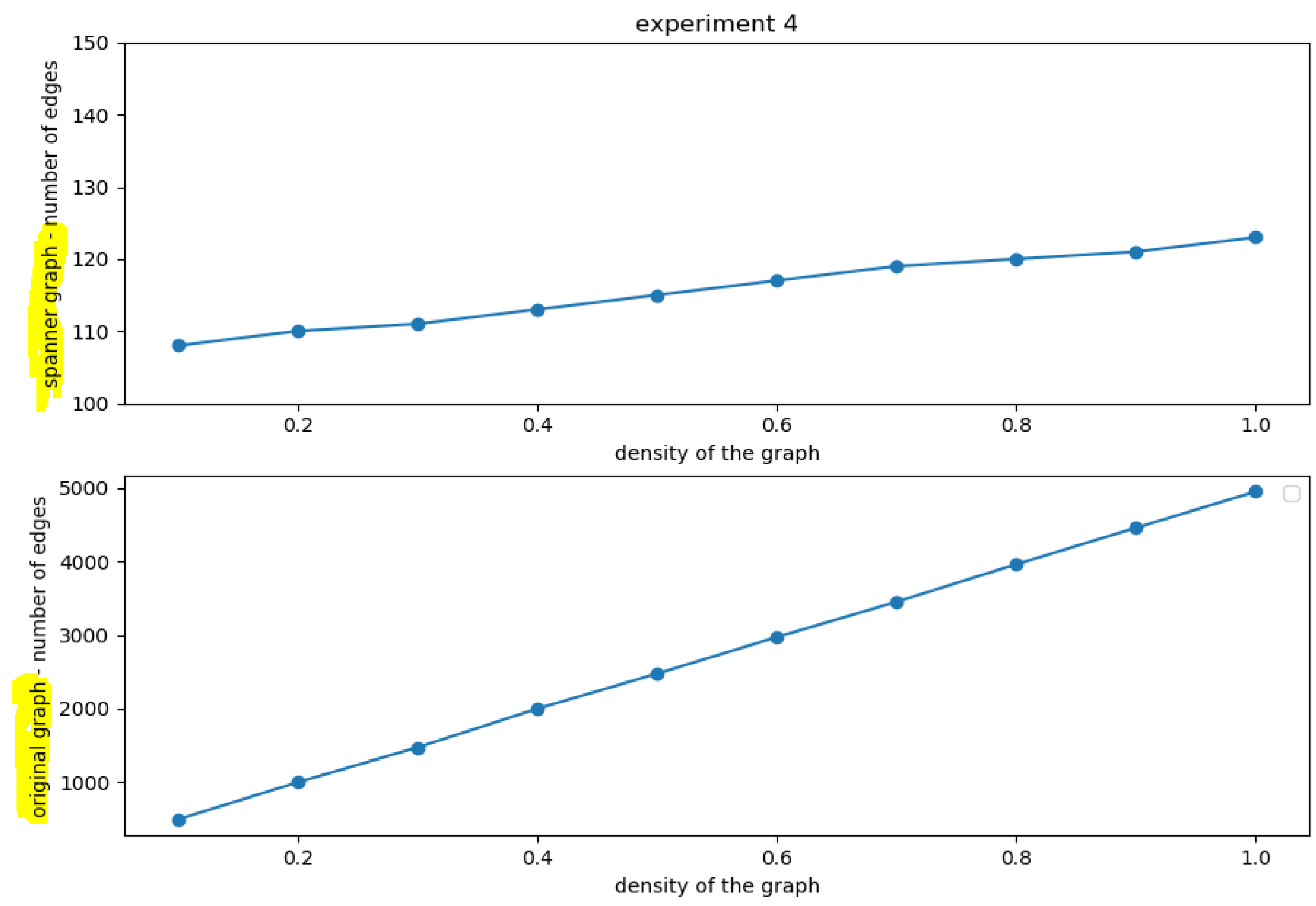
In this experiment we wanted to test how the density in the graph affects the number of the edges in the spanner graph.
We expect a minor effect on the number of the edges in the spanner graph, because by the paper we know that even for a dense graph with $\Omega(\mathrm{n}^2)$ edges, sparse spanners exist which have good stretch factors.

