IDENTIFICATION OF INFLUENTIAL NODES IN SOCIAL NETWORKS

Team Members:
15BCE0589 (Nikhil Lakkysetty)
15BCE0537 (Yeshwanth Kamisetty)

Report submitted for the
Final Project Review of

Course Code: CSE3021 – Social and Information Networks

Slot: A2 + TA2

Professor: Dr. Vasantha W B

School of Computing Science and Engineering

1. Introduction: Social networks have an important feature of "influential" nodes which have great implications ranging from forecasting marketing, advertisement, recommendation, controlling the spread of information in a network, identifying vulnerabilities in the Internet back-bone or other physical networks, and measuring the impact of organizational financial performance within highly correlated financial networks. Degree centrality is a straightforward and efficient metric, however, it is less relevant since a node having a few high influential neighbours may have much higher influence than a node having a larger number of less influential neighbours. Although some well-known global metrics such as betweenness centrality and closeness centrality can give better results, due to the very high computational complexity, they are not easy to manage very large-scale online social webs. The local degree centrality which is a semi-local centrality measure between the low-relevant degree centrality and other time-consuming measures performs almost as good as closeness centrality measure.

Apart from these centrality measures, the algorithms such as K-shell decomposition and page-rank algorithms are used to identify the influential nodes. In K-shell algorithm nodes in a graph are computed by conducting the k-Shell decomposition. Viewed as nodes in a graph, the higher the k-shell level assigned, the closer the node is to the core of the graph. The assumption is that, if these nodes are users in a social network, the users in the higher k-shell levels are more influential in the network than users in lower k-shell levels. The k-shell decomposition algorithm groups all nodes in a network that have *k* (or less) connections or that are only connected to other nodes with *k* (or less) connections. Once a node has been identified, it is marked (and removed from the network for purposes of the algorithm) and the search continues until all nodes in shell *k* have been found. The process then moves to the next larger k-shell value (and continues until all nodes have been marked). In this basic algorithm, k-shell values are assigned in a linear fashion. That is, each k-shell value is equivalent to the analyzed connection count. The page rank algorithm is a link analysis algorithm which outputs a probability distribution. It can be applied to any kind of network. PageRank interprets a hyperlink from page *x* to page *y* as a vote, by page *x*, for page *y*. However, PageRank looks at more than the sheer number of votes; it also analyses the page that casts the vote. A hyperlink from a page to another page is an implicit conveyance of authority to the target page. Pages that point to page 'I' also have their own prestige scores. This algorithm is purely based on prestige scores of web pages.

The performance is evaluated by using the number of cascade iterations and number of infected nodes. The cascading process is applied to get the number of infected nodes. Simulations are done on various real networks to evaluate the performance of these algorithms efficiently.

2. Literature Review Summary Table

Authors and Year (Reference)	Title (Study)	Concept / Theoretical model/ Framework	Methodology used/ Implementation	Dataset details / Analysis	Relevant Finding	Limitations/ Future Research/ Gaps identified
A.Flexman, A. Frieze and J. Vera in 2004	A geometrical preferential attachment model of networks	The proportion of vertices of given degree follows an inverse power law	Probabilistic Method	A random graph with n vertices	Approaching by random graphs also	Classical models of random graphs are not suitable for modelling these networks
Narayanam and Narahari in 2010	A Shapley value-based approach to discover influential nodes in social networks	Shapley value based approach	Greedy algorithm	Any social network	Discovering influential nodes	Provides a solution to the cooperative game theory
D.Kempe, J. Kleinberg and E. Tardosin 2003	Maximizing the spread of influence through a social network	Framework based on submodular functions	Hill Climbing greedy algorithm	Choosing influential sets of individual as a problem	Selecting most influential nodes	Implementing in large collaboration networks
S. Cheng, H. Shen, J. Huang, W.chen in 2014	Influence Maximization via findingself consistent ranking	Two ways: Greedy approach, Heuristic approach	IM Ranking algorithm	Any social network	Finding a set of seed nodes maximizing influence	Finding influential individuals such as expert finding, online advertising
M.U. Ilyas, H. Radha in 2011	Identifying Influential Nodes in Online Social Networks Using	Principal component centrality(PCC)	Comparing PCC with eigen vector centrality (EVC)	Massive online social network	Identifying influential nodes in online social networks	To find the increase social hubs

	Principal Component Centrality					
J. Leskovec, A. Krause in 2007	Cost-effective outbreak detection in networks	Works on submodular functions	Cost Effective Lazy Forward algorithm	Choosing influential sets of individual as a problem	Takes less time to achieve the process	Difficult to overcome hill-climbing greedy approach
Chi K. Tse, Jing Liu and Francis C.M. Lau in 2010	A network Perspective of the stock market	Minimal Spanning Tree is used for filtering networks	Finding correlation between each pair of	Stocks that were traded over two periods of time	Nodes are	To study correlations between the closing prices for all stocks

3. Objective of the project: The main objective of this project is to identify the influential nodes in complex networks. Detection of influential nodes in networks is a key problem of prime importance for a plenty of important practical applications. Our intent is to determine the extent to which algorithms and relevant graph analysis techniques can be successfully leveraged across multiple domains to potentially limit the impact of an attack, quantify vulnerabilities, and potentially inform network design. In this paper, we measured influential node discovery techniques such as local degree centrality, K-shell decomposition and Page rank algorithm in the social circles using datasets from Facebook, google+ and twitter. We evaluated the performance of these techniques by using the spreading rate and total number of infected nodes.

