Mini Project:

Low-Distortion Embedding

Supervisor: Prof. Michael Elkin.

Student: Netta Ben Ezri.

אדר תשע"ח

February 2018

Table Of contents

Introduction

This report is based on the paper “On Sparse Spanners Of weighted Graphs” by Ingo Althofer et al.

The paper provides an algorithm for constructing spanners.

Let $G = (V, E)$ be a connected n-vertex graph with arbitrary positive edge weights. A sub-graph $G' = (V, E')$ is a $k - spanner$ of G iff between each pair of verdicts u, v the distance between u and v in G is no longer then k times the distance between them in G' i.e. $d_{G'}(u, v) \leq k \cdot d_G(u, v)$. The value of k is the stretch factor associated with G'. The algorithm builds sparse spanners with a constant stretch factor k that is independed of the size of the graph.

Sparsity is measured by two criteria: size and weight, where $size(G)$ denotes the number of edges in the graph$G$, and $weight(G)$ is the sum of the graph’s edge weights, i.e. $\sum_{e \in E} w(e)$. Both $size(G)$ and $weight(G)$ must be relatively small for the spanner to be considered as sparse.

The correctness of a spanner also depends on two criteria: size and distances.
Given a graph G', a $k - spanner\ G'$ to G' have to satisfy the following demands:

1. $size\ (G') < n \cdot \left\lceil n^{\frac{1}{k}} \right\rceil$ where n is the number of vertices
2. For each, $v \in V$:$d_{G'}(u, v) \leq k \cdot d_G\ (u, v)$.

The paper describes the construction of the sparse spanner for general graphs,
Then it presents several bounds for the results and then focus on spanners for Euclidean graphs.
In this report I concentrated on construction of the spanners.

Motivation:

As the article claims spanners appear to be the underlying graph structure in various constructions in distributed systems and communication networks. Another example is in biology – in the process of reconstructing phylogenetic trees from matrices, whose entries represent genetic distances among contemporary living species there is use to spanners and their qualities. Robotics researches has studied spanners under the constraints of Euclidean geometry, where vertices of the graphs are points in space and edges are line segments joining pairs of points.

Algorithem:

The algorithm's name is $SPANNER(G,k)$. It receives a weighted graph G and k the stretch factor. The algorithm builds a subgraph G' of G wich is a $k-spanner$ graph for G.
This Is essentially a generalization of $The\ Kruskal\ Algorithm$ for generating MST's.
Let's look at the MST of G and check whether it is a possible spanner:
Clearly it is the sparsest spanner both under size and distances criteria, since it is the subgraph of G with the least total edge number and weights that still connects all vertices. Nevertheless its stretch factor could be $\Omega(n)$ in worst case scenario, when G is a n vertex cyclic graph.
Since the purpose of the algorithm is constructing spanners with a constant stretch factor independent of n , it uses $MST(G)$ as a scale for for measuring sparseness and tries to build spanners whose sparseness is close to it.
Note that sparseness can be made arbitrarily close to the $MST(G)'s$, at the expense of increasing the stretch factor k.

SPANNER $(G=(V,E)$, W , k):
1. begin:
1.1 sort E in a nondecreasing order by edge weight
1.2 $G' \leftarrow (V,\{\})$
1.3 for every edge $e=(u,v) \in E$ do:
1.3.1 begine:
1.3.1.1 $compute\ P(u,v) = the\ shortest\ path\ from\ u\ to\ v\ in\ G'$
1.3.1.2 $if\ (k \cdot weight(e) < weight(P(u,v))\ then:$
1.3.1.2.1 $add\ e\ to\ G'$
1.3.2 end
1.4 $output\ G'$
1.5 end

Implementation:

I used java to write the code that implements the algorithm and experiments. Java is a very convenient language for designing an Object Oriented Program. In this case implementing a graph data structure is easy in an object oriented manner.
This implementation consists of the following classes: Vertex, Edge, Graph, Tests and CsvFileWriter.

The first two classes are quite simple and only has getters and setters:
Class Vertex is simply a definition of a vertex. Class Edge consists of three fields- two vertices and an integer which represent the weight of the edge.

Class graph is a bit wider in the manner that in addition for its fields (which are two arrays one is an array of vectors and the other is an array of edges) and getters/setters,

it has the article's algorithm function implementation in it , as well as implementation of several known algorithms such as bubble sort and relax.
Class Tests has the test cases on it, and class CsvFileWriter is a class I added in order to write the data that resulted from a test into a csv file, in order to make the test results to be easily examined (by pressing F11) , also for further work.

Experiments:

Every test (which is a function in class Tests) is meant to examine a different aspect of the graphs produced by the algorithm. Let's review:

