CS512 Project Group 11- Fall 2019

Yaxin Zhu(yz956)
Rutgers University
Piscataway, NJ, USA
Email: yz956@scarletmail.rutgers.edu

Xiaochen Gao(xg142)
Rutgers University
Piscataway, NJ, USA
Email: xg142@scarletmail.rutgers.edu

Jiawei Wu(jw1308)
Rutgers University
Piscataway, NJ, USA
Email: jw1308@scarletmail.rutgers.edu

***Abstract*— In this project, we build a model to analyse the social network basing on a Twitter data set. In the model, we implement several algorithm and their visualization to dig out useful information from the network. The structure can be applied to various network and give useful information to communication studies and data mining**

I. Project Description

Our project is about modeling, clustering and analyzing social networks. It belongs to a novel application of known algorithms, including clustering algorithm, shortest path algorithm, etc. In general, our project uses a data set based on Twitter to create a model about social network of Twitter User. Then, using this model, we could detect the community of users and identify the central individual. Our project could be used to let Twitter users find out who to follow or which community they belong to and let advertisers or other enterprise users view the user community that best matches their needs, and the most influential users in the community. The most competitive and novelty of this application is that we add a lot of extra attributes to help us model and analyze the data, including the time the user posts and the content of the hashtag.

The whole project is planned to be completed in one month, but we have encountered many obstacles in the implementation process. The biggest one is that the relationship data between users is too large and sparse. We have to do a lot of work to extract useful information from the data set. After overcoming many difficulties, we formed the timeline in stage1 which is feasible and can be completed on time.

The project has four stages: Gathering, Design, Infrastructure Implementation, and User Interface.

A. Stage1 - The Requirement Gathering Stage.

- System description
  Our system mainly consists of 3 parts: data collection, data processing as well as analysis, and visualization, in a goal of providing social network data analysis reports to users. In processing and analysis level, our system can be applied to different scenarios including personalized interaction management, community detection and central individual identification.
- User types
  - admin Having access to data of users connection, tweet propagation path and so on collected from public social network. Being able to process those data, generate instant diagrams and give analysis.
  - enterprise user Being able to manage personalized interaction status by authorizing the system to collect their social media account data. Having access to all diagrams generated from public data.
  - personal user Being able to manage personalized interaction status by authorizing the system to collect their social media account data. Having access to some specific kinds of diagrams generated from public data.
- Interaction modes
  The enterprise and personal users authorize the system to utilize their user data. The admin controls the system in data processing and analysis. Then enterprise and personal users have access to analysis outcomes provided by the system.
- Real world scenarios1
  - Scenario1 description: Some users may be interested in current relationships even communities formed among social media users. Our system summarizes these communities via analyzing a graph of latest interactions - mainly retweeting action, during tweets propagation. After that users will obtain a clear picture of some communities for further analysis.
  - System Data Input for Scenario1: Social network
  - Input Data Types for Scenario1: Adjacent matrix
  - System Data Output for Scenario1: label of each users, visualized diagram
  - Output Data Types for Scenario1: 2-dimension matrix, elements being user objects, graph
- Real world scenarios2
  - Scenario2 description: Users are able to find the central point - a user with highest influence or centrality in any community of the social network. The so-called centrality can be measured in different ways and we plan to choose PageRank measurement in this work.
  - System Data Input for Scenario2: Social network or specified communities
  - Input Data Types for Scenario2: Adjacency Matrix, user id
  - System Data Output for Scenario2: Central points of the community of the user, visualized diagram
  - Output Data Types for Scenario2: graph vertix, graph
- Timeline and division

- Data modeling: Extract information and model by Jiawei Wu.
  Completed before November 8th.
- Algorithm implementation: Implementation of core algorithms, including clustering algorithms, shortest path algorithms, etc, by all team members.
  Completed before November 18th.
- Interface implementation: Implementation of project interface by Xiaochen Gao and YaXin Zhu.
  Completed before November 25th.
- Testing: Testing by Jiawei Wu.
  Completed before November 25th.
- Documentation: Documentation by Yaxin Zhu.
  Completed before November 30th.
- Evaluation: Evaluation by Jiawei Wu.
  Completed before November 30th.
- Project report: Report by all members.
  Completed before November 30th.
- Power point presentation: Slides by Xiaochen Gao.
  Completed before December 2nd.