4. Innovation component in the project: Degree centrality is a straightforward and efficient metric, however, it is less relevant since a node having a few high influential neighbours may have much higher influence than a node having a larger number of less influential neighbours. Although some well-known global metrics such as betweenness centrality and closeness centrality can give better results, due to the very high computational complexity, they are not easy to manage very large-scale online social webs. In the project, we implemented a semi-locality measure, which is a trade-off between the low-relevant degree centrality measure and other time-consuming measures. This semi-locality measure (local degree centrality) includes the nearest and next nearest neighbours. For example, consider tow nodes ‘A’ and ‘B’. Suppose node ‘A’ has small degree centrality but large local centrality, while node B has large degree centrality but small local centrality. Although node ‘A’ has less neighbours, these neighbours have many connections with other nodes. So, if node 'A' is infected, it can affect more nodes through its neighbours. In contrast, if node 'B' connects many 1-degree nodes, the information cannot spread further. Thus, by considering the nearest and next nearest neighbours, we implemented a semi-centrality measure to identify the influential nodes. We compared the performance of this centrality measure with the K-shell and page rank algorithm by using number of cascade iterations and total number of infected nodes.

5. Work done and implementation

a. Methodology adapted:

Our methodology mainly consists of three phases:

Phase-1: Influential node detection

To determine the influential nodes, we executed the local degree centrality measure and algorithms such as K-shell decomposition and page-rank algorithm. We captured '10' most influential nodes as output in phase-1.

Phase-2: Cascading Execution

We leveraged the Independent Cascade (IC) algorithm to execute the cascade process across graphs with the '10' most influential nodes identified in Phase I as input and captured the total number of infected nodes, the number of cascade iterations, and the set of impacted nodes during each iteration.

Phase-3: Performance evaluation of these techniques

We evaluated the performance of these techniques by using total number of infected nodes and the number of cascade iterations obtained in phase-2. The more number of infected nodes represents the high performance of the technique.

We implemented the above mentioned algorithms in 'CPP' programming language and executed in Linux environment.

b. Dataset used:

i) We used four datasets in this project. Three of them taken are taken from the Stanford's website and other one is randomly generated graph.

ii) This project is based on the reference project present in Stanford website.

iii) We implemented a semi-locality measure and algorithms like K-shell and Page rank algorithm which differs from the reference project.

c. Tools used:

i) SocNetV:

We selected this tool since it has many visualisation options and easy to get values for centrality measures. We used this tool to visualize the data sets. We obtained the influential nodes based on centrality measures present in the tool and compared with our 'semi-centrality measure'.

ii) Gephi:

We used this tool to visualize the social networks. We obtained the most influential nodes based on 'page-rank algorithm' present in the tool and verified with our implemented page-rank algorithm.

d. Screenshot and Demo:

Code for K Shell Algorithm:

```
#include "bits/stdc++.h"
#include "socialnetwork.h"
using namespace std;

vector <int> seed_points_ks;

extern Graph Node_Properties[MAX_NODES];

int Adj_matrix[MAX_NODES][MAX_NODES];

vector <int> shell[MAX_NODES];
vector <bool> incl(MAX_NODES, false);

int ithshell(int x) {
  int i;
  for(i = 0; i < MAX_NODES; ++i) {
    bool ok = false;
    for(int j = 0; j < MAX_NODES; ++j) {
      int cnt = 0;
      for(int k = 0; k < MAX_NODES; ++k) cnt += Adj_matrix[j][k];
      if(cnt <= x) {
        if(incl[j]) continue;
        incl[j] = true;
        shell[x].push_back(j);
        for(int k = 0; k < MAX_NODES; ++k) {
          if(Adj_matrix[j][k]) {
            ok = true;
            Adj_matrix[j][k] = Adj_matrix[k][j] = 0;
          }
        }
      }
    }
    if(!ok) break;
  }
}

void kshell_centrality() {
  for(int i = 0; i < MAX_NODES; ++i){
    for(int j = 0; j < Node_Properties[i].No_of_Neighbours(); ++j) {
      Adj_matrix[i][Node_Properties[i].Neighbour(j)] = 1;
      Adj_matrix[Node_Properties[i].Neighbour(j)][i] = 1;
    }
  }
```

```
  int last_shell = 0;
  for(last_shell; last_shell < MAX_NODES; ++last_shell) {
    if(ithshell(last_shell) == -1) break;
  }
  last_shell--;
  int k = TOP_K;
  for(int i = last_shell; i >= 0 && k; --i) {
    for(int j : shell[i]) {
      if(k-- <= 0) break;
      seed_points_ks.push_back(j);
    }
  }
}
```

Outputs for 4 Datasets:

Cascading_KS.out (~/Projects/Sem V/CSE3021/source) - gedit

Open Save

Cascading_LDC.out × Cascading_KS.out ×

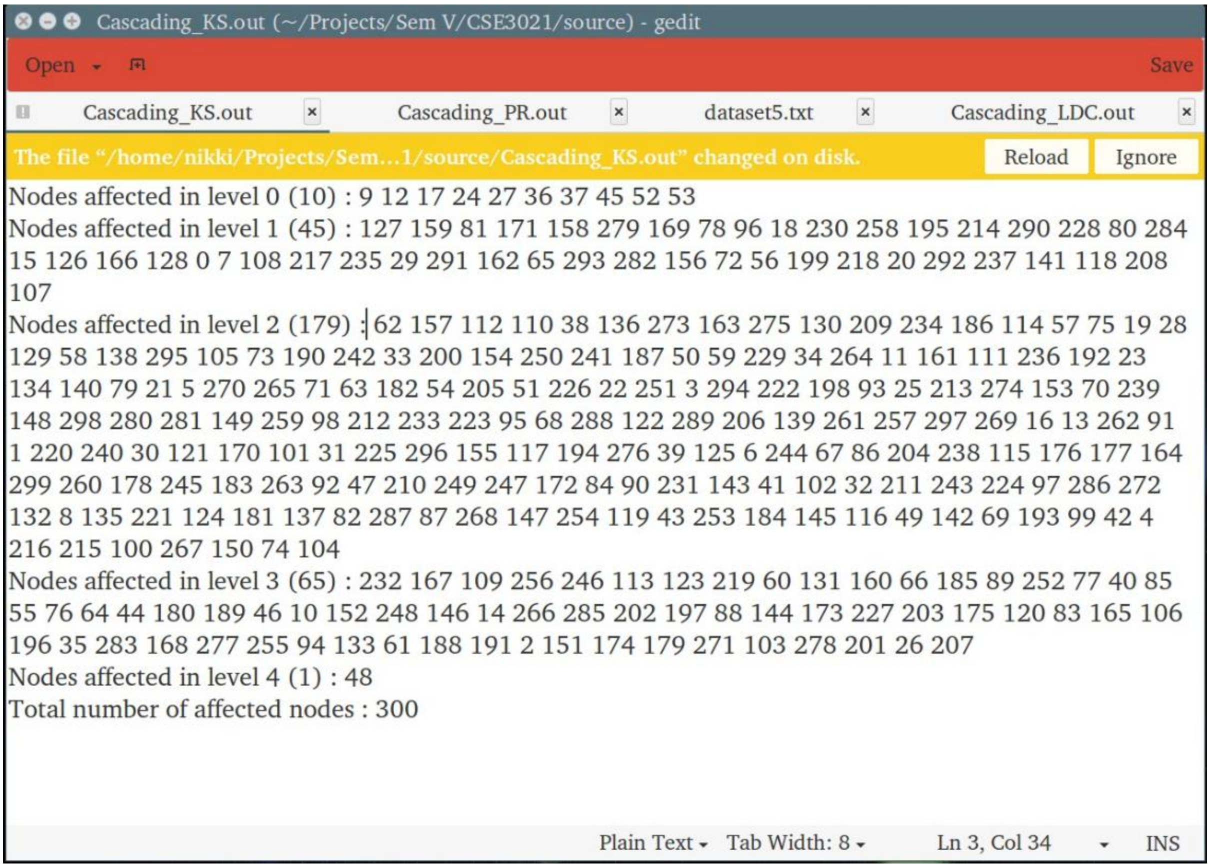
```
Nodes affected in level 0 (10) : 9 12 24 27 36 45 52 61 64 75
Nodes affected in level 1 (46) : 127 159 81 171 158 279 169 78 195 214 290 228 80 284
15 126 166 128 0 7 108 217 235 282 156 72 56 199 218 20 30 254 3 248 276 134 123 23
57 201 43 91 181 104 185 6
Nodes affected in level 2 (166) : 62 157 112 110 38 136 273 107 163 275 130 209 96 234
186 114 19 28 129 58 138 295 105 73 190 17 242 33 200 154 250 241 187 291 50 59 229
34 264 11 161 111 236 192 237 70 239 148 298 281 149 259 299 98 212 79 233 223 226
95 68 288 122 5 289 206 139 261 257 297 269 16 13 262 1 220 71 37 240 121 170 101
31 225 296 155 117 194 39 125 244 294 67 86 204 238 115 176 177 164 260 178 245
183 263 92 47 210 249 247 172 84 90 231 140 293 143 41 251 208 222 102 135 21 221
124 8 137 292 82 287 87 268 147 119 22 253 145 93 49 150 173 196 286 232 133 189
230 278 42 151 35 146 203 213 113 132 44 167 165 272 26 69 60 211 160
Nodes affected in level 3 (76) : 182 29 4 109 256 246 32 18 280 219 131 74 141 53 66 162
142 153 89 65 252 77 40 193 54 85 55 76 180 267 46 63 10 152 51 174 14 266 258 285
99 224 202 197 88 243 144 227 175 120 83 106 216 184 215 2 277 168 205 116 179 271
103 274 97 118 94 188 265 100 207 255 283 25 48 270
Nodes affected in level 4 (2) : 198 191
Total number of affected nodes : 300
```

