REPORT KSHELL DECOMPOSITION

Pankaj Verma, 2015CSB1022
Soumyadeep Roy, 2015CSB1035

December 1, 2018

1 INTRODUCTION

K-Shell decomposition helps in finding out the shell numbers of various nodes in a particular graph. This is particularly helpful in finding out the core of any graph. With the help of this knowledge we are working on ways to reach the core of the graph from any random point with only local knowledge to rely upon.

2 Literature Review

2.1 The H-index of a network node and its relation to degree and coreness

This paper discusses how degree and coreness are related with each other wih the help of a property called H-index. H-index is the property of a node that is difined by the relationship given below :

Let $G(V, E)$ denote a graph G with V vertices and E edges making it up. Now choosing any particular node let the degree of the node i be denoted by k_i. Then the zero-order H-index of node i is $h_i^{(0)} = k_i$, i.e is the degree of the node. Let the $(n-1)$-order H-index of neighbours of i be denoted as $h_{j_1}^{(n-1)}, h_{j_2}^{(n-1)}, h_{j_3}^{(n-1)}, ..., h_{j_{k_i}}^{(n-1)}$. Now the n-order H-index can be calculated as :

$$h_i^{(n)} = \mathcal{H}(h_{j_1}^{(n-1)}, h_{j_2}^{(n-1)}, h_{j_3}^{(n-1)}, ..., h_{j_{k_i}}^{(n-1)}).$$

NOTE : Here $y = \mathcal{H}(x_1, x_2, x_3, ..., x_n)$ signifies that y is the maximum integer such that there exist atleast y elements in $(x_1, x_2, x_3, ..., x_n)$ each of which is no less than y.

This paper helps in establishing the fact that as the order of H-index value increases to infinity, the value of the H-ndex converges to the coreness value of the node.

The paper also provides a method for doing asynchronous updating in a dynamic graph as such graphs are subject to rapid changes in the network. By traditional approach the addition of a single link in such network will need recalculatation of H-index sequence for entire network. However Asynchronus updating can stil gurantee a convergence to coreness.

2.1.1 Theorem : Convergence of H-index to coreness

For every node $i \ \forall \ v \in V$ of an undirected simple network $G(V, E)$, its $H - index$ sequence $h_i^{(0)}, h_i^{(1)}, h_i^{(2)}, h_i^{(3)}, ...$ will converge to the coreness of node i,

$$c_i = \lim_{n->\infty} h_i^{(n)}$$

Proof: We have $h_i^{(n)} \geq 0$ and $h_i^{(0)} \geq h_i^{(1)}$. After applying mathematical induction we get $h_i^{(n)} \geq h_i^{(n+1)}$. Since $h_i^{(0)}, h_i^{(1)}, h_i^{(2)}, ...$ is a continuously decreasing sequence and all elements are nonnegetive, therefore this sequence has got a lower bound. First we consider, if $G^{'}(V^{'}, E^{'}) \subset G(V, E)$ for any node $i^{'}$ and any integer $n \geq 0$, $h_{i,G}^{(n)} \geq h_{i,G'}^{(n)}$. Second we set k_{min} as the minimal degree of G. then we for any node i and any integer $n \geq 0$, $h_i^{(n)} \geq k_{min}$. clearly it is valid for $n = 0$. so we applied mathamatical induction and it is hence proved. If we set $G^{'}$ the c_i core of G. here we can see $G^{'} \subset G$ and in $G^{'}$, $k_{min} \geq c_i$, therefore $h_i^{\infty} \geq k_{min} \geq c_i$.

we denote $G^{"}(V^{"}, E^{"})$ the induced subgraph containing nodes j s.t. $h_j^{\infty} \geq h_i^{\infty}$. node i itself belong to $G^{"}$.In G for any node $j \in V^{"} h_j^{\infty} > h_i^{\infty}$. there at least h_i^{∞} neighbours of j node with h^{∞} value no less than h_i^{∞}.Hence $G^{"}$'s degree of node are no less than h_i^{∞}, so $G^{"}$ is a subgraph of G's h_i^{∞}-core and it gives us $c_i \geq h_i^{\infty}$ where c_i is the coreness of i. In the end, from above two inequality we conclude that therom is proved.