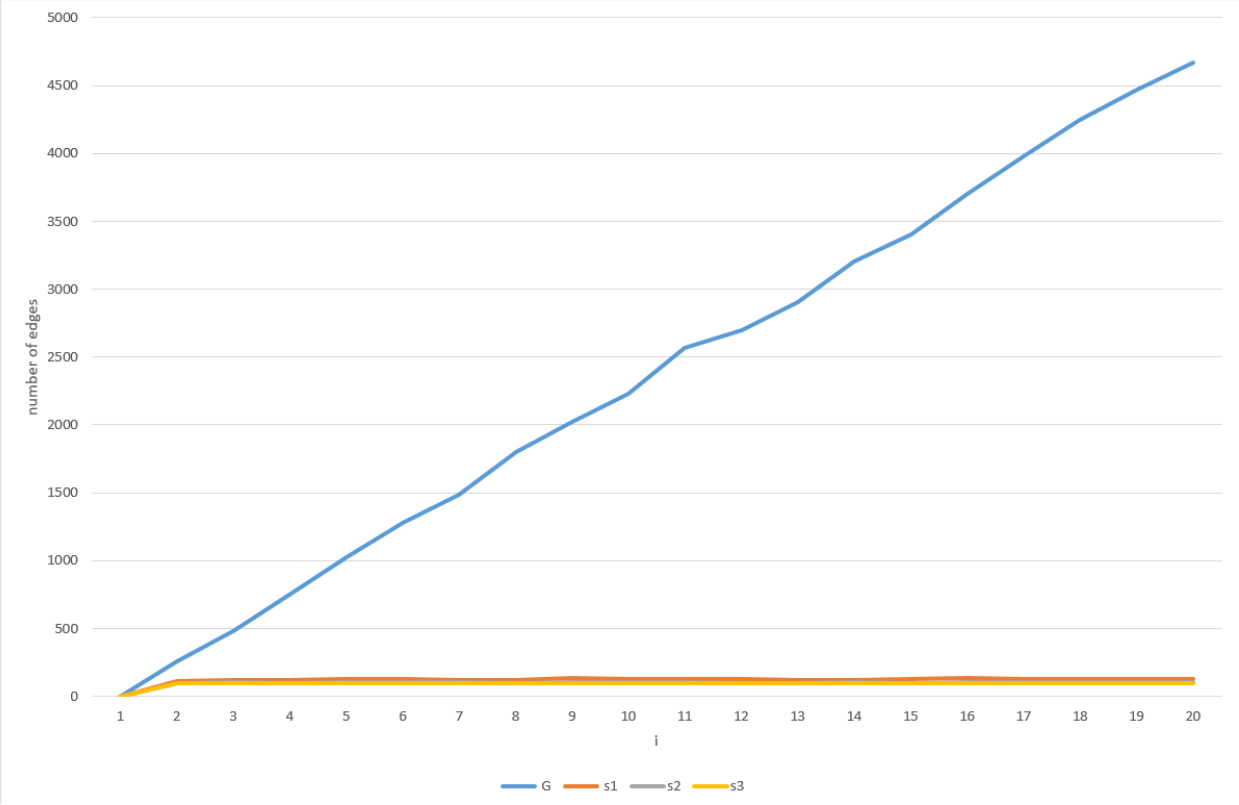
Size(G) Aspect: function edgesAspectExperiment :

This test's purpose is to examine the number of edges in the spanner, in comparison to the number of edges in the original graph G, and different stretch factors.
This test receives a sampleSize – number of graphs, numOfVertics – the number of vertices and three stretch factors $t1, t2, t3$ (and the output file name), and produces sampleSize graphs and 3 sampleSize spanners, each one with their compatible to the original graph i and it's stretch factor $t1, t2$ or $t3$.
As the article proves, given a $n - vertex$ graph G and its corresponding $(2t-1) - spanner\ G'$ it must be satisfied that $size(G') < n \cdot \left\lceil n^{\frac{1}{k}} \right\rceil$. The goal is to see how much $size(G')$ is close to the upper-bound, and how k, the stretch factor and p – the probability for an edge to be in the graph which is defined to be $\frac{i}{sampleSize}$ so the graphs get denser as i increases.
Here are the results:
In the first test we took a sample size of 20 graphs, each of them is a 100 vertices graph.
$t1 = 2 \rightarrow s1[i]$ is a $3 - spanner$.
$t1 = 4 \rightarrow s2[i]$ is a $7 - spanner$.
$t1 = 7 \rightarrow s3[i]$ is a $13 - spanner$.