Cascading_KS.out (~/Projects/Sem V/CSE3021/source) - gedit

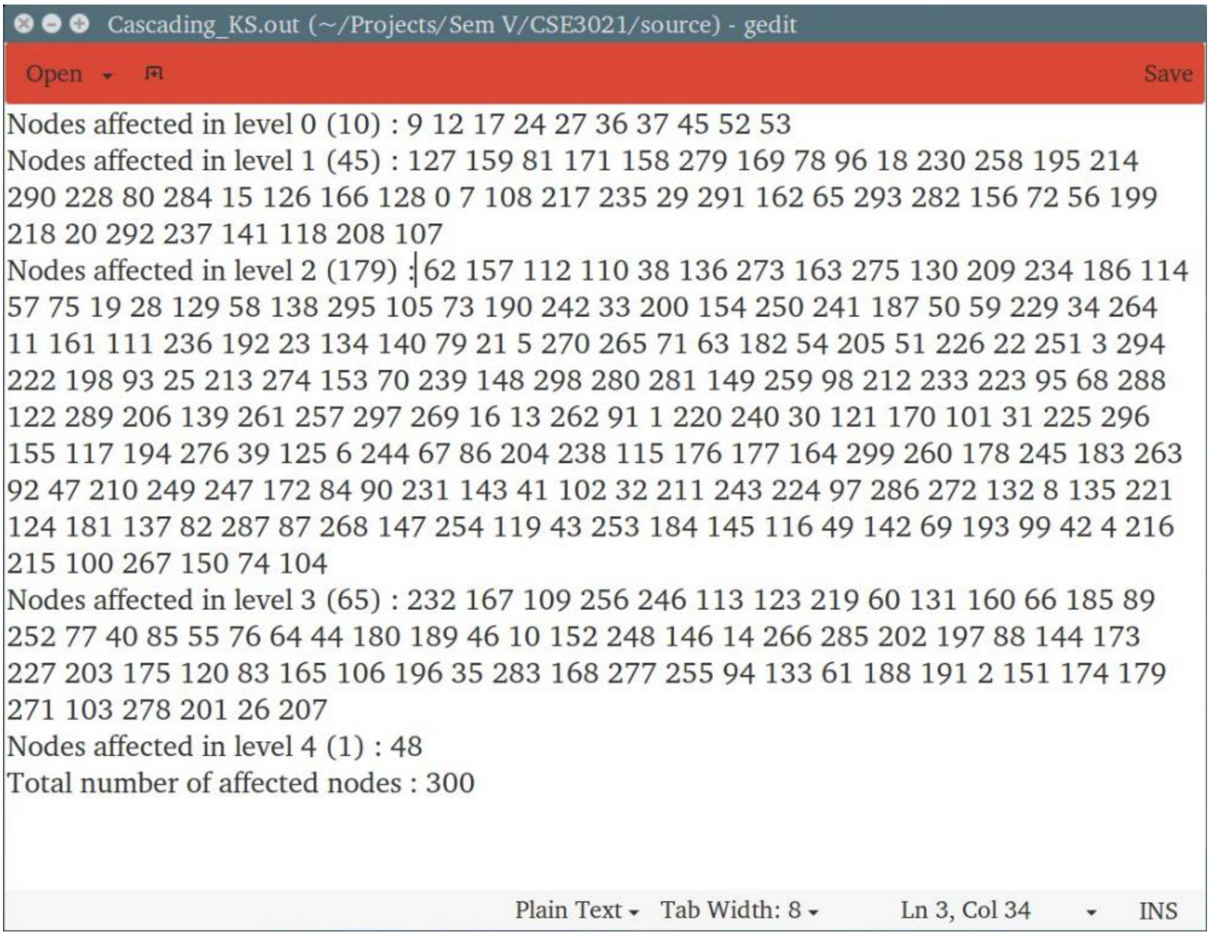
Open Save

Cascading_LDC.out × Cascading_PR.out × Cascading_KS.out ×

```
Nodes affected in level 0 (10) : 15 40 41 42 54 66 68 71 75 77
Nodes affected in level 1 (50) : 127 87 106 170 196 36 129 6 34 78 96 130 86 150 155 48 135 102 121 11
61 24 158 30 190 13 73 166 189 126 21 28 142 163 103 156 128 69 80 29 180 81 7 104 164 181 123 113
140 100
Nodes affected in level 2 (131) : 84 109 12 5 10 136 8 131 72 122 53 2 115 183 83 199 62 195 47 118 152
99 137 169 167 117 4 132 50 70 134 44 0 60 160 17 57 90 46 168 157 178 192 149 23 101 85 175 26 79
105 197 25 18 184 186 182 151 22 125 112 98 116 43 179 14 110 193 147 153 133 67 143 154 64 108 74
97 55 20 148 139 82 65 171 3 187 94 19 120 16 162 88 93 159 198 173 95 124 49 63 141 161 138 191
176 58 37 31 59 194 27 119 92 172 39 33 52 45 38 145 51 91 114 174 111 89 9 144 177 146
Nodes affected in level 3 (9) : 165 1 107 35 32 56 188 76 185
Total number of affected nodes : 200
```



Nodes affected in level 0 (10) : 9 12 17 24 27 36 37 45 52 53
Nodes affected in level 1 (45) : 127 159 81 171 158 279 169 78 96 18 230 258 195 214 290 228 80 284 15 126 166 128 0 7 108 217 235 29 291 162 65 293 282 156 72 56 199 218 20 292 237 141 118 208 107
Nodes affected in level 2 (179) : 62 157 112 110 38 136 273 163 275 130 209 234 186 114 57 75 19 28 129 58 138 295 105 73 190 242 33 200 154 250 241 187 50 59 229 34 264 11 161 111 236 192 23 134 140 79 21 5 270 265 71 63 182 54 205 51 226 22 251 3 294 222 198 93 25 213 274 153 70 239 148 298 280 281 149 259 98 212 233 223 95 68 288 122 289 206 139 261 257 297 269 16 13 262 91 1 220 240 30 121 170 101 31 225 296 155 117 194 276 39 125 6 244 67 86 204 238 115 176 177 164 299 260 178 245 183 263 92 47 210 249 247 172 84 90 231 143 41 102 32 211 243 224 97 286 272 132 8 135 221 124 181 137 82 287 87 268 147 254 119 43 253 184 145 116 49 142 69 193 99 42 4 216 215 100 267 150 74 104
Nodes affected in level 3 (65) : 232 167 109 256 246 113 123 219 60 131 160 66 185 89 252 77 40 85 55 76 64 44 180 189 46 10 152 248 146 14 266 285 202 197 88 144 173 227 203 175 120 83 165 106 196 35 283 168 277 255 94 133 61 188 191 2 151 174 179 271 103 278 201 26 207
Nodes affected in level 4 (1) : 48
Total number of affected nodes : 300



Nodes affected in level 0 (10) : 9 12 17 24 27 36 37 45 52 53
Nodes affected in level 1 (45) : 127 159 81 171 158 279 169 78 96 18 230 258 195 214 290 228 80 284 15 126 166 128 0 7 108 217 235 29 291 162 65 293 282 156 72 56 199 218 20 292 237 141 118 208 107
Nodes affected in level 2 (179) : 62 157 112 110 38 136 273 163 275 130 209 234 186 114 57 75 19 28 129 58 138 295 105 73 190 242 33 200 154 250 241 187 50 59 229 34 264 11 161 111 236 192 23 134 140 79 21 5 270 265 71 63 182 54 205 51 226 22 251 3 294 222 198 93 25 213 274 153 70 239 148 298 280 281 149 259 98 212 233 223 95 68 288 122 289 206 139 261 257 297 269 16 13 262 91 1 220 240 30 121 170 101 31 225 296 155 117 194 276 39 125 6 244 67 86 204 238 115 176 177 164 299 260 178 245 183 263 92 47 210 249 247 172 84 90 231 143 41 102 32 211 243 224 97 286 272 132 8 135 221 124 181 137 82 287 87 268 147 254 119 43 253 184 145 116 49 142 69 193 99 42 4 216 215 100 267 150 74 104
Nodes affected in level 3 (65) : 232 167 109 256 246 113 123 219 60 131 160 66 185 89 252 77 40 85 55 76 64 44 180 189 46 10 152 248 146 14 266 285 202 197 88 144 173 227 203 175 120 83 165 106 196 35 283 168 277 255 94 133 61 188 191 2 151 174 179 271 103 278 201 26 207
Nodes affected in level 4 (1) : 48
Total number of affected nodes : 300

