**On sparse spanners of weighted graphs**

I. Alth\"ofer, G. Das, D. Dobkin, D. Joseph, and J. Soares

Bar Hadad 318182987

Maya Israeli 313196404

**Table of contents**

Introduction 2

The algorithm 3

Implementation 4

Experiment 1 5

Experiment 2 7

Experiment 3 9

Experiment 4 11

Experiment 5 14

Experiment 6 16

Conclusion 19

Future work and bibliography 20

**Introduction**

The algorithm we have chosen to implement was taken from the paper "On sparse spanners of weighted graphs" by I. Alth\"ofer, G. Das, D. Dobkin, D. Joseph, and J. Soares.

The algorithm gets a graph G = (V, E) and a stretch factor T and returns a T-spanner G'=(V,E') for G.

Spanner definition:

Let G = (V, E) be a connected n-vertex graph with arbitrary positive edge weights. A sub-graph G' = (V, E') is a T-spanner of G if between each pair of vertices v1, v2 the distance between v1 and v2 in G' is at most T-times longer than the distance in G. T is called the stretch factor associated with T.

Sparsity can be measured according to the weight of the total edges and the number of edges.

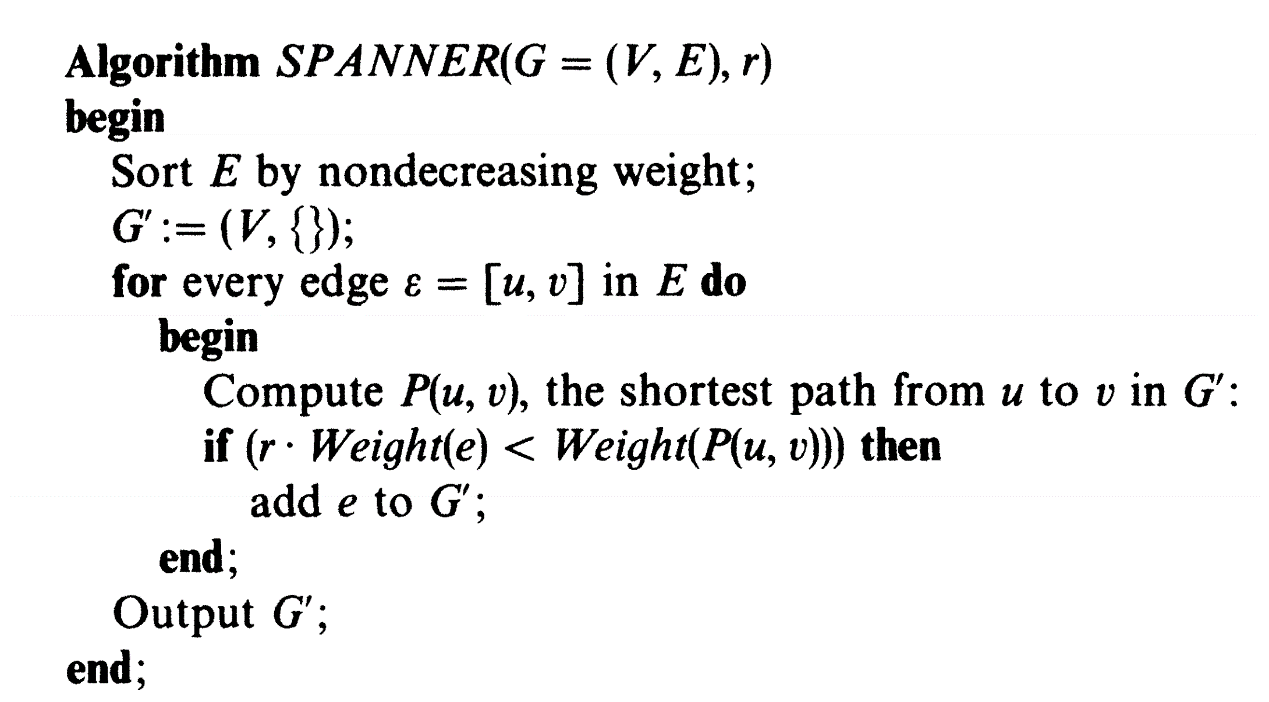
One of the examples for application of spanners is in the field of biology, through the process of reconstructing phylogenetic trees. Another example is in distributed systems and communication networks, in which the spanners appears to be the underlying graph structure.

The paper describes the construction of the sparse spanners for general graphs and presents several bounds for the results. In addition, the paper mentions spanners for Euclidean graphs.

**The algorithm itself:**

Pseudo code:

G is a connected graph \*



1. Let G = (V, E) graph and r = stretch factor.
2. The algorithm sorts the edges (E) by nondecreasing weight.
3. The algorithm builds Empty Graph G' with the same vertices as G.
4. For each edge e = {u, v} in the original graph edges (E): The algorithm finds the weight – W, of the lightest path from u to v in G’ (the new graph we are currently building). Then it checks if (r \* weight of e < W), if so – the algorithm adds the edge e to G'.
5. The algorithm returns G', the T spanner we build.