In order to conduct the experiment, we ran the algorithm on 20 random graphs with the following parameters:

- stretch factor r = 4
- random weight for each edge , from 1 to 50
- verticals number = 100

We ran the algorithm 20 times for each p, starting from p=0.1 and increasing in 0.1 each time.

In the following chart, we can see the results of the experiment:



As we can see from the graph, the results correlate our expectations. Although the number of the edges in the original graph always raises (because we raise the density), the number of the edges in the spanner graph almost remains the same. It means that the density does not affect the number of the edges in the spanner graphs.

Experiment 5 – test with changing weight

After we tested the effect of the density of the graph on the number of the edges in the spanner graph, we want to test how the weight range in the original graph (weight per edge) effects the number of the edges in the spanner graph and whether or not the stretch factor effects the results.

If we look at experiments 2 and 3, we can say that we expect to see that as the weight range getting bigger, we get less edges in the spanner graph.

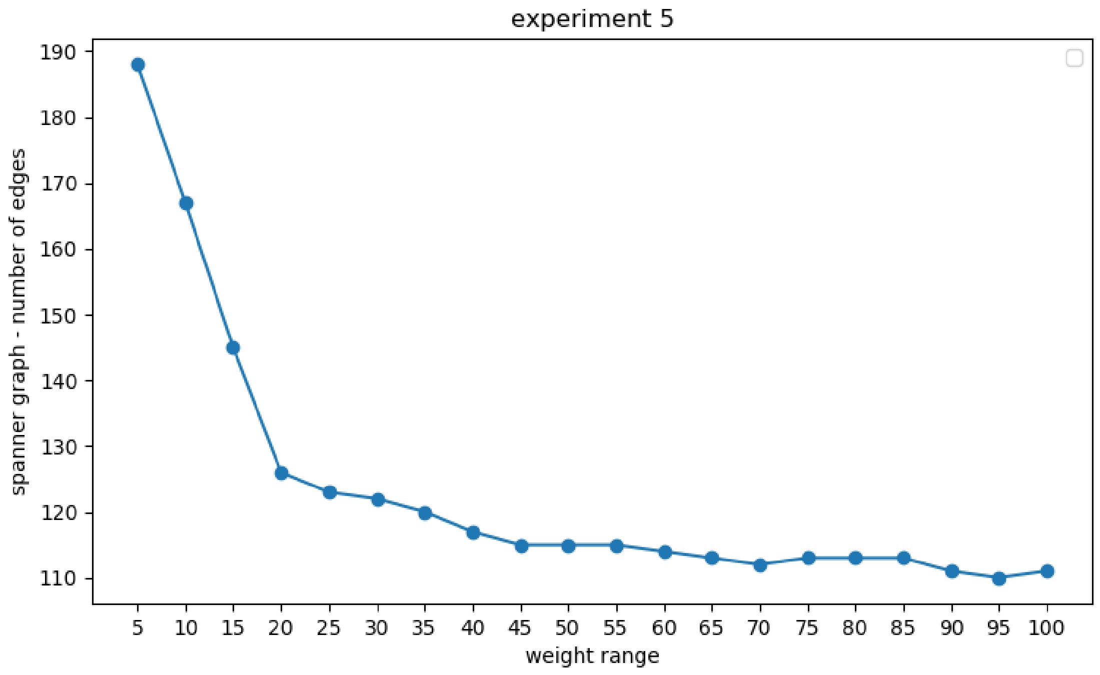
In order to conduct the experiment, we ran the algorithm on 20 random graphs with the following parameters:

- stretch factor r = 4, r = 7and stretch factor r = 9
- p = 0.5
- verticals number = 100

We ran the algorithm 20 times for each w range, starting from (0, 5) and increasing in 5 each time the range until we got the range (0, 100)