Code for Cascading Algorithm:

```
void cascade_ALG() {
  freopen("Cascading_ALG.out","w",stdout);
  vector <int> affected[300];
  extern vector <int> seed_points_dc;
  vector <bool> visited(MAX_NODES, false);
  affected[0] = seed_points_dc;
  for(int i : seed_points_dc) visited[i] = true;
  int lvl = 0;
  while(!affected[lvl].empty()) {
    for(int i : affected[lvl]) {
      for(int j = 0; j < Node_Properties[i].No_of_Neighbours(); ++j) {
        if(visited[Node_Properties[i].Neighbour(j)]) continue;
        visited[Node_Properties[i].Neighbour(j)] = true;
        affected[lvl + 1].push_back(Node_Properties[i].Neighbour(j));
      }
    }
    lvl++;
  }
  int sum = 0;
  for(int i = 0; i < lvl; ++i) {
    sum += affected[i].size();
    cout << "Nodes affected in level " << i << " (" << affected[i].size() << ") : ";
    for(int j : affected[i]) {
      cout << j << " ";
    }
    cout << "\n";
  }
  cout << "Total number of affected nodes : " << sum << "\n";
}
```

Code for Local degree centrality measure:

```
#include "bits/stdc++.h"
#include "socialnetwork.h"
using namespace std;

vector <int> seed_points_ldc;

extern Graph Node_Properties[MAX_NODES];

void local_degree_centrality() {
  vector <int> ldc[MAX_NODES];
  for(int i = 0; i < MAX_NODES; ++i) {
    vector <bool> vis(MAX_NODES, false);
    vector <int> tmp;
    for(int j = 0; j < Node_Properties[i].No_of_Neighbours(); ++j) {
      if(vis[Node_Properties[i].Neighbour(j)]) continue;
      vis[Node_Properties[i].Neighbour(j)] = true;
      ldc[i].push_back(Node_Properties[i].Neighbour(j));
      tmp.push_back(Node_Properties[i].Neighbour(j));
    }
    for(int k : tmp) {
      for(int j = 0; j < Node_Properties[k].No_of_Neighbours(); ++j) {
        if(vis[Node_Properties[k].Neighbour(j)]) continue;
        vis[Node_Properties[k].Neighbour(j)] = true;
        ldc[i].push_back(Node_Properties[k].Neighbour(j));
      }
    }
    Node_Properties[i].LDC_Value = ldc[i].size();
  }
  vector <pair <double, int> > tmp;
  for(int i = 0; i < MAX_NODES; ++i) {
    tmp.push_back(make_pair(Node_Properties[i].LDC_Value, i));
  }
  sort(tmp.rbegin(), tmp.rend());
  for(int i = 0; i < TOP_K; ++i) {
    seed_points_ldc.push_back(tmp[i].second);
  }
}
```

Output for 4 Datasets:

Cascading_LDC.out (~/Projects/Sem V/CSE3021/source) - gedit

Open Save

```
Nodes affected in level 0 (10) : 284 228 57 269 290 5 129 236 200 297
Nodes affected in level 1 (111) : 37 27 240 190 30 121 170 195 233 101 31 225 296 155
117 194 276 39 125 288 122 289 3 206 139 110 261 257 24 16 149 13 262 214 244 213
166 185 113 136 132 81 64 107 44 267 22 179 163 271 217 10 23 34 114 7 259 299 295
98 212 282 154 161 298 79 223 226 95 68 239 178 18 134 250 90 4 260 273 46 40 171
59 71 63 249 123 251 120 204 160 60 6 227 169 246 281 254 77 197 111 67 20 242 88
158 42 232 148 153 116
Nodes affected in level 2 (167) : 29 291 162 65 293 15 126 128 159 108 41 150 140 147
49 266 235 61 173 196 21 174 55 231 252 72 85 141 62 56 237 118 70 199 33 253 263
103 176 51 287 73 2 135 105 186 91 192 238 157 181 14 193 243 69 131 133 268 278
183 151 28 137 109 83 87 66 94 172 294 142 124 82 32 205 210 86 216 119 234 84 230
202 285 112 211 92 224 280 127 264 275 221 80 8 152 220 241 189 115 229 50 168 104
54 0 89 274 182 258 177 130 75 38 283 219 138 209 9 96 45 19 201 43 74 53 256 208
187 184 143 164 100 36 52 11 58 203 144 76 78 47 247 222 218 277 106 1 286 245 102
17 270 265 35 146 279 255 25 198 180 97 26 12 99 167 292
Nodes affected in level 3 (12) : 272 165 248 48 215 145 93 156 188 175 191 207
Total number of affected nodes : 300
```

Plain Text Tab Width: 8 Ln 5, Col 37 INS

Cascading_LDC.out (~/Projects/Sem V/CSE3021/source) - gedit

Open Save

```
Nodes affected in level 0 (10) : 139 190 154 21 169 90 57 92 140 128
Nodes affected in level 1 (107) : 12 94 105 11 31 109 64 124 2 46 16 163 104 80 166 44 95 98 49 29 25 82
54 183 63 141 193 161 198 79 102 199 186 43 135 177 173 32 100 194 133 138 119 167 184 18 66 182
159 152 67 172 60 148 85 117 179 10 36 110 170 97 111 59 99 84 26 146 112 58 125 176 150 33 123 61
126 113 27 115 144 13 108 132 81 53 195 191 151 107 50 9 147 142 149 22 137 77 189 168 121 71 129
88 6 39 78
Nodes affected in level 2 (83) : 87 127 17 181 19 93 136 28 30 4 42 143 34 65 62 187 73 89 165 114 37 47
106 116 131 51 24 118 3 197 83 55 120 153 130 75 156 7 101 160 196 171 74 158 86 134 164 14 157
185 188 96 69 5 35 70 145 0 56 155 180 23 20 68 41 38 1 76 45 103 48 8 175 162 91 15 122 52 174 72
40 178 192
Total number of affected nodes : 200
```