Using the above theorem and defination of H-index, the paper implemented synchronous updating in eight reperesentive real network. Among these two are soial network (Facebook and Sex), two collaboration network(Jazz and NS), one communiction network (Email), one information network (PB), one transportation network (USAir) and one technological network(Router).

The paper concludes that the sequence of H-indices quickly converges to the coreness. Apart from degree, H-index and coreness, all intermediate states $h^{(2)}, h^{(3)}, h^{(4)}, ...,$ can also be considered as centrality measures.

2.1.2 Theorem : Asynchronous updating

Given an undirected simple network $G(V, E)$ for every node $j \in V$, we define $g_j = k_j$. In each iteration of the asynchronus updating process, a node i is randomly selected and if g value updated, that is,

$$\mathcal{H}(g_{j_1}, g_{j_2}, g_{j_3}, ..., g_{j_{k_i}}) - > g_i$$

Where $j_1, j_2, j_3, ..., j_{k_i}$ are the neighbouring nodes of i. if $|V|$ is finite, this updating process will reach a steady state $(g_1^\infty, g_2^\infty, g_3^\infty, ..., g_{|V|}^\infty,)$ after a finite number of iterations such that the updating at any node will not changes its g vlaue, namely,

$$\forall\, i \in V, g_i^\infty = \mathcal{H}(g_{j_1}^\infty, g_{j_2}^\infty, g_{j_3}^\infty, ..., g_{j_{k_i}}^\infty).$$

In the steady state, for every node i we have $g_i^\infty = c_i$.

Proof: Initially we have time t=0 and for every node j, $g_j^{(0)} = k_j$. At each time step randomly select a node and perform the $\mathcal{H}$ operator on it. When $t > 0$ node i selected and $g_i^{(t)} = \mathcal{H}(g_{j_1}, g_{j_2}, g_{j_3}, ..., g_{j_{k_i}})$. First prove that if any node $j \in V$ selected at t_1 and t_2 time such that $t_2 > t_1 \geq 0$ then $g_j^{(t_1)} \geq g_j^{(t_2)}$. At $t = 1$, $g_j^{(t_1)} \geq g_j^{(t_2)}$. We apply mathamatical induction. above inequality holds when $n \geq t_1$ and $n \geq t_2$ and we next prove this hold for $n+1 \geq t_1$ and $n+1 \geq t_2$. if i selected at time step t=n+1 and take t' an arbitrary earlier updating time step of node i such that $n \geq t' \geq 0$.

$$g_i^{t'} = \mathcal{H}(g_{j_1}^{\phi_1}, g_{j_2}^{\phi_2}, ...g_{j_{k_i}}^{\phi_{k_i}})$$

$$g_i^{n+1} = \mathcal{H}(g_{j_1}^{\varphi_1}, g_{j_2}^{\varphi_2}, ...g_{j_{k_i}}^{\varphi_{k_i}})$$

Here for any $m(k_i \geq m \geq 1)$, $n \geq \varphi_m \geq \phi_m$ and $g_{j_m}^{\phi_m} \geq g_{j_m}^{\varphi_m}$ therfore $g_i^{t'} \geq g_i^{n+1}$

for 0¡t_0 ¡ t_1 ¡ t_2 ¡...... we have $g_i^{t_0}, g_i^{t_1}, g_i^{t_2}, g_i^{t_3}...$ is a monotonously non increasing sequence and each element also nonnegative so g_i^∞ has limitation.

we first prove that for any node $j \in V, g_j^\infty \geq c_j$. it is proved by contradiction and for second part analogous to proof of theorem 1,after convergence any node $i \in V$ all nodes j such that $h_j^\infty \geq h_i^\infty$. it induced subgraph of $G^{'}$s g_i^∞ core which is c_i so we conclude $c_i \geq g_i^\infty$.

so above two inequality we proved therom.

2.2 Estimating Shell-Index in a Graph with Local Information

This main idea of this paper is to find the shell index of a node using local information. This will help to avoid the major drawback of K-Shell decomposition i.e. requirement of the the entire graph, which is not feasible for large scale dynamic networks.

2.2.1 Theorem

The Shell index of a node u can be computed as $k_s(u) = \mathcal{H}(k_s(v)| \in ngh(u))$, where $ngh(u)$ is the set of the neighbours of node u .

Proof : In the i^{th} iteration, nodes which have i or less connections with other nodes are removed. As it is the i^{th} iteration, therefore these removed nodes are connected with nodes which have shell index i or greater than i.