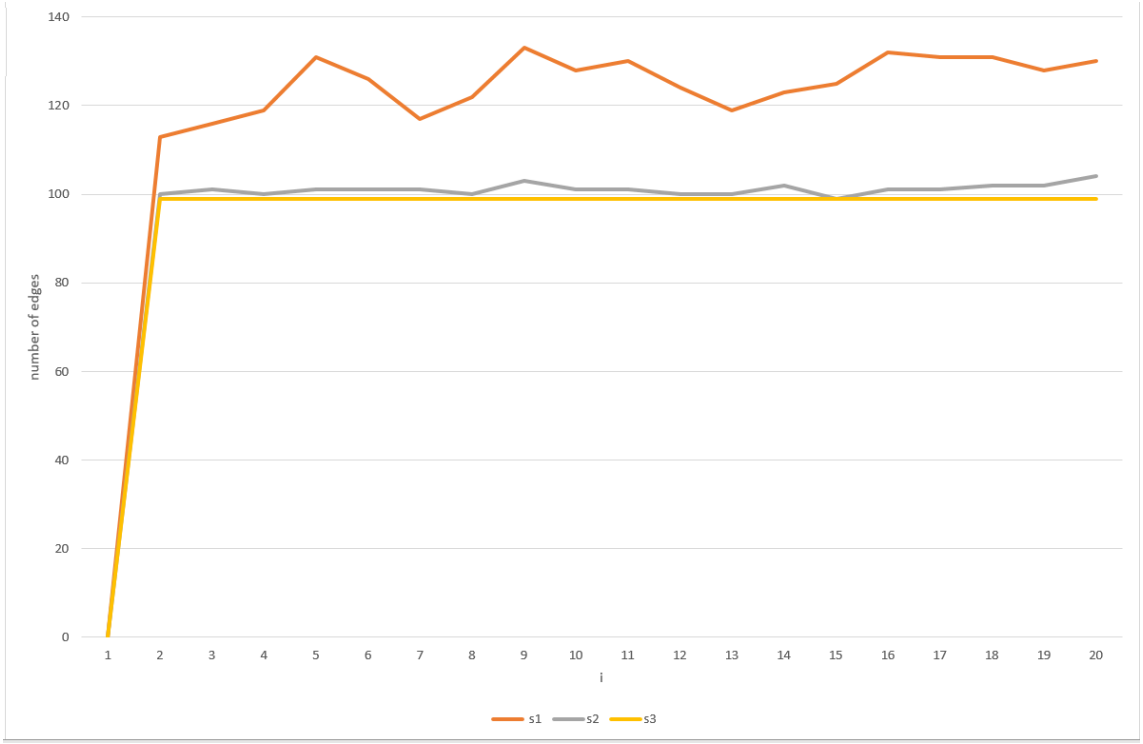
1	G	s1	s2	s3
2	0	0	0	0
3	254	113	100	99
4	484	116	101	99
5	752	119	100	99
6	1027	131	101	99
7	1277	126	101	99
8	1482	117	101	99
9	1797	122	100	99
10	2022	133	103	99
11	2229	128	101	99
12	2565	130	101	99
13	2697	124	100	99
14	2904	119	100	99
15	3206	123	102	99
16	3401	125	99	99
17	3699	132	101	99
18	3981	131	101	99
19	4247	131	102	99
20	4470	128	102	99
21	4670	130	104	99

On the cells in the chart above there are the number of edges in graphs $G[i]$, $spanner1[i], spanner2[i]$ and $spanner3[i]$.

Let's look at this in a more visualized way:



As you can see, the spanners has very few edges so they are very sparsed. Also, as printed in the console/command line :
The upper bound for stretch factor t1 = 2 is: 1000.
The upper bound for stretch factor t2 = 4 is: 400.
The upper bound for stretch factor t3 = 7 is: 200.
Now, in order to see in more specific the difference between the spanners and how much they are close to their upper bounds, let's observe only on them:



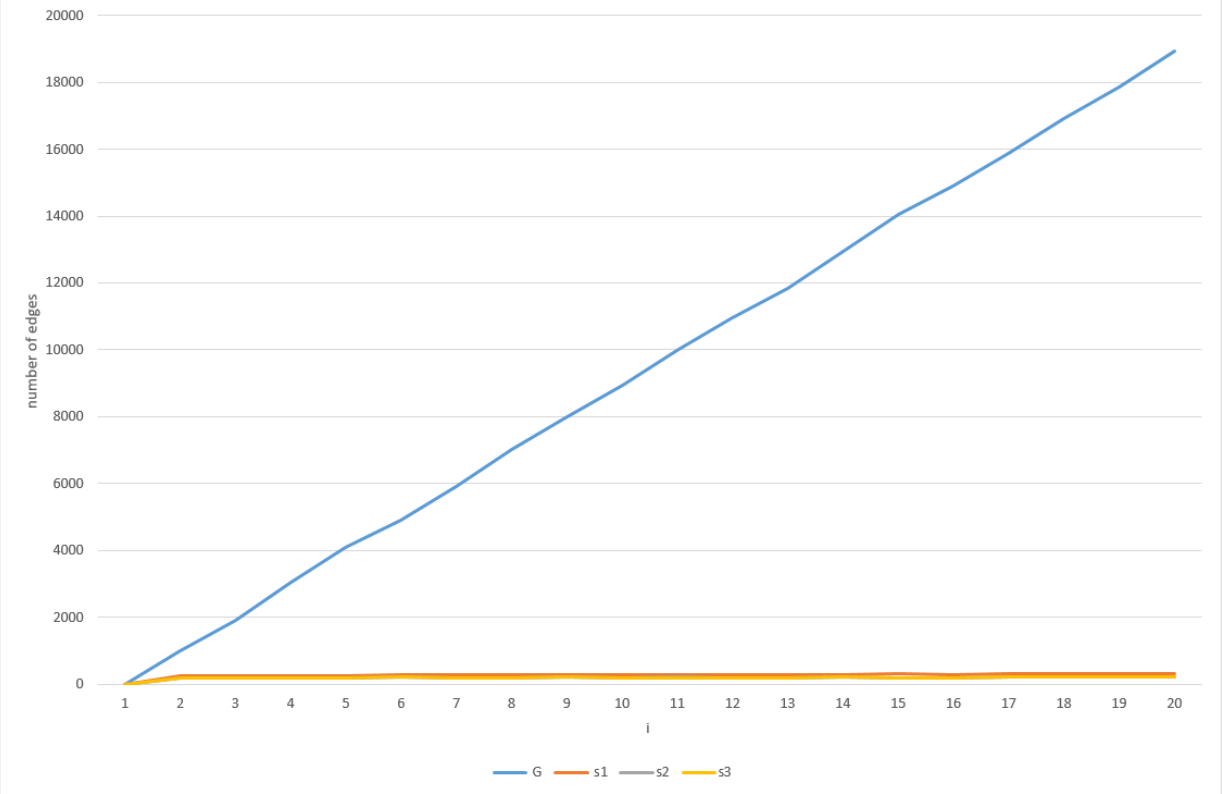
Please note that spanner s2 is very close to spanner s3, and that s1 is relatively far from them, and that all spanners are far away from their upper bound. Since t1<t2<t3 I assume that the bigger stretch factors will consolidate to the same sizes.