Cascading_LDC.out (~/Projects/Sem V/CSE3021/source) - gedit

Open Save

Cascading_KS.out | Cascading_PR.out | dataset5.txt | Cascading_LDC.out

```
Nodes affected in level 0 (10) : 129 228 290 119 5 273 267 121 190 259
Nodes affected in level 1 (44) : 46 40 68 57 171 288 122 289 3 299 295 98 24 212 282 21 244 224 199
236 239 178 18 134 56 131 159 176 269 14 256 296 249 155 174 55 284 231 149 246 251 105 0 107
Nodes affected in level 2 (77) : 206 22 242 11 238 214 28 197 125 82 268 133 243 152 196 158 113
195 287 286 45 140 104 90 126 202 237 162 254 281 43 70 120 110 204 160 294 2 79 17 164 64 35
52 117 193 136 9 168 4 179 163 213 181 63 41 114 233 143 132 150 37 27 240 30 217 262 39 189 67
200 100 232 225 1 36 275
Nodes affected in level 3 (61) : 94 248 10 71 166 218 137 153 172 194 103 50 291 111 16 73 226 38 42
283 192 156 167 13 60 54 270 61 47 184 124 169 29 142 6 96 230 128 69 135 205 219 127 81 118
255 88 77 157 235 144 15 139 74 221 65 177 186 80 7 108
Nodes affected in level 4 (45) : 253 297 145 93 49 173 148 241 227 272 99 44 8 247 263 280 276 161
95 201 257 72 215 83 138 102 32 211 23 182 141 188 62 130 209 87 234 293 112 92 115 85 91 78
187
Nodes affected in level 5 (20) : 258 264 86 265 250 151 278 183 123 292 208 106 26 84 175 223 170
279 75 229
Nodes affected in level 6 (7) : 198 245 53 266 58 12 185
Total number of affected nodes : 264
```

Plain Text ▾ Tab Width: 8 ▾ Ln 8, Col 37 ▾ INS

Cascading_LDC.out (~/Projects/Sem V/CSE3021/source) - gedit

Open Save

Cascading_KS.out | Cascading_LDC.out

```
Nodes affected in level 0 (10) : 284 228 57 269 5 200 129 236 161 190
Nodes affected in level 1 (111) : 37 27 240 30 121 170 195 233 101 31 225 296 155 117
194 276 39 125 288 122 289 3 206 139 110 261 257 297 24 16 149 13 262 214 244 213
166 185 113 136 132 81 64 107 44 267 22 179 163 271 217 10 23 34 114 7 239 178 18
134 250 90 4 260 246 281 254 77 197 111 67 20 242 88 158 42 273 46 40 68 171 59 71
63 249 123 251 120 204 160 60 6 227 259 169 83 290 219 106 216 294 196 135 96 41
150 140 147 49 266 235
Nodes affected in level 2 (167) : 29 291 162 65 293 15 126 128 159 108 61 173 21 174 55
231 252 72 85 141 62 56 237 118 70 199 148 298 33 280 253 98 263 103 176 51 287 73
2 105 186 91 192 238 157 181 14 193 243 69 299 212 131 133 268 86 278 183 151 28
137 109 87 66 94 172 142 124 82 32 205 95 210 79 119 234 84 230 202 285 112 211 92
224 127 264 275 221 153 80 8 152 220 241 189 115 154 229 50 168 104 54 0 89 274 182
258 177 130 75 226 38 283 138 207 209 9 45 19 201 43 74 53 223 256 208 187 184 143
164 100 36 52 11 58 203 144 76 78 47 247 17 270 265 35 146 279 102 255 286 232 99
282 26 167 295 180 25 198 97 222 218 12 188 248 215 165
Nodes affected in level 3 (12) : 272 1 48 145 93 292 156 277 245 175 191 116
Total number of affected nodes : 300
```

