We are solving the problem of the bounds on expected runtime of the evolutionary algorithm that generates test for Dijkstra algorithm such that this algorithm will relax all edges during this test.

1 First steps

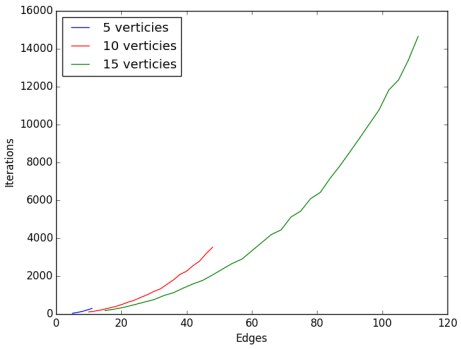
1.1 Experiments

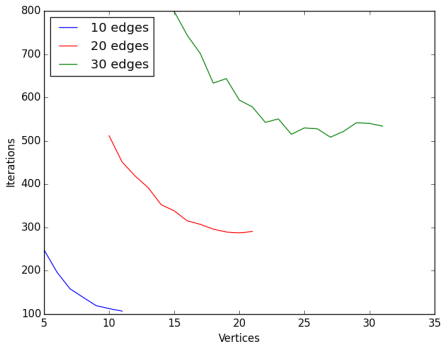
First we made some experiments that calculated:

- average number of algorithm iterations depending on the number of edges with the fixed number of vertices;
- average number of algorithm iterations depending on the number of vertices with the fixed number of edges;
- average number of leaves in the generated graph during the algorihm runtime;
- average number of vertices in the connected component;

1.2 Algorithm iterations

We made 200 algorithm runs for each number of edges and vertices and counted the number of iterations. For fixed numbers we made experiments with 5, 10 and 15 vertices and 10, 20 and 30 edges. Here are results:





We can see that expected runtime depends straightly on the number of edges and inversely depends on the number of vertices.

1.3 First analysis

The algorithm run can be divided into two phases:

- the phase of growth, when all the vertices become reacheble from the start one;
- the phase of relaxing all edges that left unrelaxed in te first phase.

The analysis of the first phase is comlex because of two facts:

– in some cases size of connected component decreases by one;
– in some cases connected component can be extended by more than one vertex.

But we have an assumption that the first fact doesn't have much influence on the upper bound of the first phase because it's qute rare situation. The second fact only decreases the upper bound on runtime of the first phase. So we can make simple upper bound on the expectation of runtime of the first phase.

The probability of the extention of the connected component by one vertex is not less then multiplication of probabilities of taking the edge not from the spanning tree on the probability of putting it's new start into the connected component and it's end outside the connected component:

$$P_{inc} \geq \frac{E-i+1}{E} \cdot \frac{i(V-i)}{V^2},$$

where E – number of edges, V – number of vertices in the graph, i – number of vertices in the current connected component.

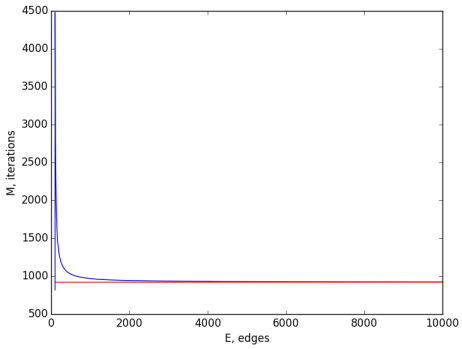
So the expectation of the extension of the connected component by one vertex is following:

$$E_{inc} = P_{inc}^{-1} = \frac{EV^2}{(E-i+1)i(V-i)}$$

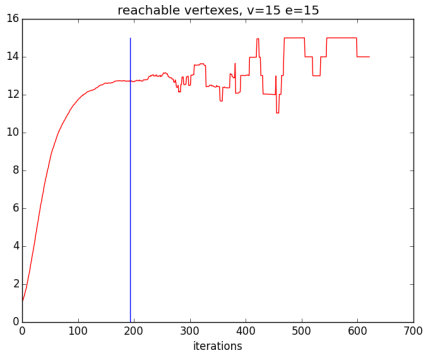
The expected runtime of the first phase can be calculated as the sum of expected times of extension by one vertex for every size of the connected component:

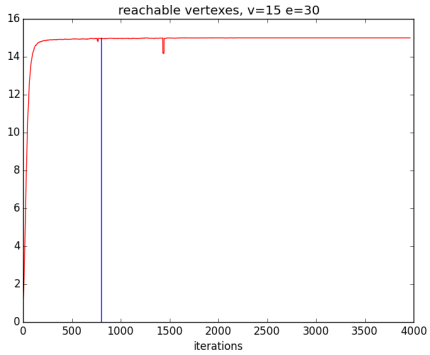
$$\begin{aligned}
E_{i=V} &= \sum_{i=1}^{V-1} \frac{EV^2}{(E-i+1)i(V-i)} = \sum_{i=1}^{V-1} \frac{EV}{E-i+1}\left(\frac{1}{V-i}+\frac{1}{i}\right) = \\
&= \sum_{i=1}^{V-1}\left(\frac{EV}{E-V+1}\left(\frac{1}{V-i}-\frac{1}{E-i+1}\right)+\frac{EV}{E+1}\left(\frac{1}{i}+\frac{1}{E-i+1}\right)\right) = \\
&= EV\left(\frac{1}{E-V+1}+\frac{1}{E+1}\right)\sum_{i=1}^{V-1}\frac{1}{i} + EV\left(\frac{1}{E+1}-\frac{1}{E-V+1}\right)\sum_{i=1}^{V-1}\frac{1}{E-i+1} \approx \\
&\approx \frac{V(2E-V)}{E-V}\ln V - \frac{V^2}{E-V}(\ln E - \ln(E-V))
\end{aligned}$$

Here are the graphic of this functuion for $V = 500$. The red line is $2V \ln V$ constant that is asimptote of this function for large number of edges:



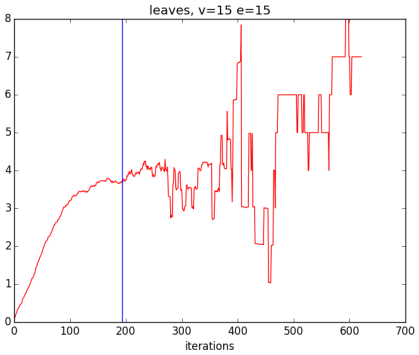
We can see that with number of edges that is only little greater than number of vertices the expected runtime of the first phase is really largel. But experiments showed that in this case algorithm finds optimum before the end of the first phase. In the next graphics we can see the average number of vertices in the connected component for each iteration. The vertical line shows the average number of algorithm iterations.

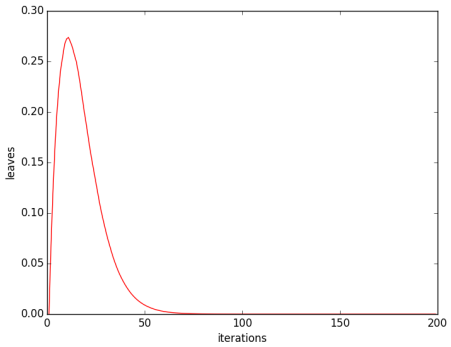




So we can see that for small number of edges the expected runtime of the algorithm is less then expected runtime of the first phase and this case should be considered separatly.

Also to prove the upper bound we need to prove that the case when the size of connected component decreases is rare enough. But there are too many cases when the size of the component can decrease and counting their probabilities is a complex problem.


leaves, v=15 e=15
0
1
2
3
4
5
6
7
8
0
100
200
300
400
500
600
700
iterations


0.30
0.25
0.20
0.15
0.10
0.05
0.00
leaves
0
50
100
150
200
iterations

2