2.2.2 Hill Climbing Based approach to identify top rank nodes

In this algorithm inputs are Graph G, Initial node u, Repeat count number k (maximum no of times a crawler is allowed to reach a local maxima) and $maxindex$ (maximum $H_2 - Index$ in the graph G). Starting from the node u the crawler traverses to one of its non visited neighbours which also has the highest $H_2 - Index$. Upon continuing this process the crawler at one point reaches a maximum. If the maximum is a local maxima then the counter (initially 0) is increased by 1 and the flow of travel is passed on to one of its non visited neighbours randomly. This processes continues untill the crawler reaches the $maxindex$ or the counter reaches the value of k, at which point traversal is terminated.

Some modifications in the given algorithm includes traversal to highest degree neighbour instead of randomly choosen node in case of local maxima.

3 Work Done

Various methods were applied to reach from periphry to core based on only a limited number of node traversal. The following methods are discussed below :

3.1 Random walk

Here starting from any node in the graph, the core of the graph was to be reached. Movement from one node to the next neighbouring node was done randomly.

3.2 Hill climbing

Here starting from any node in the graph, the core of the graph was to be reached based on the value of $H_2 - Index$ of the neighbouring node. Two variations of the hill climbing method was seen.

3.2.1 Travelling highest degree node which is least travelled

Traversal to subsequent nodes is decided based on the highest $H_2 - Index$ neighbours of a node and among those the least travelled node. This allows traversal to be towards higher $H_2 - Index$ nodes mostly.

3.2.2 Travelling highest degree node which is least travelled but with a chance of random walk in every step

Traversal for this process is similar to the previous method but with a slight change. In every step of traversal there is a probability that the next node will be decided randomly from its neighbours rather than the usual process of highest $H_2 - Index$.

4 Dataset

- Facebook Combined
- Ca-CondMat
- Ca-GrQc
- Com-dblp.ungraph

- Email-EU-core
- RoadNet-CA
- P2p-Gnutella31
- Loc-gowalla_edges
- Gemsec_deezer_dataset RO_edges
- Gemsec_deezer_dataset HR_edges
- Gemsec_deezer_dataset HU_edges

4.1 Dataset Details

Dataset List					
Sl.no	Dataset	Directed or Undirected	Number of Nodes	Number of Edges	Maximum Shell number
1	Facebook Combined	Undirected	4039	88234	96
2	Ca-CondMat	Directed	23133	186936	21
3	Ca-GrQc	Directed	5242	28980	26
4	Com-dblp.ungraph	Undirected	317080	1049866	47
5	Email-EU-core	Undirected	1005	16706	35
6	RoadNet-CA	Directed	1965206	-	-
7	P2p-Gnutella31	Directed	62586	147892	6
8	Loc-gowalla_edges	Directed	196591	1900654	51
9	Gemsec_deezer_dataset RO_ edges	Undirected	41773	125826	7
10	Gemsec_deezer_dataset HR_ edges	Undirected	54573	498202	21
11	Gemsec_deezer_dataset HU_ edges	Undirected	47538	222887	11

5 Results

5.1 Random Walk

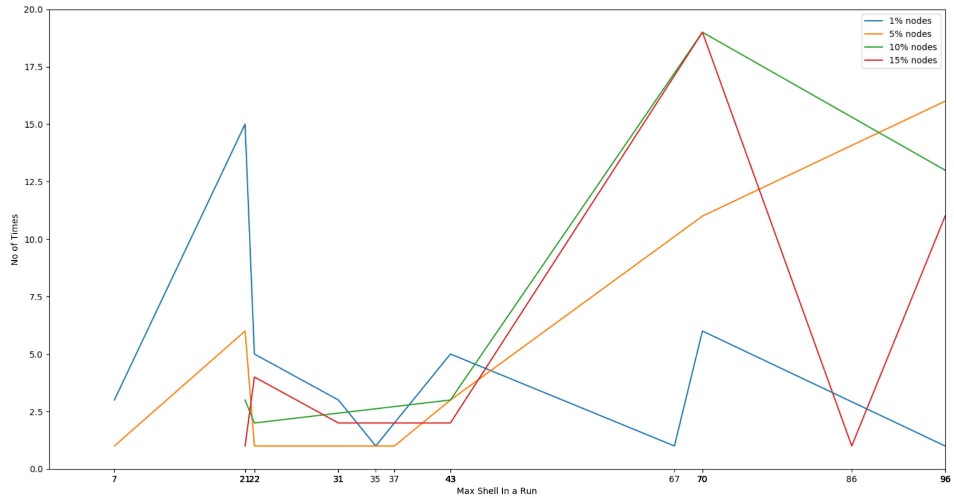

Fig 1.1 Graph for Random Walk for Dataset 1.