I wanted to see how the size of n (number of vertices) affects the results so I run this test another time, now with different parameters, so in the next run:
Sample size remained 20.
Number of vertices = 200.
$t1 = 2 \rightarrow s1[i]$ is a $3 - spanner$.
$t1 = 5 \rightarrow s2[i]$ is a $9 - spanner$.
$t1 = 9 \rightarrow s3[i]$ is a $17 - spanner$.

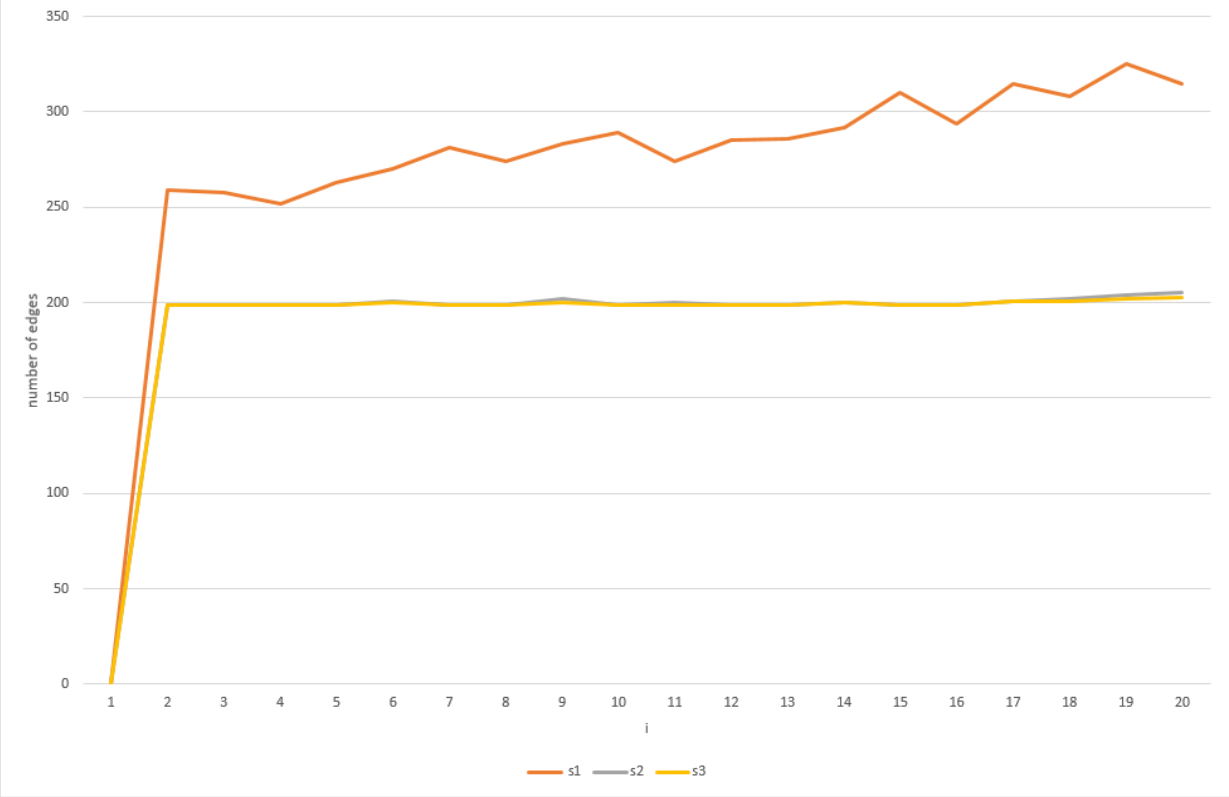
Here are the results:

1	G	s1	s2	s3
2	0	0	0	0
3	993	259	199	199
4	1924	258	199	199
5	3039	252	199	199
6	4087	263	199	199
7	4912	270	201	200
8	5920	281	199	199
9	7022	274	199	199
10	7987	283	202	200
11	8951	289	199	199
12	9979	274	200	199
13	10959	285	199	199
14	11835	286	199	199
15	12948	292	200	200
16	14038	310	199	199
17	14915	294	199	199
18	15864	315	201	201
19	16915	308	202	201
20	17875	325	204	202
21	18922	315	205	203

Just like before we will observe the in a more visualized way:



We can clearly see that the spanner's sizes are significantly lighter than the original graph.
So, from this experiment I conclude two things:
First, the algorithm's upper bound is a lot higher than the actual results.
Second, this experiment confirms my assumption that the higher stretch factors will consolidate to the same weights, since I increased the stretch factors and now they are almost the same. Let's have a look at them more closely:



As you can see, also here s2 and s3 are even closer than in the last experiment, and I believe this is due to the increasing of $t1, t2, t2$ and possibly the number of vertices.
Also, as printed in the command line:
The upper bound for stretch factor t1 = 2 is: 3000.
The upper bound for stretch factor t2 = 8 is: 400.
The upper bound for stretch factor t3 = 9 is: 400.
So the distance between the actual sizes of the spanner and the upper bound provided by the article grows as n, the number of vertices and t the stretch factor grow.

