Abstract

WRITE A GOOD ABSTRACT

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Using graphs as representation of real world network and graph traversal algorithms to iterate over graphs are seen in different scientific domains [2]. Graph simulations have been used to study how a disease would spread through a social network[3], predict how efficient viral marketing could be and how a new trend would spread through a community [4]. Seeing how data propagates through a network is known as information diffusion. We can compare the diffusion as a company trying to promote a new product to the public, the public would be the network and each node can either adopt the new trend, or ignore it. One important aspect of information diffusion is the starter set, or the seed nodes. A seed node is the initial starter node that is activated, much like a person reciving a free sample from a company trying to promote a new product. By giving someone such a free product it can help promote the product, but choosing which person to give the sample to is important. Choosing someone with few friends and few connection would be a waste of product, giving too much would hurt the companys profit and giving too little would result in not enough exposure.

To select the best seed nodes is an NP-hard problem [4]. One of the solution is the greedy algorithm [5] proposed by Kempe et al. The greedy algorithm goes through all the nodes and computes the effect on the network that nodes had. The greedy algorithm takes the k top most influential nodes as the seed nodes. When a new seed node is to be added to the starter set S, the algorithm iterate over all the node in the network that is not in S and compute the spread. This project is a collaboration with the Arctic Green Computing Group.

1.0.1 Assignment text

"Information diffusion is a field of network research where a message, starting at a set of seed nodes, is propagated through the edges in a graph according to a simple model. Simulations are used to measure the coverage and speed of the diffusion and are useful in modeling a variety of phenomena such as the spread of disease, memes on the Internet, viral marketing and emergency messages in

disaster scenarios.

The effectiveness of a given spreading model is dependent on the initially infected nodes, or seeds. Seed selection for an optimal spread is an NP hard problem and is normally approximated by selecting high-degree nodes or using heuristic methods such as discount-degree or choosing nodes at different levels of the k-core.

This project will e xplore the feasibility of hardware accelerated seed selection in large graphs as a variation of breadth-first search (BFS) where the decision to visit and infect a child node relies on the outcome of a coin flip. The student is expected to conduct a thorough review of the literature on seed selection algorithms, diffusion models, with emphasis on parallelisation and hardware acceleration and cache models to reduce memory bottlenecks. Hardware design and simulations are possible if time permits."

I have chosen to interpret the task as to understand the fundamental concepts of graph theory, the core concept of information diffusion and seed selection. I have chosen to focus on one specific model of information diffusion which is the independent cascade model, which will be further discussed in Section 2. I have been looking at different proposed solutions that might be able to improve the independent cascade model and the seed selection.

1.0.2 Report setup

I have in this report performed a similar experiment as the experiment done in [4] with the greedy algorithm, the high degree algorithm, the random algorithm and a modified version of the greedy algorithm we are calling independent greedy algorithm in **??**. We will go through some of the notation commonly used in graph theory in 2 and we will look at how we can perform standard graph algorithm such as BFS as matrix-vector multiplication. We will propose some idea that could potentially improve the seed selection algorithm, the independent cascade model or the diffusion of information.

1.0.3 Detail of solution

Chapter 2

Background

In this chapter, we will look at the fundamental concepts and background information for the different diffusion model, the seed selection problem and performing breadth-first search using matrix multiplication. This chapter will contain notations that we would use throughout the report. One aspect we will focus on, is how we can perform graph algorithm such as breadth first search as matrix multiplication. The motivation for transforming breadth first search as matrix-vector multiplication is that displaying the graph algorithm as a matrix multiplication can display the data access pattern for the algorithm and can be readily optimized [6]. We will look at the independent cascade model, which is a special case of breadth first search [2]. By looking at how to improve BFS, we can apply such optimization to ICM and the seed selection algorithm.

2.1 Network terminology and glossary

The fundamental unit in a network is a *Vertex*(*pl.vertices*), sometimes called a node. For this report, both vertex and node will be used. The "bridge" or the line connecting two vertices is called an *Edge*, and is shown in Figure 2.1. Different networks have different types of edges, some are *Directed*, while others are *undirected*. A directed edge is an edge that runs in only one direction (such as a one-way road between two points), and undirected runs in both directions. Directed edges can be thought of as sporting arrows indicating their orientation, while undirected edges have no such orientation. A graph is directed if all of its edges are directed.

Each node has a value that is called *Degree*. Degree for a vertex v_1 is the amount of edges connected to v_1. Note that the degree is not necessarily equal to the number of vertices adjacent to a vertex, since there may be more than one edge between any two vertices. A directed graph has both an in-degree and an out-degree for each vertex, which are the numbers of in-coming and out-going edges respectively. The *Component* to which a vertex belongs is that set of vertices that can be reached from it by paths running along edges of the graph.