B. Stage2 - The Design Stage.

In the following part, we give analysis of the three main algorithms we use - two for community detection and one for centrality computing. For each algorithm, we draw its flow diagram, analyse its time and space complexity and show the data structure we use to implement the algorithm.

Louvain:

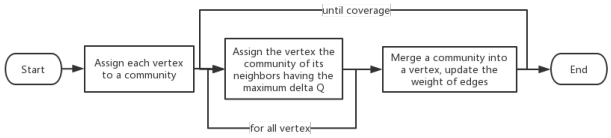
- Short Textual Project Description.
  Louvain is a hierarchical clustering method[1]. At beginning, we assign all vertices to different clusters. In each iteration, we traverse all the vertex and assign it to the cluster of its neighbor with the maximum ΔQ. After that we merge the vertices in a cluster into a single vertex and update the weight of edges. We repeat these process until coverage.

  The definition of ΔQ is given in equation 1.

$$\Delta Q = \left[\frac{\sum_{in}+k_{i,in}}{m} - \left(\frac{\sum_{tot}+k_i}{2m}\right)^2\right] - \left[\frac{\sum_{in}}{2m} - \left(\frac{\sum_{tot}}{2m}\right)^2 - \left(\frac{k_i}{2m}\right)^2\right] \quad (1)$$

  - ΔQ: Gain in modularity of a community c when we add vertex i in it. Larger ΔQ means better it is to add i to community c
  - $\sum_{in}$: Sum of weight of all edges in the community c
  - $\sum_{tot}$: Sum of weight of all edges connected to vertices in the community c
  - k_i: sum of weight of all edges connected to vertex i
  - $k_{i,in}$: Sum of weight of all edges connected to community c and vertex i

  Time complexity is hard to calculate as we are not sure when the algorithm will coverage. We estimate that the average time complexity is $O(|V|^2)$
  Space complexity of Louvain is highly related to the data structure we use, the space complexity of Louvain we implement is $O(|V|^3)$
- Flow Diagram.



- High Level Pseudo Code System Description.

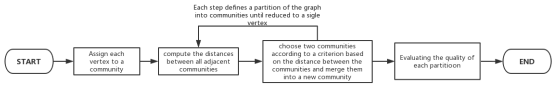
Algorithm 1: Louvain

Input: Graph G
Output: Label of Community L
Initialize each vertex of G to its own community
while *L not coverage* **do**
  for *each vertex i in $|V|$* **do**
    Compute $\Delta Q(i,j)$ for each of its neighbor j
    if $Max(\Delta Q) = \Delta Q(i,j) > 0$ then
      Add vertex j to the community of i
    end
  end
  Merge all communities into single vertices
  Update Weights of Edges
end
Mark Label L of each vertex

- Algorithms and Data Structures.
  - We use a $|V| \times |V|$ adjacency matrix to store the edges.
  - We design a class named Community to store the community we get after each iterations. Each community stores the vertices it contains and some basic information to compute ΔQ.

Walktrap:

- Short Textual Project Description.
  Walktrap is based on the following intuition: random walks on a graph tend to get "trapped" into densely connected parts corresponding to communities[2]. Using this, the algorithm define a measurement of the structural similarity between vertices and between communities, thus defining a distance r_{ij}. Then, it can be used in a hierarchical clustering algorithm(merging iteratively the vertices or communities which minimize the function $\Delta\sigma(C_i, C_j)$ into new communities). One obtains this way a hierarchical community structure that may be represented as a tree.
  This algorithm runs in time $O(|V|^2 \cdot E)$ and space $O(|V|^2)$ in the worst case, and in time $O(|V|^2 \cdot \log|V|)$ and space $O(|V|^2)$ in most real-world case.
- Flow Diagram.



- High Level Pseudo Code System Description.