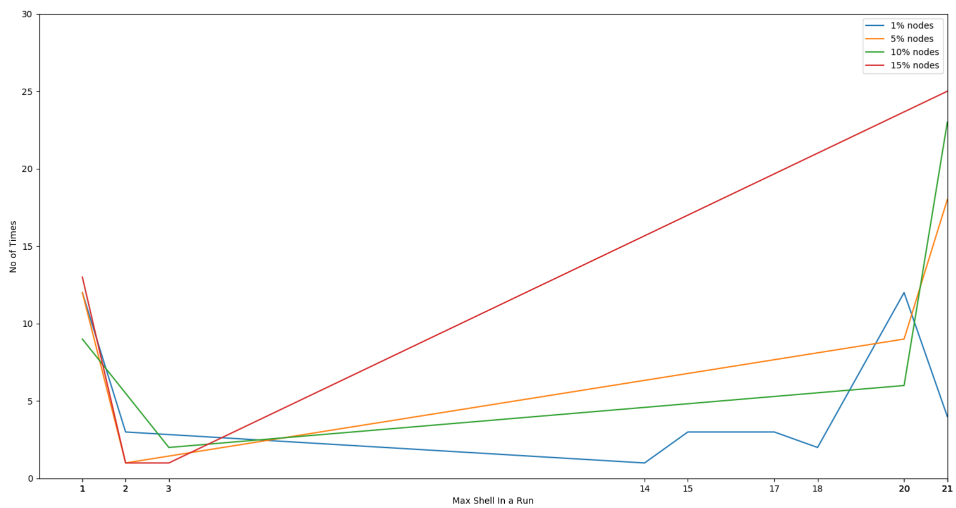

Fig 1.2 Graph for Random Walk for Dataset 2.

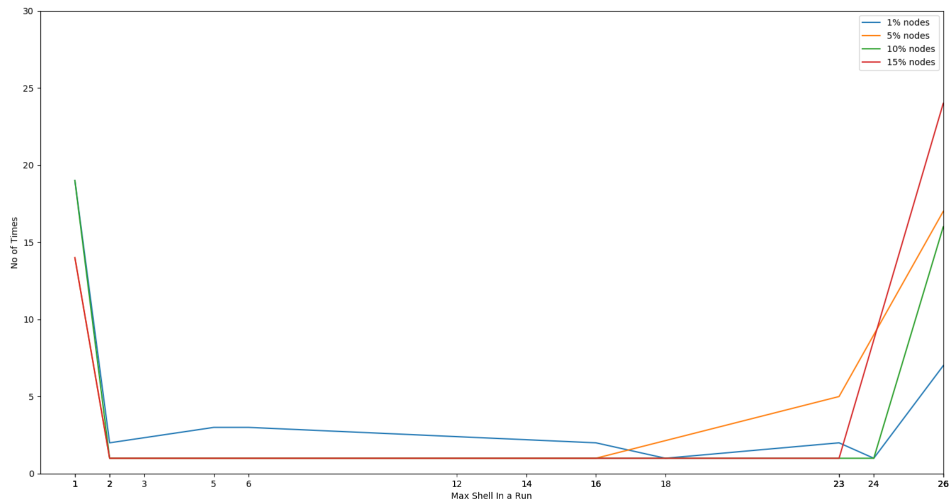

Fig 1.3 Graph for Random Walk for Dataset 3.

