Instituto Superior Técnico – Universidade de Lisboa

2016 / 2017

Inter-domain Routing

Network Algorithms and Performance

Group nº 13:

* Dominik Gizynski, 85736
* João Caetano, 75218

# Introduction

In this project the objective is to present statistics related to the number of hops needed to reach a node in an internet graph using the best route and the shortest path.

In the first script, given an internet graph, the user has the possibility to choose the destination and the starting point, then the best type of route from the starting AS to the destination AS is outputted.

The second script is similar to the first one with the addition that outputs also the number of hops from the starting AS to the destination one.

The third and last script runs the best type of route algorithm and a shortest path algorithm from all the nodes to all the nodes and outputs two graphs showing a cumulative distribution function. The first graph is for the number of hops in the best routes and the second in shortest paths.

# Algorithms Explanation and Pseudo-codes

## Elect\_route

This algorithm can be understood as a Dijkstra with a FIFO queue.

The input of this algorithm is an internet graph, the destination and two already initialized dictionaries for the route types and number of hops of each AS node. The route types are initialized as Null (bullet) and the number of hops as infinite

The route type for the destination is set as costumer, the number of hops as 0 and the queue is initialized with the destination as its only member.

While the queue has at least one element the algorithm will run.

The first element of the queue is retrieved, lets call it AS, and its neighbours explored. For each neighbour the path that is sent by the AS is calculated. If this path is a *better* *path* than the one the neighbour has then the path and the number of hops are updated and the neighbour inserted in the queue. By *better* one understands that the type of path is better (costumer < peer < provider < bullet) and if the type of path is the same a better path is the one with less hops.

The pseudocode for the main algorithm is presented next:

*routes[all nodes] = Null // bullet*

*hops[all nodes] = infinite*

*elect\_root(graph, destination, routes[], hops[]):*

*queue = FIFO()*

*routes[destination] = costumer*

*hopes[destination] = 0*

*Q.insert(destination)*

*while(size of queue > 0):*

*AS = first element of queue*

*for each neighbour of AS:*

*new\_route = route exported from AS to the neighbour // done by function get\_route*

*if new\_route better than routes[neighbour]: // done by function better\_route*

*routes[neighbor] = new\_route*

*hopes[neighbour] = hopes[AS] + 1*

*Q.insert(neighbor)*

*return*

The pseudocode for the function that returns the exported path from one node to other according to the nodes relation is presented next:

*get\_route(AS, neighbor, routes[]):*

*relation = relation between AS and neighbour nodes*

*exported\_route = routes[AS] // route to be exported*

*route = Null*

*if relation == 1: // export to provider*

*if exported\_route == costumer:*

*route = costumer*

*else if relation == 2: // export to peer*

*if exported\_route == costumer*

*route = peer*

*else: // export to costumer*

*if exported\_route != Null: // exported route = costumer or peer or provider*

*route = provider*

*return route*

The pseudocode for the function that returns True if the new calculated path is better that the one previously saved in the node is presented next:

*better\_route(neighbour, AS, new\_route, routes[], hops[]):*

*answer = False*

*route = routes[neighbour] // current neighbour route type*

*if (route == new\_route) and (hops[AS] + 1 < h[neighbor]) and (route != Null): // if node gets a new route of the same type as the one he has but with less hops*

*answer = True*

*else if route == peer and new\_route == costumer:*

*answer = True*

*else if route == provider and (new\_route == costumer or new\_route == peer):*

*answer = True*

*else if route == Null and new\_route != Null:*

*answer = True*

*return answer*

For the algorithm in the first part of the project that only concerns with route types and not number of hops the differences from the pseudocode above is that in the main algorithm the dictionary hops is not introduced and in the better\_route function the condition with the same root type is not checked. This algorithm has an asymptotic complexity of O(m) where m is he number of links between AS nodes.

The algorithm in the second part is heavier because it expands more node in the queue (the ones with the same path but lesser number of hops). For this algorithm, the asymptotic complexity is O(nm) where n is the number of nodes because each route type can be updated (n-1) times in the worst case.

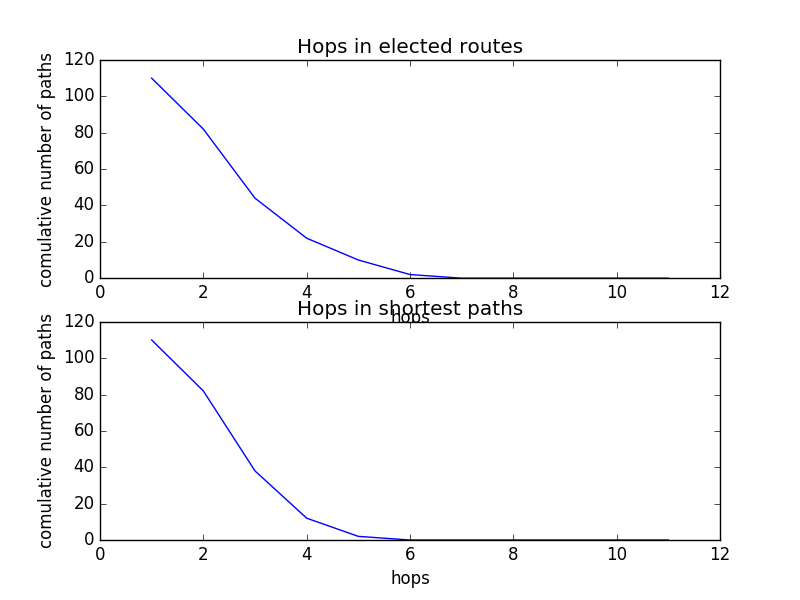
## Third script

The third script runs the above algorithm for every node as destination so the complexity increases to O(n2m).

The algorithm used for the shortest paths was the Dijkstra that also runs for every node and uses a binary heap as a queue so the resulting complexity is O(n2log(m)).

In the end two graphs representing the cumulative distribution function for the number of hops in the best paths and in the shortest paths are shown.

The graphs for the network given as example in the assignment are presented next:



The y axis indicates the number of paths that uses at least a specific number of hops to reach the destination while the x axis shows that specific number of hops.

# Short Discussion

The type of routes and number of hops was tested for numerous examples and proved to give the right answer always. Also, it is easily understood that the algorithm converges to the optimal solution.

In the graph above we can see that the curvature in the shortest paths graph is deeper than in the elected routes one and that the curve is more prolongated in the elected routes. This is what was expected because the best route is not always the shortest path due to the Gao-Rexford routing policies.