Algorithm 2: Walktrap

Input: Graph G
Output: Partition P

for *each* $v \in V$ **do**
  Assign v to a community C_i
end
for $i = 1$ *to* n **do**
  compute the probability vector $P^t_{C_i\bullet}$
end
$\mathcal{P}_1 = \{\bigcup_{i=1}^{n} C_i\}$
for $k = 1$ *to* n **do**
  forall C_i, C_j *such that* $(C_i, C_j) \in E$ **do**
    compute $\Delta\sigma(\mathrm{C}_i, C_j)$
  end
  Choose two communities C_1 and C_2 in $\mathcal{P}_k$ that minimize $\Delta\sigma(\mathrm{C}_i, C_j)$
  Merge them into a new community $C_3 = C_1 \cup C_2$
  Create the new partition $\mathcal{P}_{k+1}$ = $(\mathcal{P}_k \backslash \{C_1, C_2\}) \cup C_3$
end
Evaluating the quality of partition $\mathcal{P}_k$ ($k = 1$ to n), and choose the best partition

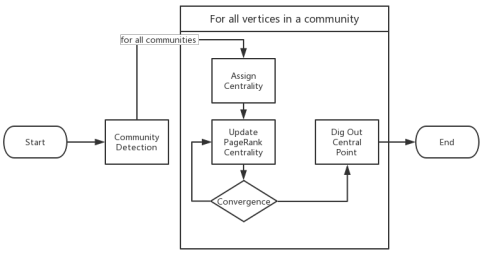
- Algorithms and Data Structures.
  - We used a $|V| \times |V|$ adjacency matrix to store the nodes and the edges.
  - We also used a $|V| \times |V|$ matrix to store the probability vectors $P^t_{C_i\bullet}$
  - And we used a tree to store each partition $\mathcal{P}_k$

PageRank:

- Short Textual Project Description.
  Besides community detection, we plan to measure centrality of each node(in real scenario, user) with PageRank centrality, which is computed iteratively as[3]:

$$C_i = \alpha \sum_j a_{ji} \frac{C_j}{L(j)} + \frac{1-\alpha}{N}$$

  where $L(j) = \sum_i a_{ji}$ is the number of neighbors of node j.
  Time complexity: $O(|V| + |E|)$
- Flow Diagram.



- High Level Pseudo Code System Description.

Algorithm 3: PageRank Centrality Algorithm

Input: Sub-graph G_c of a community, dumpling factor α
Output: Node with highest centrality

Initialize C_i **of each vertex** i **in** G_c **to** $\frac{1}{|V_c|}$
Compute $L(i)$ **for each vertex** i
while *not coverage* **do**
  for *each vertex* i *in* G_c **do**
    $C_i \leftarrow C_i + \frac{1-\alpha}{|V_c|}$
    for *all neighbors* j *of* i **do**
      $C_i \leftarrow C_i + \alpha \cdot \frac{C_j}{L(j)}$
    end
  end
end
Return $\max(C_i)$ **for all vertices** i

- Algorithms and Data Structures.
  We use a $|V| \times |V|$ adjacent matrix to store the nodes and edges and a $|V|$ vector to store the updating centrality of each vertex.

C. Stage3 - The Implementation Stage.

The language we use to implement this project is python, and we use following package in our program.

- numpy: basic calculation
- tkinter: implement user's interface
- networkx: plotting structure of graph
- matplotlib, PIL: image visualization

In the following part, we will describe the dataset we use and the result of our algorithm.

- Datasets: We choose data set callsed "ego-Facebook", it is a dataset wildly used in social network study. And following are some basic information of this dataset.

Nodes	4039
Edges	88234
Average clustering coefficient	0.6055
Number of triangles	1612010
Diameter (longest shortest path)	8
90-percentile effective diameter	4.7

- Sample output: The algorithm labels all the vertices and stores it in a .csv file, we have part of our result as

following:

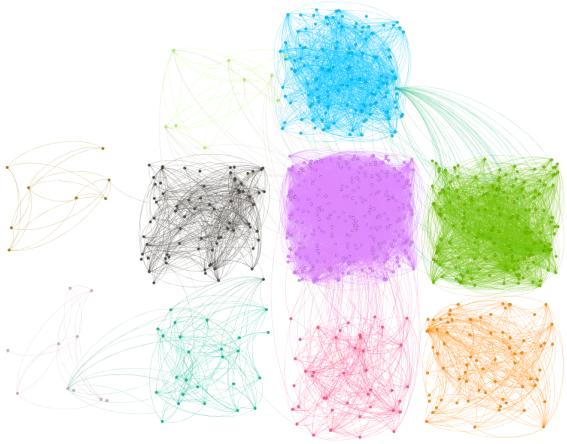
1	0	7
2	1	7
3	2	6
4	3	7
5	4	1
6	5	5
7	6	0
8	7	7
9	8	2
10	9	7
11	10	7
12	11	7
13	12	7
14	13	0
15	14	6