**Implementation of the Algorithm:**

We choose to implement our project using Python.

We used "NetworkX" package, which is a recommended python package for graphs.

We used "NetworkX" to build and maintain the weighted graph, to sort to edges by weight in non-decreasing order and to find the shortest path (Weight) using Dijkstra.

Functions we used:

* G.Add\_edge(u, v, w) function which add the edge {u, v} with weight w to G.
* H.add\_nodes\_from(g) function which add the nodes of graph g to graph H.
* Sorted – function which sort the graph edges by weights in non-decreasing order.
* Has\_path (H, u, v) – function which check if there is a path from u to v in H.
* Dijkestra\_path\_length(H, u, v) – function which calculate the weight of the minimal path from u to v.

**Experiment 1**

In this experiment we want to test what happens with constant weight on the edges of the input graph and stretch factor 1. We would like to compare the number of edges in the spanner graph with the number of the edges in the input graph.

According to the paper and the theory, we expect that in this case the graph will remain the same.

In order to test this case, we ran the algorithm on 150 random graphs with the following data:

1. Stretch factor = 1

2. Constant weight on each edge = 1

3. Probability for appearance of each edge: 0.5

4. Number of vertices = 30

The results of the experiment are displayed in the following chart:



As we expected, all the cases show that the number of edges in the original graph didn't changed in the spanner graph.

**Experiment conclusion:**

When we examine graphs with constant weight on the edges of the input graph and stretch factor 1, we get spanner graph with the same number of edges as the original graph.

**Experiment 2**

In this experiment we want to check the correlation between the stretch factor and the number of edges of the spanner graph.

In order to test this case, we ran the algorithm on 100 random graphs with the following data:

1. Stretch factor = 1,3,5,7

2. Constant weight on each edge = 1

3. Probability for appearance of each edge: 0.5

4. Number of vertices = 30

The results of the experiment are displayed in the following chart:



**Experiment conclusion:**

We can see from the chart that for each graph, as the stretch factor increases - the number of edges in the spanner graph decreases.

**Experiment 3**

In this experiment we want to test what happens with different weights on the edges of the input graph and a stretch factor equal to 1. We would like to compare the number of edges in the spanner graph with the number of the edges in the input graph.

In order to test this case, we ran the algorithm on 100 random graphs with the following data:

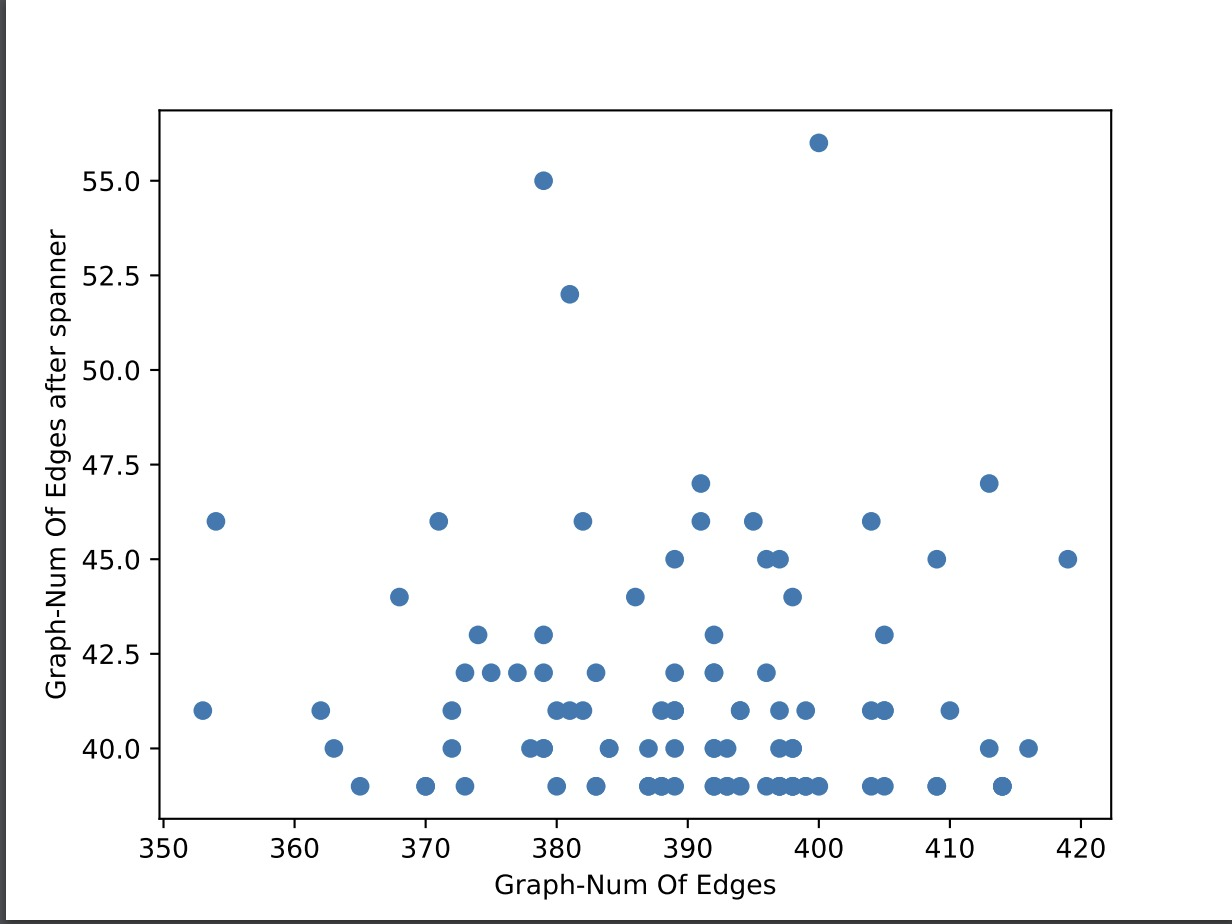
1. Stretch factor = 1

2. Different weight on each edge

3. Probability for appearance of each edge :0.5

4. Number of vertices = 40

The results of the experiment are displayed in the following chart:



**Experiment conclusion:**

When we examine graphs with different weights on the edges and a stretch factor equal to 1, we get spanner graph with fewer edges in comparison to the original graph. This, in contrast to the first experiment (with constant edge weights) in which the spanner graph number of edges stay the same.

**Experiment 4**

In this experiment we want to test the correlation between the weight of the input graph and the spanner graph.

In order to test this case, we ran the algorithm on 20 random graphs with same and different densities and the following data:

1. Stretch factor = 1

2. Different random positive weight on each edge (between 1 to 50)

3. Probability for appearance of each edge:

First, we test with fixed probability = 0.5, then we try different probabilities (for appearance of each edge) in order to change the graph density: changes from 0.1 to 1 in increments of 0.1.

4. Number of vertices = 100

The results of this experiment are displayed in the following charts:

For fixed density = 0.5

For different densities, 0.1 to 1:



In order to plot the graph, we used “polyfit” function which gives us the best polynom approximation of all the values we get when testing 20 graphs and different probabilities.

**Experiment conclusion:**

In the first experiment with the fixed density, we can see from the chart that the total weight of the spanner graph decreases drastically, comparing it to the input graph.

In the second experiment with the different densities, we can see that while the total weight of the original graph and the density increases together, the total weight of the spanner graph decreases when the density increases.

**Experiment 5**

In this experiment we extend the last experiment (experiment 4) and examine the stretch factor's impact.

In order to test this case, we ran the algorithm on 20 random graphs with the following data:

1. Stretch factor = 1,3,5,7

2. Different random positive weight on each edge (from 1 to 50)

3. Probability for appearance of each edge: 0.5

4. Number of vertices = 100

The results of the expanded experiment are displayed in the following chart:



**Experiment conclusion:**

We can see from the chart that the total weight of the spanner graph decreases drastically when the stretch factor increases.

**Experiment 6**

In this experiment we want to check the correlation between the graph density and the number of edges in the spanner graph.

In order to test this case, we ran the algorithm on 20 random graphs, and change probability for appearance of each edge from 0.1 to 1 in increments of 0.1 (in order to change the graph density).

We ran the experiment with the following data:

1. Stretch factor = 5

2. Different random positive weight on each edge (from 1 to 50).

3. Probability for appearance of each edge: change from 0.1 to 1 in

increments of 0.1

4. Number of vertices = 100

****

****

**Experiment conclusion:**

We can see that we get only a minor effect on the number of edges in the spanner graph. As we can see from the graphs, while in the original graph there is a drastic change in the number of edges (jump from ~500 to ~4500), in the spanner graph there is almost no effect at all (jump from ~104 to ~114 only).

**Conclusions**:

After we have done the experiments, we have got to some conclusions.

1. For graphs with constant weight on the edges and stretch factor 1, we get spanner graph with the same number of edges as the original graph.
2. For graphs with constant weight on the edges and different stretch factor, we can see that as the stretch factor increasing- the number of edges in the spanner graph decreasing.
3. For graphs with different weights on the edges and a stretch factor equal to 1, we get spanner graph with fewer edges compare to the original graph.
4. For graphs with stretch factor 1 and different weights on the edges we can see that the total weight of the spanner graph decreases drastically, comparing it to the input graph.
5. For graphs with different stretch factor and different weight, we can see that the total weight of the spanner graph decreases drastically when the stretch factor increasing. In addition, we can see that for each graph there is some limit in which the stretch factor has no impact at all.
6. For graphs with fixed stretch factor (equal to 5), different random positive weight on each edge and different densities, we can see that we get only a minor effect on the number of edges in the spanner graph.

**Future Work:**

For future work we would like to investigate more about the influence of the stretch factor (on the weights, number of edges, etc.).

Moreover, we would like to test our experiments with much bigger graphs, much bigger stretch factor, more different probabilities for edges and much more graphs in order to get more accurate results.

In order to do so, we will have to work on stronger computers. This will give us the ability to explore better the effect on spanner graph when there are more graphs, and when the graph and the stretch factor are getting much bigger.

**Bibliography**

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