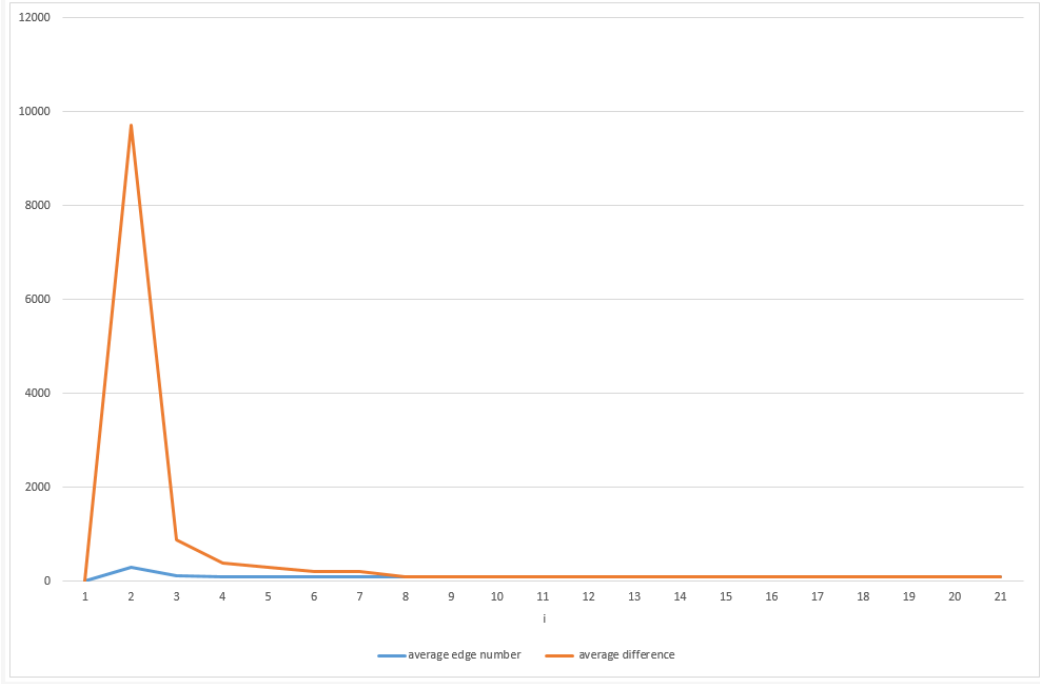
Average Differential Aspect Experiment:

function averageDifferentialAspectExperiment

In this test the virtue that is been examined is the average distance in the spanner, and the difference between it and the upper bound of the algorithm.
Let us denote d the difference between the theoretical upper bound of the article and the actual size of the produced spanner. So $d = n \cdot \left\lceil n^{\frac{1}{k}} \right\rceil - size(S)$ where S denotes the $(2k-1) - spnner\ of\ G$ which was produced by the test.
I used the following assigning:
Sample size = 40.
Number of vertices = 100.
p – probability for an edge to exist = 0.5.
Maximum stretch factor = 20.
The test generates sampleSize graphs with the p the probability for an edge to exist and generates for every $i \leq maximum\ stretch\ factor$ a $(2 \cdot i\ -1)spanner$ and examine the average size of the spanners and the difference between the upper bound to them.
The results are shown in the following chart:

1	average edge number	average difference
2	0	0
3	288.45	9711.55
4	125.175	874.825
5	105.475	394.525
6	100.8	299.2
7	99.825	200.175
8	99.2	200.8
9	99	101
10	99	101
11	99	101
12	99	101
13	99	101
14	99	101
15	99	101
16	99	101
17	99	101
18	99	101
19	99	101
20	99	101
21	99	101
22	99	101

Like in previous tests we would want to take a glance in a more visualized way:



As we can see, the average edges number and average difference behave in the same manner. Also, the average difference is becoming more close to a constant when i grows and > $\log(n)$.