4024	4023	53
4025	4024	53
4026	4025	53
4027	4026	55
4028	4027	53
4029	4028	55
4030	4029	53
4031	4030	55
4032	4031	53
4033	4032	53
4034	4033	54
4035	4034	53
4036	4035	53
4037	4036	55
4038	4037	56
4039	4038	53

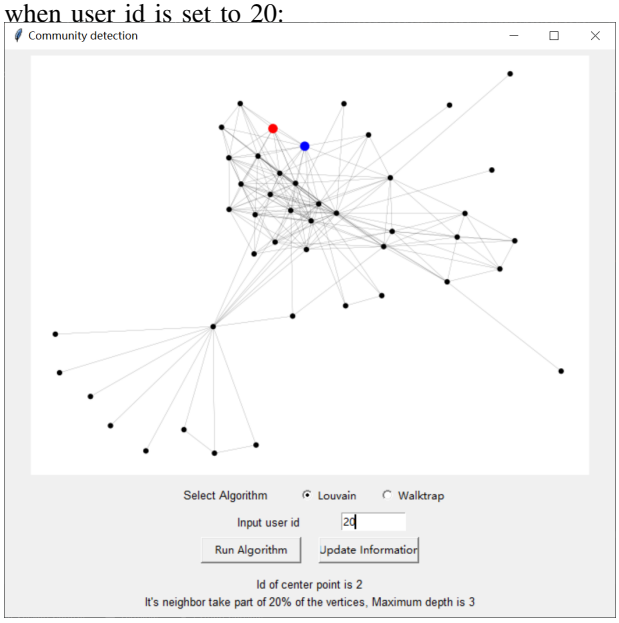
It is hard to analyse the performance of community detection algorithm as we do not have a good indicator to evaluate its performance.
We evaluate the performance of our algorithm algorithm in the following way:

- Firstly, we run our algorithm on the dataset.
- Secondly, we pick 10 communities detected by the algorithm randomly.
- Thirdly, we print image of edges and verteies in the chosen communities.
- Finally, we decide whether our community detcetion algorithm is good by judging whether there are much more edges in the same communities rather than between two communities.

The following is the image we generated via Gephi[4]:



- Working code: There are two main tasks of our algorithm, the first one is divide the vertices into several communities, the second is it present the community structure of a specific user and tell what is the center point in the community.
We are already verified the first task in the last part, and we will show its performance of the second task in the coming part.
- Demo and sample findings: The algorithm succeed in finding the center point of a community based on the user id given. The following part is the output of algorithm when user id is set to 20:

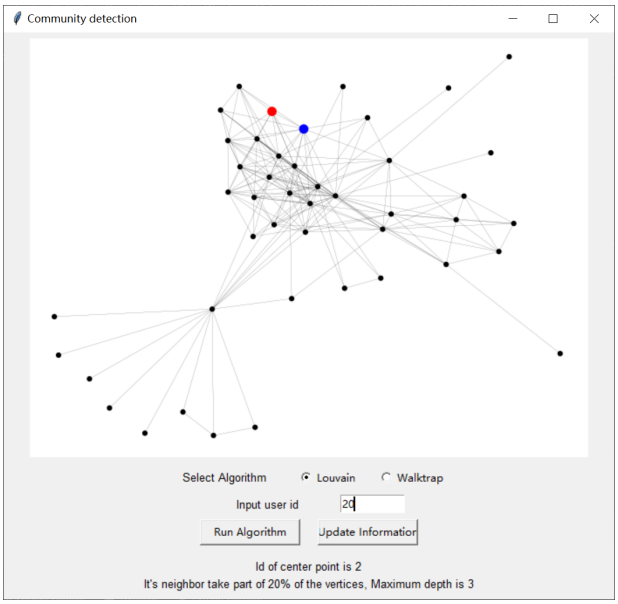


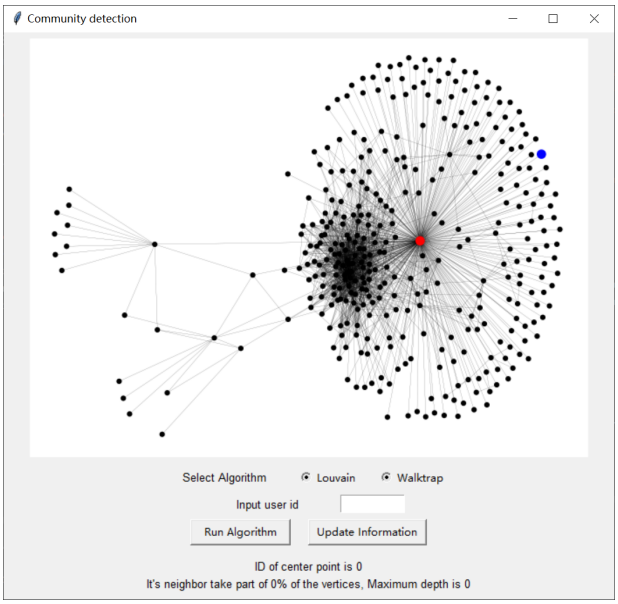
The blue point mark where the user is and the red point mark where the center point is. The algorithm print out the structure of the community clearly and calculate the maximum depth of the center point and the percentage of its neighbors in all the vertices.

- Data size: Among 4039 nodes, the algorithm detect 56 communities in total.
- In most of the communities, the maximum depth(largest shortest path between center point and other points)of the center point is always 2 or 3. And the percentage of its neighbors in all the vertices is always over 50%.

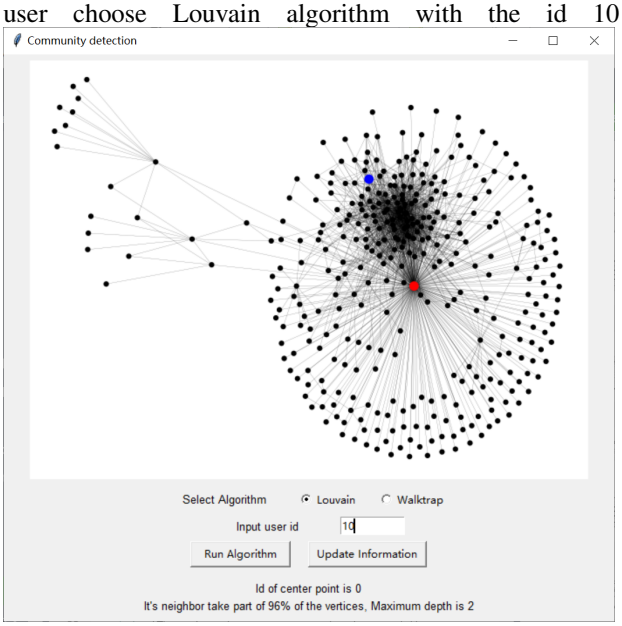
D. Stage4 - User Interface.

We design an user interface for our program, and it is shown as following:





- Two different sample navigation user paths through the data exemplifying the different modes of interaction and the corresponding screenshots. The user choose Louvain algorithm with the id 10



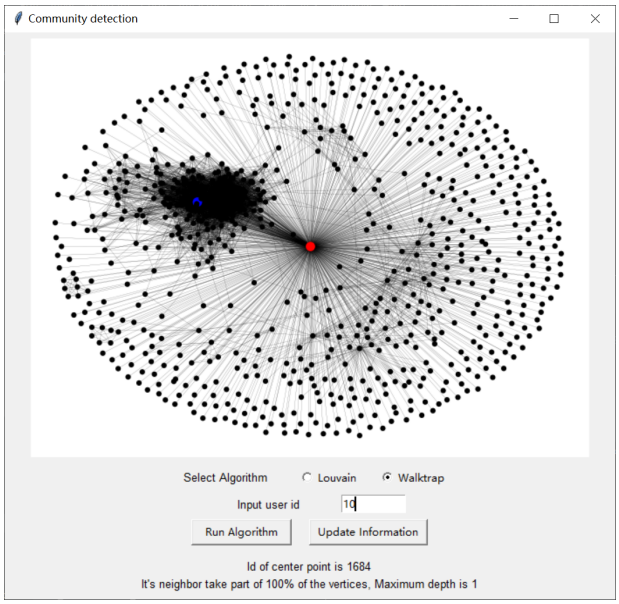
The user choose Walktrap algorithm with the id 10