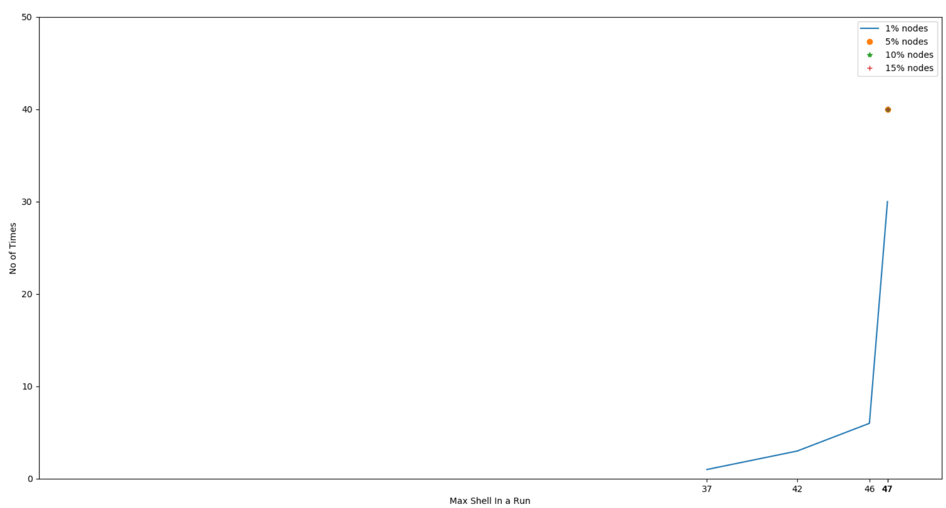

Fig 1.4 Graph for Random Walk for Dataset 4.

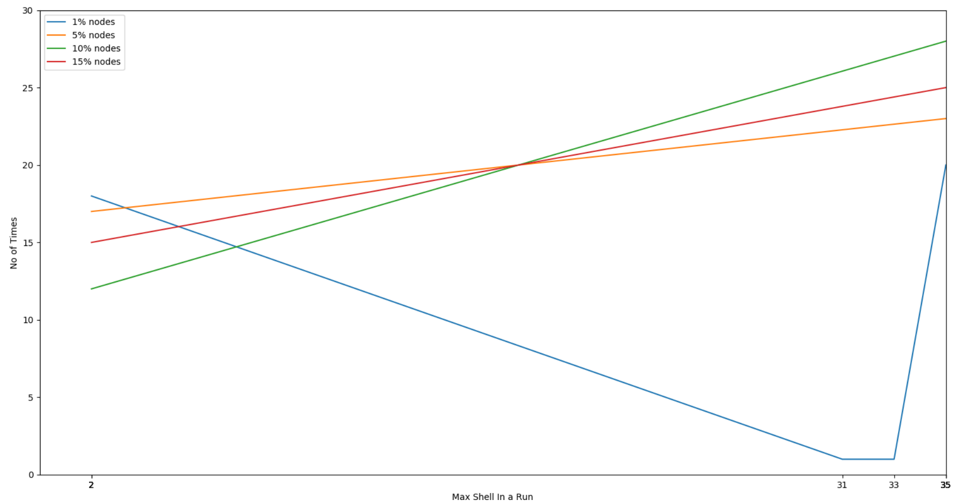

Fig 1.5 Graph for Random Walk for Dataset 5.

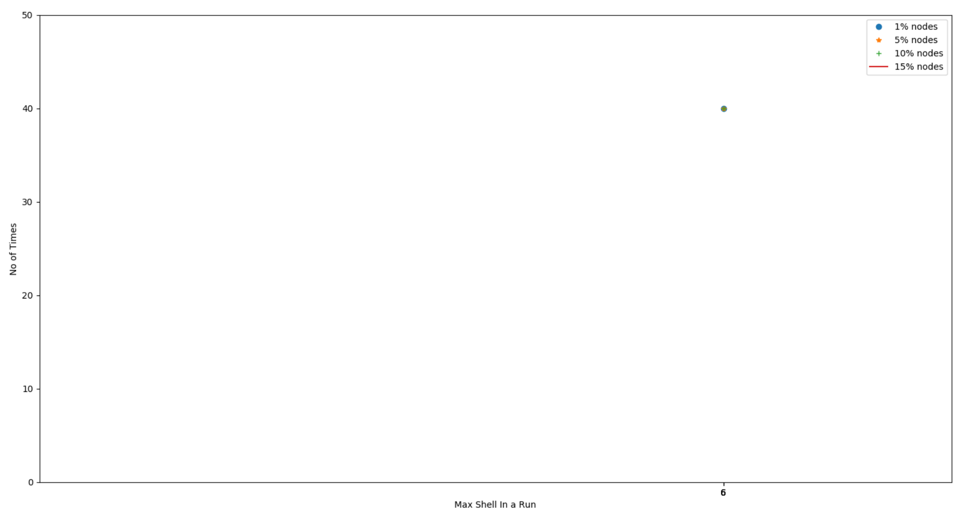

Fig 1.6 Graph for Random Walk for Dataset 7.