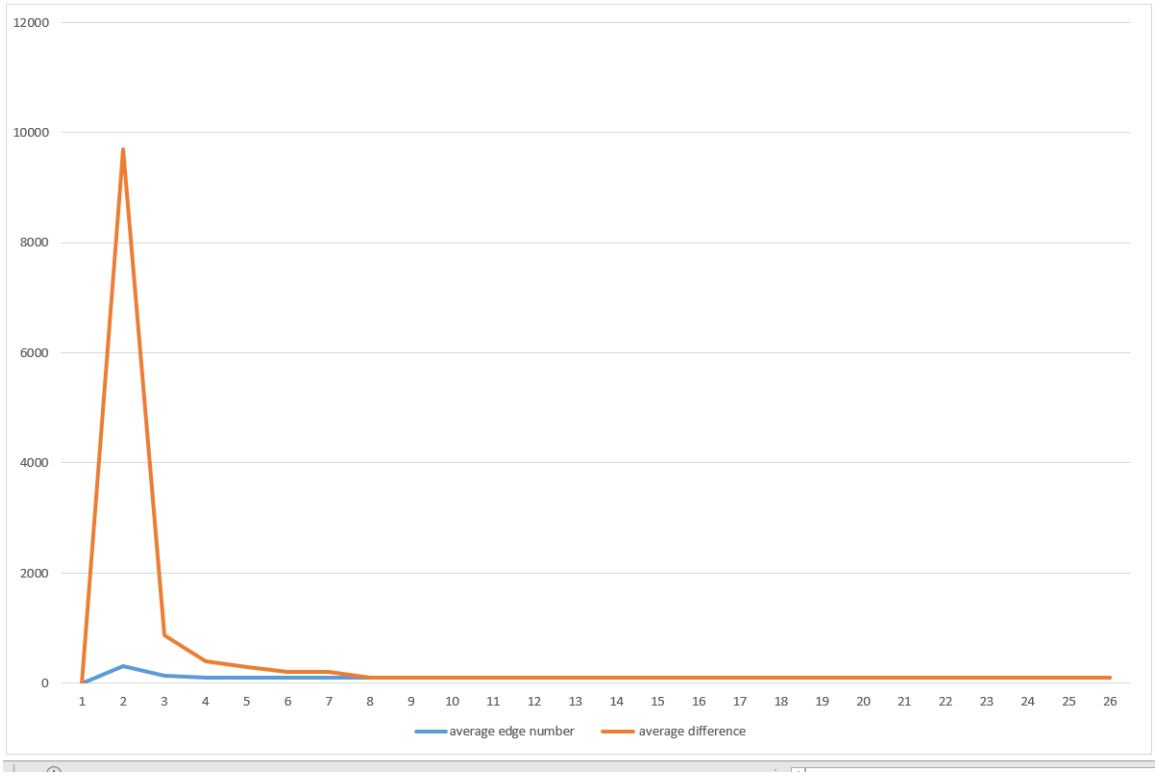
In order to confirm this, I ran another test with bigger number of vertices.
So, in order to get a bigger picture of how the spanners behave with more dense Graphs and more stretch factor that are supposed to give varied results I used this assignment in this next experiment:
Sample size = 60
Number of vertices = 100
p – the probability for an edge to exist = 0.75
Maximum stretch factor = 25

Thanks to the upper bound on the stretch factor I know that increasing the maxStretchFactor will mean setting the upper bound to $n \cdot \left\lceil n^{\frac{1}{i}} \right\rceil \approx 2n$.

The results are:

1	average edge number	average difference
2	0	0
3	305.8833333	9694.116667
4	127.3833333	872.6166667
5	106.9833333	393.0166667
6	101.9	298.1
7	99.91666667	200.0833333
8	99.3	200.7
9	99.11666667	100.8833333
10	99.05	100.95
11	99	101
12	99	101
13	99	101
14	99	101
15	99	101
16	99	101
17	99	101
18	99	101
19	99	101
20	99	101
21	99	101
22	99	101
23	99	101
24	99	101
25	99	101
26	99	101
27	99	101

We can clearly see that at a certain point the average edge number and average difference are constants:



Also, please notice that the lower bound is $\approx size\left(MST(G)\right) = n - 1$.

Weights Aspect Expirement: function weightsAspectExpirement

In this experiments the criteria been measured is weight.
For an n vertices graph G and it's $(2k-1)-spanner\ G'$ I compared actual $Weight(G')$ and the upper bound for it which is $weight(G') < weight\big(MST(G)\big)(1+\frac{n}{2k})$.

In the weightsAspectExpirement.csv file there are the upper bound and the actual weight average at the spanners for every stretch factor up until maxStretchFactor.
Also denoted (printed on the screen) are Average $weight(G)$ and Average $weight(MST(G))$.

The function receives a sample size, number of vertices , p the probability for an edge to exist and a max stretch factor.

I ran 3 tests: all of them with the same sample size, number of vertices and maximum stretch factor , but the all have different p's.

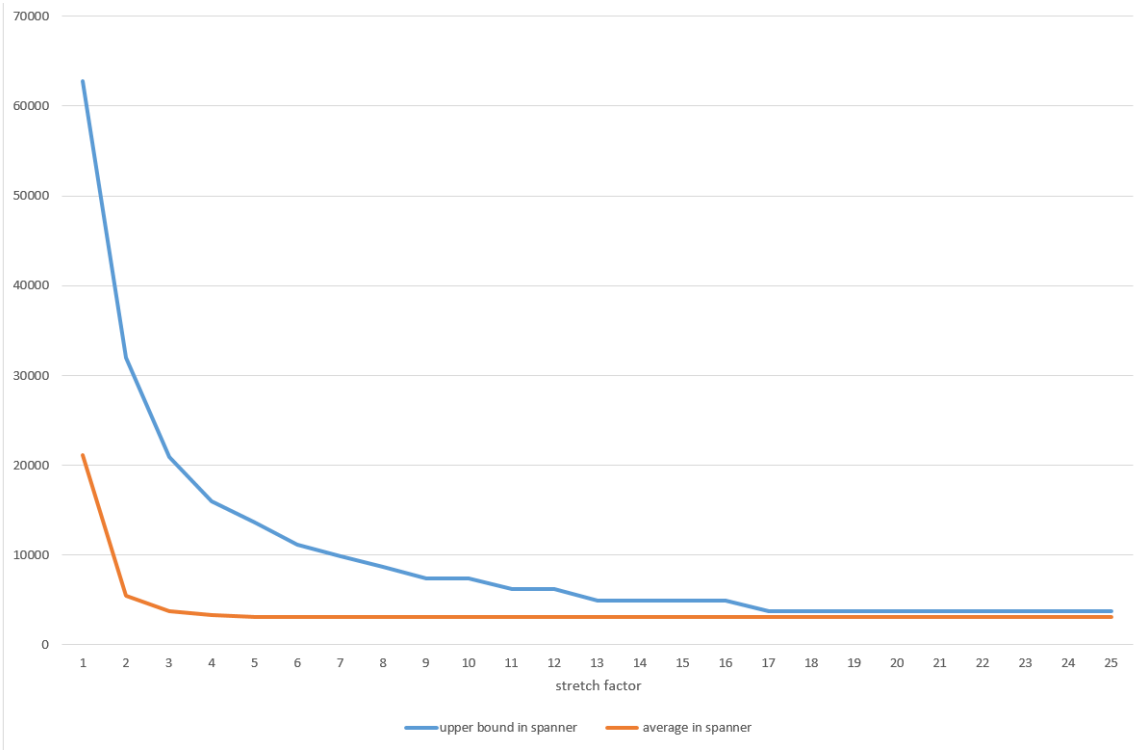
The results are:

Test #1:

Average weight in G in the sample is 24883.6
Average weight in $MST(G)$ in the sample is 1231.28
Data chart:

	upper bound in spanner	average ir
2	62795.28	21159.72
3	32013.28	5433.64
4	20931.76	3686.48
5	16006.64	3241.68
6	13544.08	3111.04
7	11081.52	3058.52
8	9850.24	3040.12
9	8618.96	3028.84
10	7387.68	3028.84
11	7387.68	3028.84
12	6156.4	3028.84
13	6156.4	3028.84
14	4925.12	3028.84
15	4925.12	3028.84
16	4925.12	3028.84
17	4925.12	3028.84
18	3693.84	3028.84
19	3693.84	3028.84
20	3693.84	3028.84
21	3693.84	3028.84
22	3693.84	3028.84
23	3693.84	3028.84
24	3693.84	3028.84
25	3693.84	3028.84
26	3693.84	3028.84

Graphical display:

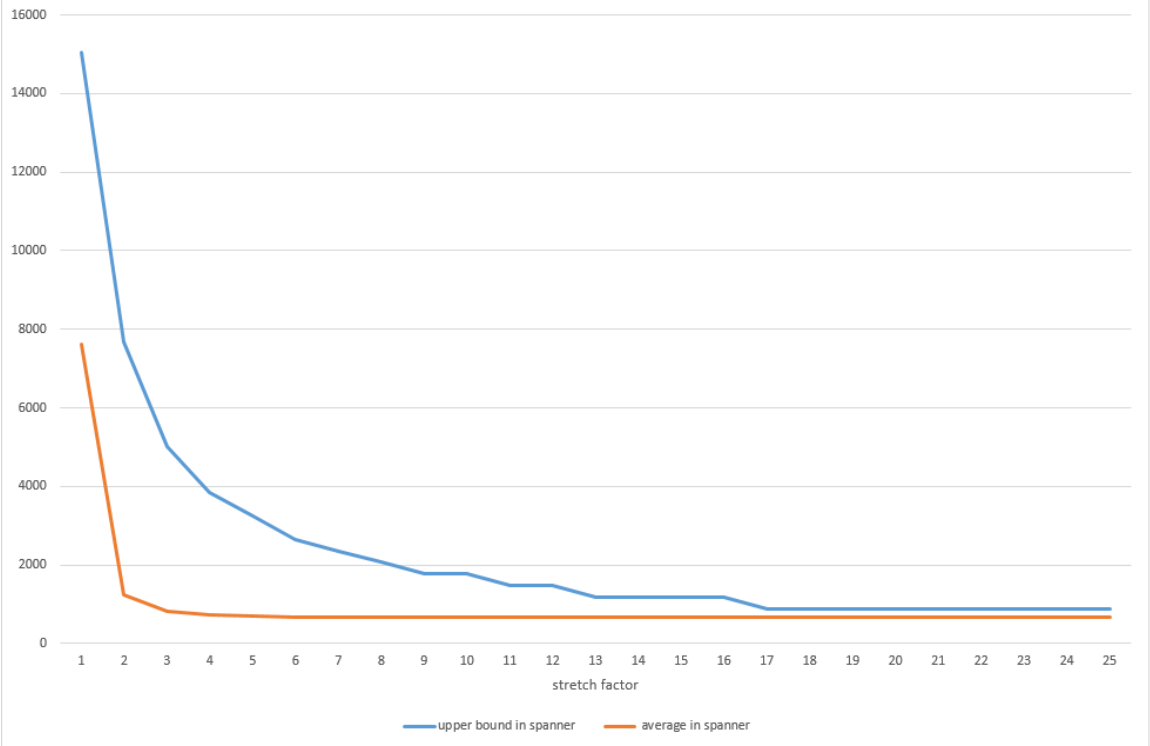


Test #2:

Average weight in G in the sample is 125635.4
Average weight in $MST(G)$ in the sample is 294.8
Data chart:

	upper bound in spanner	average in spanner
2	15034.8	7608.92
3	7664.8	1235.12
4	5011.6	824.2
5	3832.4	718.08
6	3242.8	686.32
7	2653.2	672.44
8	2358.4	667.4
9	2063.6	664.28
10	1768.8	663.32
11	1768.8	663.04
12	1474	663.04
13	1474	663.04
14	1179.2	663.04
15	1179.2	663.04
16	1179.2	663.04
17	1179.2	663.04
18	884.4	663.04
19	884.4	663.04
20	884.4	663.04
21	884.4	663.04
22	884.4	663.04
23	884.4	663.04
24	884.4	663.04
25	884.4	663.04
26	884.4	663.04