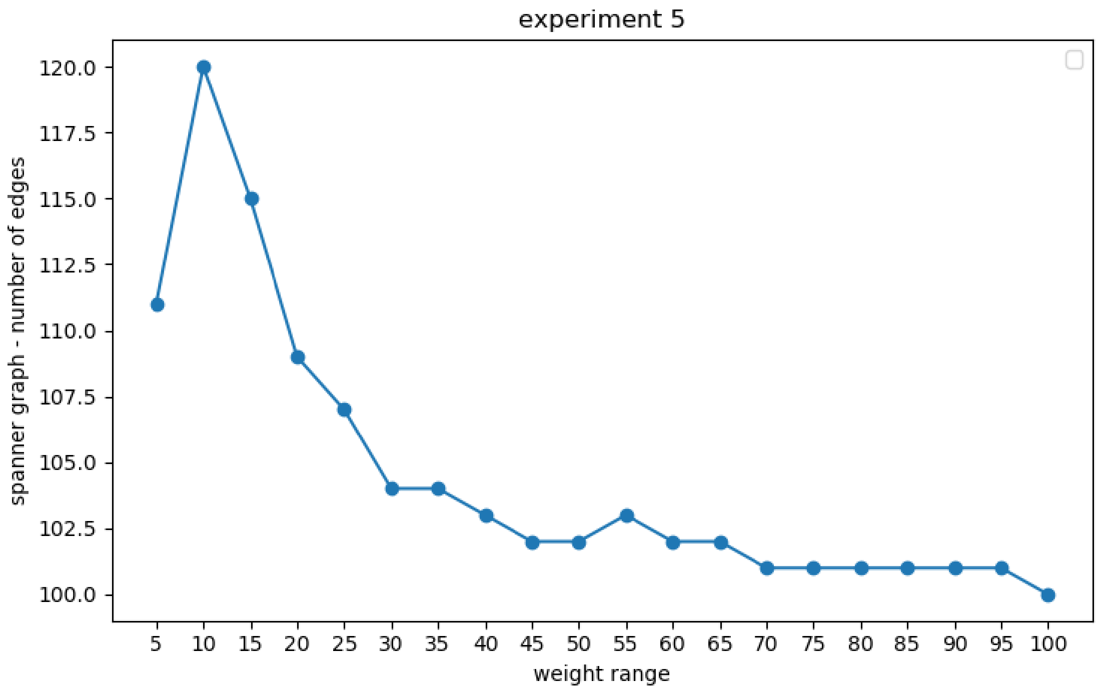
In the following chart, we can see the results of the experiment with

r = 4:

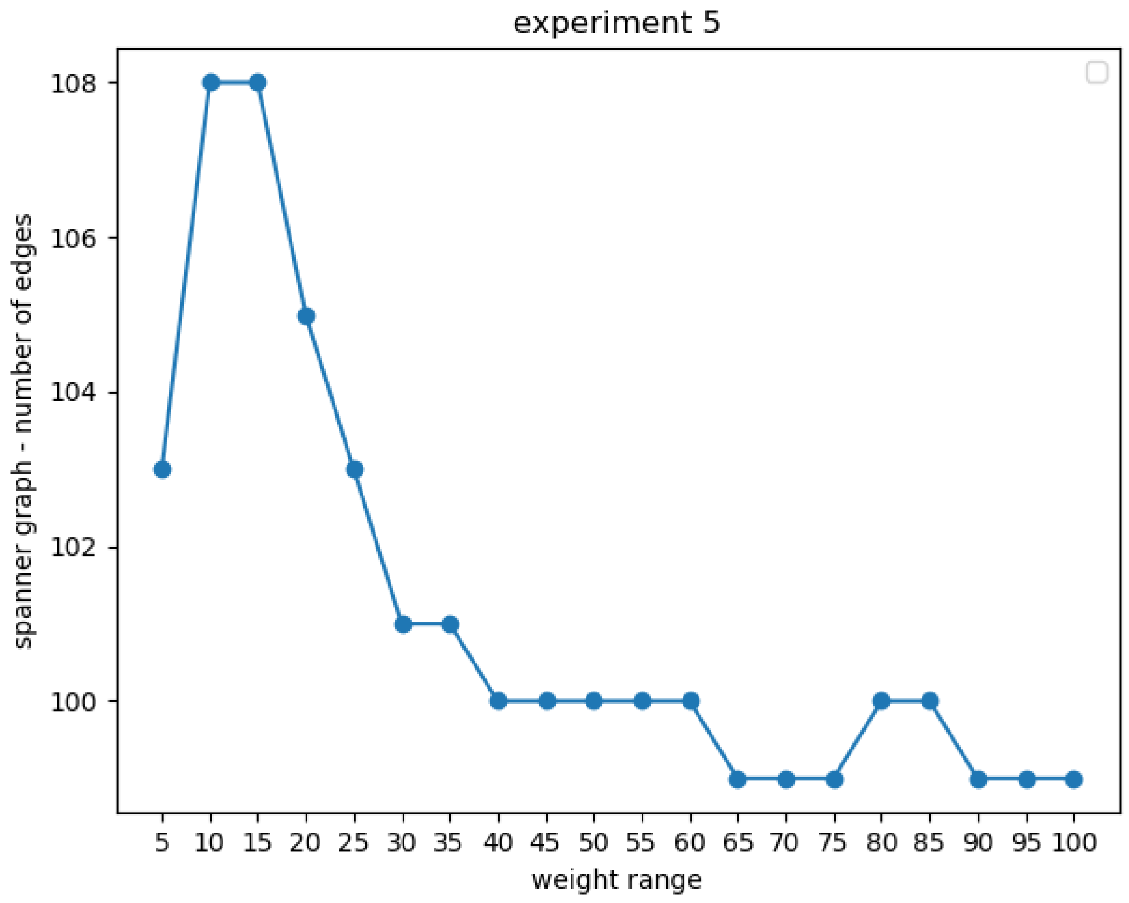


In the following chart, we can see the results of the experiment with

r = 7:



In the following chart, we can see the results of the experiment with r = 9:



As we can see, if the weight range in the original graph is bigger, the spanner graph has less edges.

In addition, we can see from the experiment that as the stretch factor is higher, the weight range effects less. With r = 4 with have big effect, and with r = 7 and r = 9 we have significantly less effect.

Conclusions and directions for future work

Conclusions:

After we have done the experiments, we found out that most of what we thought that would happen in the experiments actually happed.
We found out the stretch factor has major effect on the number of the spanner graphs, Also we got to the conclusion that the weight range do effect the number of the spanner graph edges, as the weight range goes higher, the number of the edges goes down. In addition we got to the conclusion that if the stretch factor is bigger, the weight range effects less on the number of the edges in the spanner graph.
Also, from the experiments we got to the conclusion that the density of the original graph does not really effect the number of the edges of the spanner graph, which sets up with the conclusion that the number of edges in the original graph does not affect the number of edges in the spanner graph, what effects is the stretch factor.

Future work:

For future work we would like to explore more about the relation between the stretch factor and the weight range.
We found that as the stretch factor getting higher, the effect of the weight range on the number of edges gets lower. We would like to conduct more experiments about this in the future, and find out if there are more correlations and effects between each one of them.

In addition, in the future work we would like to conduct the experiments in much larger number of graphs, and on larger graphs.
The reasons for that is that first of all, when we conduct experiments on larger sets, we get more precise answers and results. Secondly, we want to test if larger graphs with more nodes will yield different results and maybe different or more precise results. In order to that in future work, we will have to work on stronger computers.
By that, we could also explore better the effect on spanner graph when the stretch factor is getting really high, and we would check if on this high numbers the stretch factor still has all his effects on the spanner graphs and on the number of edges.

Bibliography

- *I. Alth\"ofer, G. Das, D. Dobkin, D. Joseph, and J. Soares,*
  On sparse spanners of weighted graphs,
  Discrete Comput. Geom., 9 (1993), pp. 81--100.

- Geometric spanner
  From Wikipedia, the free encyclopedia