- The user can choose which algorithm the program use to perform community detection by clicking the radio button, and we also have an entry for user to input his or her id.
- The Program is able to warn user if he or she has information unchosen or unfilled by popping out a warning window

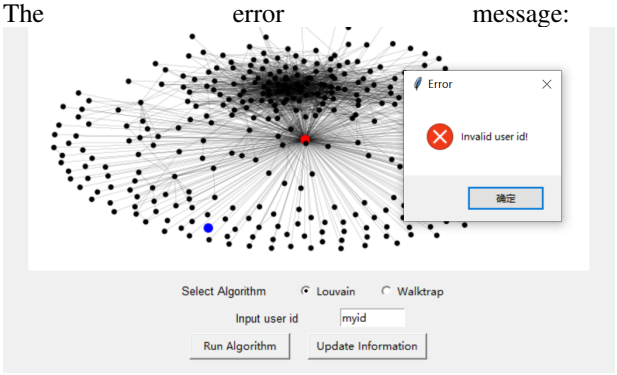
- The Program is able to handle the error occurred by a invalid input, a window will pop out with the error message to inform the user

- The progeam is able to generate human readable result[5].
- After the result is generated, the Program will pop out an notification to remind user to click the ”update information” button to update the result shown in the user interface

- The initial UI screenshot

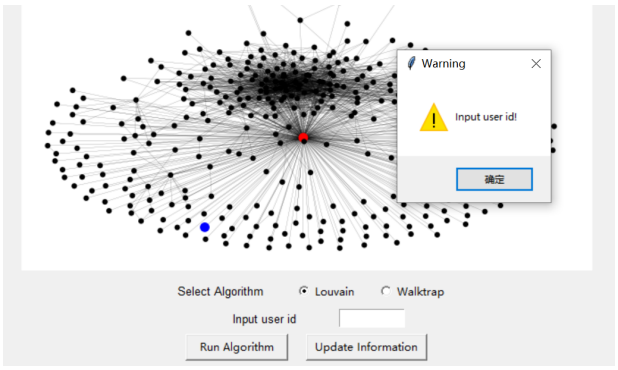


- The error messages popping-up when users give invalid input:
  - The error message:



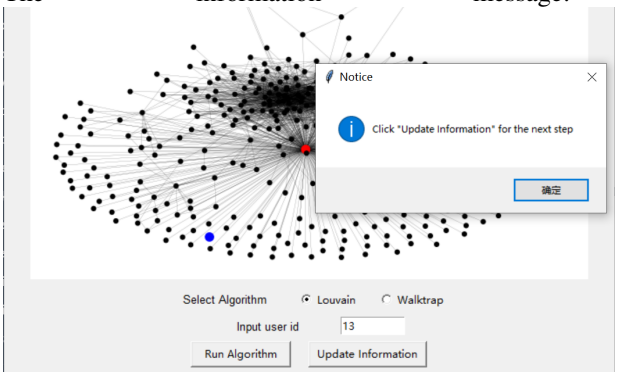
  - The error message explanation: The user id must be an non-negative integer within the range of all user id.

- The warning messages popping-up when users have information unfilled:
  - The warning message:



  - The warning message explanation: User must choose an community detection algorithm and input the user id.

- The information messages or results that pop-up in response to user interface events.
  - The information message:



  - The information message explanation and the corresponding event trigger After the program finish finding out all the communities, it will inform user to update the user interface.

REFERENCES

[1] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.

[2] P. Pons and M. Latapy, "Computing communities in large networks using random walks," in *International symposium on computer and information sciences*. Springer, 2005, pp. 284–293.

[3] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer networks and ISDN systems*, vol. 30, no. 1-7, pp. 107–117, 1998.

[4] M. Bastian, S. Heymann, and M. Jacomy, "Gephi: an open source software for exploring and manipulating networks," in *Third international AAAI conference on weblogs and social media*, 2009.

[5] A. Hagberg, P. Swart, and D. S Chult, "Exploring network structure, dynamics, and function using networkx," Los Alamos National Lab.(LANL), Los Alamos, NM (United States), Tech. Rep., 2008.