Plain Text ▾ Tab Width: 8 ▾ Ln 5, Col 37 ▾ INS

Code for Page Rank Algorithm:

```
#include "bits/stdc++.h"
#include "socialnetwork.h"
using namespace std;

vector <int> seed_points_pr;

extern Graph Node_Properties[MAX_NODES];

double stochastic_matrix[MAX_NODES][MAX_NODES];

const double damp_factor = 0.85;
const double epsilon = 0.0005;

vector <double> mat_multiply(vector <double> x) {
  vector <double> ans(MAX_NODES, 0);
  for(int i = 0; i < MAX_NODES; ++i) {
    for(int j = 0; j < MAX_NODES; ++j) {
      ans[i] += stochastic_matrix[i][j] * x[j];
    }
  }
  return ans;
}

vector <double> power_iteration() {
  vector <double> prev(MAX_NODES, 1);
  vector <double> cm(MAX_NODES, 0.15);
  vector <double> curr(MAX_NODES), tmp;
  double nlp = 1, clp;
  while(1) {
    tmp = mat_multiply(prev);
    for(int i = 0; i < MAX_NODES; ++i) {
      curr[i] = cm[i] + tmp[i];
    }
    clp = -1.0;
    for(int i = 0; i < MAX_NODES; ++i) {
      if(curr[i] > clp) clp = curr[i];
    }
    for(int i = 0; i < MAX_NODES; ++i) {
      curr[i] /= clp;
    }
    if(abs(nlp - clp) < epsilon) break;
    prev = curr;
    nlp = clp;
```

```
    if(abs(nlp - clp) < epsilon) break;
    prev = curr;
    nlp = clp;
  }
  return curr;
}

void pagerank_centrality() {
  memset(stochastic_matrix, 0, sizeof(stochastic_matrix));
  for(int i = 0; i < MAX_NODES; ++i){
    for(int j = 0; j < Node_Properties[i].No_of_Neighbours(); ++j) {
      stochastic_matrix[i][Node_Properties[i].Neighbour(j)] = 1;
      stochastic_matrix[Node_Properties[i].Neighbour(j)][i] = 1;
    }
  }
  for(int i = 0; i < MAX_NODES; ++i) {
    for(int j = 0; j < MAX_NODES; ++j) {
      if(stochastic_matrix[i][j] > 0.5) stochastic_matrix[i][j] = 1.0 / Node_Properties[i].No_of_Neighbours();
    }
  }
  // Transpose
  for(int i = 0; i < MAX_NODES; ++i) {
    for(int j = i + 1; j < MAX_NODES; ++j) {
      swap(stochastic_matrix[i][j], stochastic_matrix[j][i]);
    }
  }
  // Multiply with damping factor
  for(int i = 0; i < MAX_NODES; ++i) {
    for(int j = 0; j < MAX_NODES; ++j) {
      stochastic_matrix[i][j] *= damp_factor;
    }
  }
  vector <double> pagerank = power_iteration();
  priority_queue <pair<double, int> > pq;
  for(int i = 0; i < MAX_NODES; ++i) {
    pq.push(make_pair(pagerank[i], i));
  }
  int k = TOP_K;
  while(k--) {
    seed_points_pr.push_back(pq.top().second);
    pq.pop();
  }
}
```

Outputs for 4 Datasets:

Cascading_PR.out (~/Projects/Sem V/CSE3021/source) - gedit

Open Save

Cascading_LDC.out Cascading_PR.out

```
Nodes affected in level 0 (10) : 284 228 290 57 269 129 169 200 190 276
Nodes affected in level 1 (117) : 37 27 240 30 121 170 195 233 101 31 225 296 155 117
194 39 125 288 122 5 289 3 206 139 110 261 257 297 24 16 149 13 262 259 299 295 98
212 282 154 163 161 298 79 223 226 95 68 214 244 213 166 185 113 136 132 81 236 64
107 44 267 22 179 271 217 10 23 34 114 7 273 46 40 171 59 71 63 249 123 251 235 12
50 45 229 264 138 11 111 192 246 281 254 77 197 67 20 242 250 88 158 42 41 150 140
147 49 266 278 183 291 90 151 28 137 61
Nodes affected in level 2 (160) : 29 162 65 293 15 126 128 159 108 204 173 196 21 174
55 231 252 72 160 85 141 62 56 237 118 70 239 199 148 33 253 263 103 176 51 287 73
2 135 105 186 91 238 157 181 4 134 14 193 243 69 60 131 133 268 109 83 87 66 94 172
294 142 124 82 32 205 178 18 260 210 86 216 119 234 84 230 202 285 112 211 92 224
280 127 275 221 232 153 116 80 8 152 220 241 189 115 168 104 54 0 222 219 218 177
277 106 1 286 38 144 43 130 47 245 102 89 274 182 258 75 283 209 9 96 19 120 6 227
201 74 53 256 208 187 184 143 164 100 36 52 58 203 76 78 247 25 198 180 279 272 248
175 156 165 99 26 167 17 255
Nodes affected in level 3 (13) : 270 97 48 215 145 93 292 35 146 265 188 191 207
Total number of affected nodes : 300
```

Cascading_PR.out (~/Projects/Sem V/CSE3021/source) - gedit

Open Save

Cascading_LDC.out | Cascading_PR.out

Nodes affected in level 0 (10) : 92 190 140 139 21 169 154 90 128 142
Nodes affected in level 1 (107) : 186 117 172 53 195 191 151 125 107 50 9 16 147 182 148 110 44 95 98 124 49 29 25 82 54 183 63 141 193 161 198 79 102 26 149 22 84 67 137 77 189 176 144 11 168 121 12 94 105 31 109 64 2 46 163 104 80 166 119 167 184 18 57 66 159 152 60 85 179 10 36 135 32 170 97 138 111 199 43 177 173 100 194 133 59 99 146 112 58 150 33 123 61 126 71 129 88 6 39 27 78 187 130 162 81 0 30
Nodes affected in level 2 (83) : 165 13 62 96 86 196 160 69 158 136 113 91 143 37 87 131 108 72 35 73 157 197 55 5 52 181 20 24 118 3 83 23 7 155 145 132 175 19 89 171 74 134 115 114 47 28 164 14 51 185 188 70 116 153 93 56 180 34 156 68 41 120 127 17 1 38 65 4 174 42 76 106 75 101 8 45 15 103 48 122 40 178 192
Total number of affected nodes : 200

Cascading_PR.out (~/Projects/Sem V/CSE3021/source) - gedit

Open Save

Cascading_KS.out | Cascading_PR.out | dataset5.txt | Cascading_LDC.out

The file "/home/nikki/Projects/Sem...1/source/Cascading_PR.out" changed on disk. Reload Ignore