Graphical display:

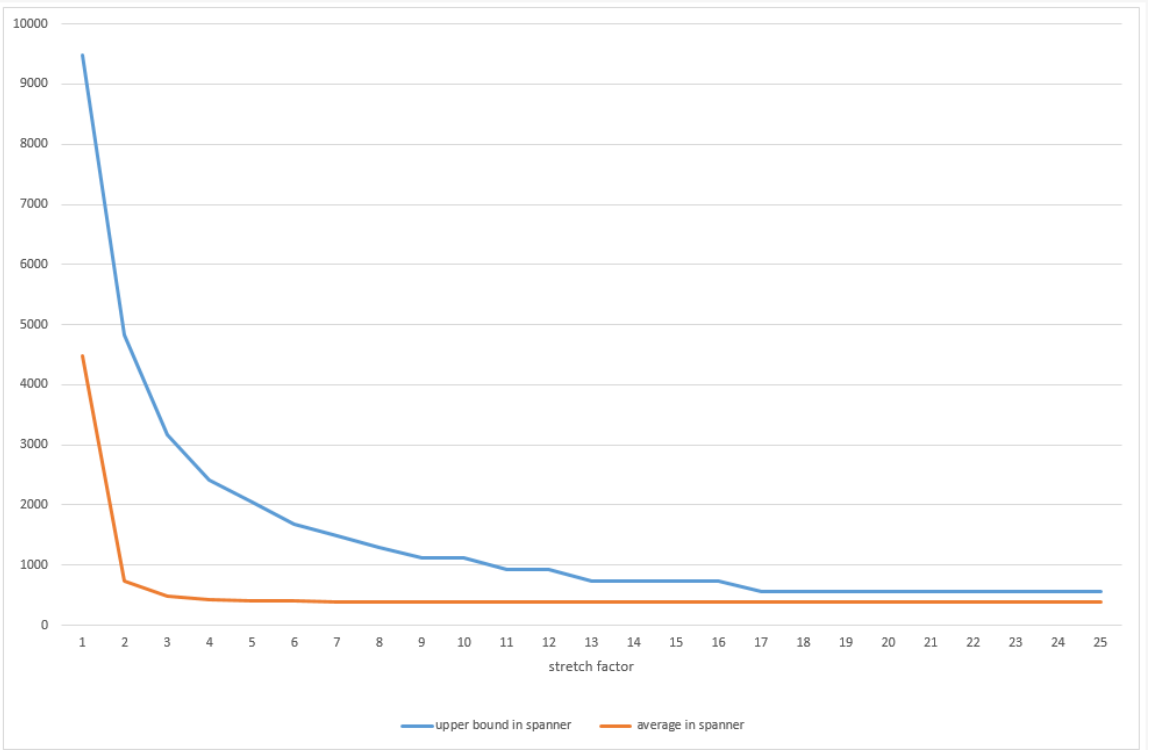


Test #3:

Average weight in G in the sample is 224067.76
Average weight in $MST(G)$ in the sample is 185.96
Data chart:

	upper bound in spanner	average in spanner
2	9483.96	4482.2
3	4834.96	725.6
4	3161.32	487.92
5	2417.48	427.08
6	2045.56	406.32
7	1673.64	398.48
8	1487.68	394.88
9	1301.72	393.6
10	1115.76	393.04
11	1115.76	393.04
12	929.8	393.04
13	929.8	393.04
14	743.84	393.04
15	743.84	393.04
16	743.84	393.04
17	743.84	393.04
18	557.88	393.04
19	557.88	393.04
20	557.88	393.04
21	557.88	393.04
22	557.88	393.04
23	557.88	393.04
24	557.88	393.04
25	557.88	393.04
26	557.88	393.04

Graphical display:



You can see from the graphical display that the average weight in the spanners trend behaves the same, regardless of p, i.e. the density of the graph will not affect it's average weight.
From the data charts we see that the denser the original graphs are, the lightest the average weights gets.

Conclusions And Directions For Future Work:

From the edges aspect experiments I have learned that $size((2k-1)-spanner(G))$ is far higher than the actual size. Therefore, for future theoretical work I would suggest finding a lower upper bound for it.

From the average differential aspect experiment I have learned that the density of G does not affect the average size, and that as stretch factor grows the spanner's differential decreases and eventually become a constant. Then, via facts that was proved in the article and some math I deduced a lower bound to $size((2t-1)-spanner(G))$ which is $MST(G)=n-1$.

From the weights aspect experiment I have learned that in terms of weight, or rather average weight of a spanner $average-weight((2t-1)-spanner\ (G) \underset{t\to\infty}{\longrightarrow} 2\cdot MST(G)$. of course this is exaggeration and t doesn't need to be a really large number.

For future work I would have run weights aspect experiment more times, with various maximum stretch factors and see how different factors affect the average weight of the spanners.

Bibiliography:

"On Sparse Spanners Of Weighted Graphs" by Ingo Althofer, Gautam Das, David Dobkin, Deborah Joseph and Jose Soares, 1993.