Fig 1.7 Graph for Random Walk for Dataset 8.

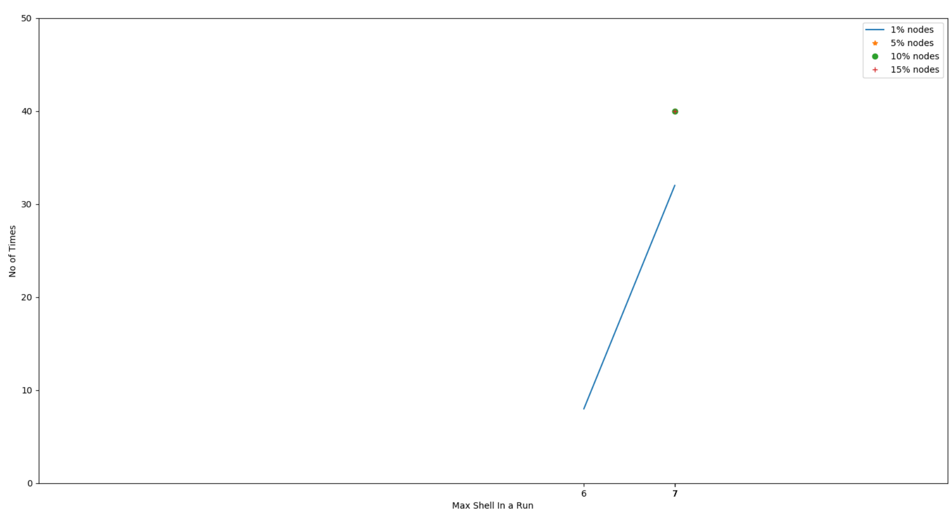

Fig 1.8 Graph for Random Walk for Dataset 9.

Fig 1.9 Graph for Random Walk for Dataset 10.

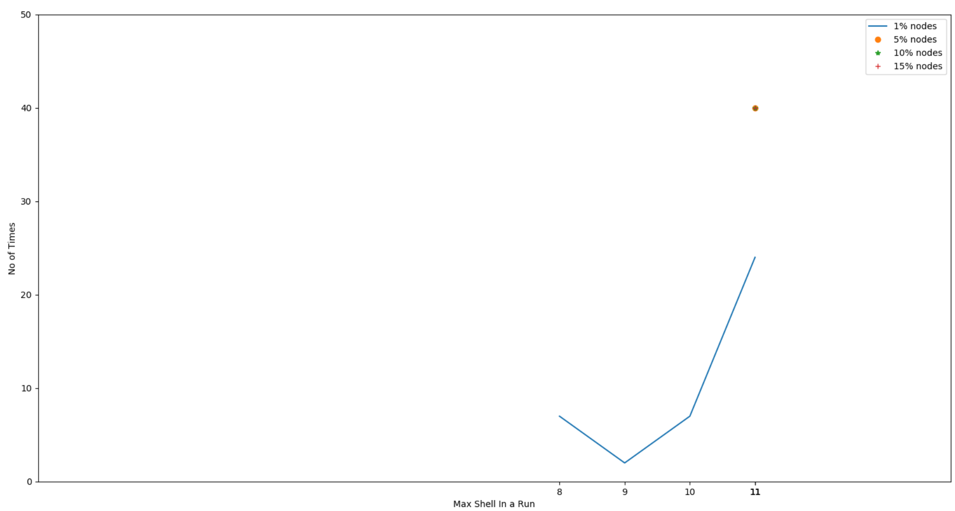

Fig 1.10 Graph for Random Walk for Dataset 11.

After plotting the graph of random walk for all the datasets it is seen that in all datasets, random walk leads to reaching the maximum shell number. Also in most graphs, increasing the percentage of nodes for random walk almost always increases the number of time maximum shell number is reached. Especially for datasets $7, 8, 9, 10$ and 11 where for some percentage of nodes all the runs reach the maximum shell. This may signify that for these datasets either the number of shells are too less comapared to the number of nodes or that the lower numbered shells are very well connected to the core shells.