Nodes affected in level 0 (10) : 284 228 57 269 129 169 200 42 190 161
Nodes affected in level 1 (115) : 37 27 240 30 121 170 195 233 101 31 225 296 155 117 194 276 39 125 288 122 5 289 3 206 139 110 261 257 297 24 16 149 13 262 214 244 213 166 185 113 136 132 81 236 64 107 44 267 22 179 163 271 217 10 23 34 114 7 273 46 40 68 171 59 71 63 249 123 251 235 12 50 45 229 264 138 11 111 192 246 281 254 77 197 67 20 242 250 88 158 164 287 168 130 86 280 180 234 141 184 41 150 140 147 49 266 83 290 219 106 216 294 196 135 96
Nodes affected in level 2 (165) : 29 291 162 65 293 15 126 128 159 108 61 204 173 21 174 55 231 252 72 160 85 62 56 237 118 70 239 199 148 298 33 253 98 263 103 176 51 73 2 105 186 90 91 238 157 181 134 14 193 243 69 60 299 212 131 133 4 268 278 183 151 28 137 109 87 66 94 172 142 124 82 32 205 95 178 18 260 210 79 119 84 230 202 285 112 211 92 224 127 275 221 153 80 8 152 220 241 189 115 154 104 54 0 89 274 182 258 177 75 226 38 283 207 209 9 19 120 6 227 259 201 43 74 53 223 256 208 187 143 100 36 52 58 203 144 76 78 47 247 25 102 198 232 222 279 272 248 175 282 156 165 1 99 286 26 167 295 17 270 265 93 255 146 188 215
Nodes affected in level 3 (10) : 97 218 48 145 292 35 277 245 191 116
Total number of affected nodes : 300

Cascading_PR.out (~/Projects/Sem V/CSE3021/source) - gedit

Open Save

Cascading_KS.out | Cascading_PR.out

Nodes affected in level 0 (10) : 284 228 57 269 129 169 200 42 190 161
Nodes affected in level 1 (115) : 37 27 240 30 121 170 195 233 101 31 225 296 155 117 194 276 39 125 288 122 5 289 3 206 139 110 261 257 297 24 16 149 13 262 214 244 213 166 185 113 136 132 81 236 64 107 44 267 22 179 163 271 217 10 23 34 114 7 273 46 40 68 171 59 71 63 249 123 251 235 12 50 45 229 264 138 11 111 192 246 281 254 77 197 67 20 242 250 88 158 164 287 168 130 86 280 180 234 141 184 41 150 140 147 49 266 83 290 219 106 216 294 196 135 96
Nodes affected in level 2 (165) : 29 291 162 65 293 15 126 128 159 108 61 204 173 21 174 55 231 252 72 160 85 62 56 237 118 70 239 199 148 298 33 253 98 263 103 176 51 73 2 105 186 90 91 238 157 181 134 14 193 243 69 60 299 212 131 133 4 268 278 183 151 28 137 109 87 66 94 172 142 124 82 32 205 95 178 18 260 210 79 119 84 230 202 285 112 211 92 224 127 275 221 153 80 8 152 220 241 189 115 154 104 54 0 89 274 182 258 177 75 226 38 283 207 209 9 19 120 6 227 259 201 43 74 53 223 256 208 187 143 100 36 52 58 203 144 76 78 47 247 25 102 198 232 222 279 272 248 175 282 156 165 1 99 286 26 167 295 17 270 265 93 255 146 188 215
Nodes affected in level 3 (10) : 97 218 48 145 292 35 277 245 191 116
Total number of affected nodes : 300

6. Results and discussion

Datasets	K Shell		Local Degree Centrality		Page Rank	
Dataset 1 300 Nodes	Level 1	46	Level 1	111	Level 1	117
	Level 2	166	Level 2	167	Level 2	160
	Level 3	76	Level 3	12	Level 3	13
	Level 4	2				
	Total Affected	290	Total Affected	290	Total Affected	290
Dataset 2 200 Nodes	Level 1	50	Level 1	107	Level 1	107
	Level 2	131	Level 2	83	Level 2	8
	Level 3	9	Level 3	83		
	Total Affected	190	Total Affected	190	Total Affected	190
Dataset 3 300 Nodes	Level 1	45	Level 1	44	Level 1	115
	Level 2	179	Level 2	77	Level 2	165
	Level 3	65	Level 3	61	Level 3	10
	Level 4	1	Level 4	45		
			Level 5	20		
			Level 6	7		
	Affected	290	Affected	254	Total Affected	290
Dataset 4 300 nodes	Level 1	45	Level 1	111	Level 1	115
	Level 2	179	Level 2	167	Level 2	165
	Level 3	65	Level 3	12	Level 3	10
	Level 4	1				
	Affected	290	Affected	290	Affected	290

First, we detected top 10 influential nodes using algorithms like page rank, local degree centrality and k shell. These nodes are considered to be present at level-0 and serves as input to cascading algorithm. Then, we applied independent cascading algorithm to find number of infected nodes and cascade iterations. In the above table, we compared those algorithms based on number of infected nodes and cascade iterations. We found that page rank algorithm performs better compared to other algorithms since the all the nodes are infected and number of levels obtained are less compared to other algorithms. Thus, we conclude that page rank algorithm suits better to detect influential nodes.

7. References

1. R. Narayanam, Y. Narahari. A Shapley value-based approach to discover influential nodes in social networks. IEEE Transactions on Automation Science and Engineering, PP(99). Pages 1-18. ISSN 1545-5955.
2. Xiaoying Pang, Shanwei Yan, David Zucker, Attack Tolerance and Resiliency of Large Complex Networks.
3. A. Flexman, A.Frieze, J. Vera, A Geometric Preferential Attachment Model of Networks, WAW 2004.
4. Surajit Dasgupta, Chandan Prakash, Intelligent Detection of Influential Nodes in Networks, International Conference on Electrical, Electronics and Optimization Techniques(ICEEOT)-2016.
5. C.K Tse, J.Liu, F.C.M. lau, A Network Perspective of the Stock Market, Journal of Empirical Finance, 2010.
6. S. Cheng, H. Shen, J. Huang, W.Chen, X.Cheng, Influence Maximization via Finding Self Consistent Ranking, arXiv:1402.3939vl, 2014
7. J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance Cost-effective outbreak detection in networks, 2007.
8. M.U. Ilyas, H.Radha. Identifying Influential Nodes in Online Social Networks Using principal Component Centrality. IEEE International Conference on Communications (ICC), 2011. Pages 1-5. ISSN 1550-3607.