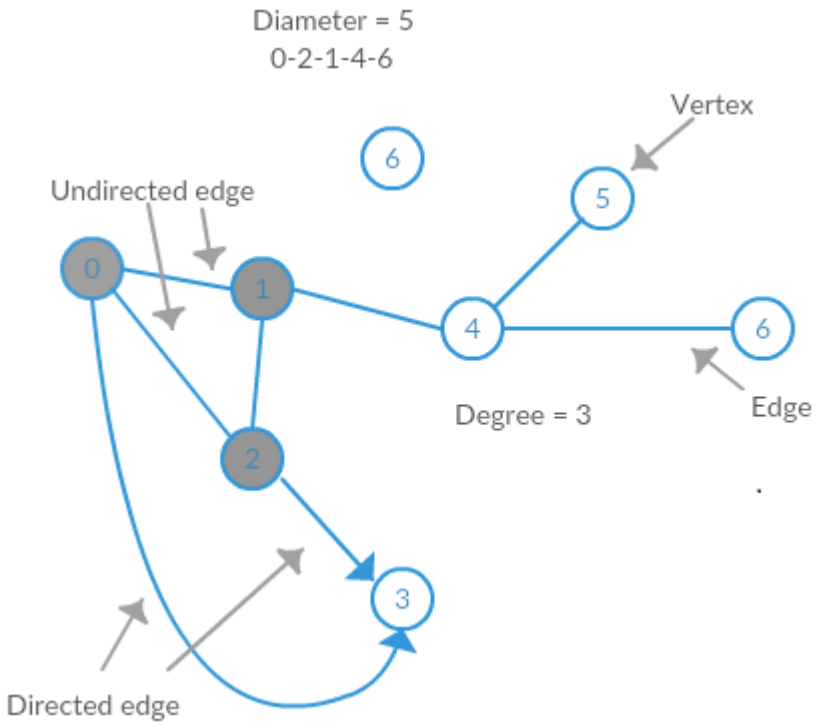


Figure 2.1: Simple network

In a directed graph a vertex has both an in-component and an out-component, which are the sets of vertices from which the vertex can be reached and which can be reached from it.

A *Geodesicpath* is the shortest path through the network from one vertex to another. Note that there may be and often is more than one geodesic path between two vertices. The *Diameter* of a network, however, is the length (in number of edges) of the longest geodesic path between any two vertices. A few authors have also used this term to mean the average geodesic distance in a graph, although strictly the two quantities are quite distinct.

2.2 Network

A *network* is a collection of vertices and edges [7]. In the real world, multiple systems takes the form of networks around the world, examples are the internet, the World Wide Web, social media like Facebook, twitter etc. There are different types of network. These include from *social network* [7], $information\ networks$ [7], *technological networks* [7] ,and *biological network* [7]. Different network have different properties, we will look at those most relevant to social networks.

2.2.1 Social network

A social network is a set of individuals connected to each other via some form of contact or interactions [7]. The nodes are people while the edges are the connections between people. The social network display information regarding connection, interaction or location of a set of people.It forms patterns regarding friendships, business interactions between companies and families history. The social network is often used in social science [7]. Some notable experiments are [8], which we will discuss more in **??**

2.3 Network properties

Network properties are different characteristic properties that network displays. We will focus on those that are more relevant to social network and how it is relevant to the ICM, data diffusion and seed selection.

2.3.1 The small world effect

The small world effect was first demonstrated by Stanley Milgram in the 1960s during his famous letter passing experiment [9]. The experiment was about passing letters from person to person to reach a designated target with only small steps. For the published case, the chain was around six [10], meaning there were only six passes necessary for the letter to reach its destination. This shows us that for most pair of vertices in a network can reach each other with a short path. A more precise wording is that "Networks are said to show the small world effect if the value of l scales logarithmic or slower with the network size for a fixed mean degree." [7]. We have defined l to be the mean geodesic distance between vertex pairs in a network.

The small world problem can be summarized as: "what is the probability that any two people, selected arbitrarily from a large population, such as that of the United States, will know each other?". [**?**]. This in itself is not that interesting, the [**?**] asked even tho person a and z does not know each other, do they have a set of individuals $\{ b_1, b_2, ...b_n\}$ who are mutual friend or even a "chain" of such individual($a - b - c - ... - y.z$).

For the data diffusion problem, this kind of effect would result in that the diffusion through a network would need around 6 steps to have traveled through the entire network if the transition probability is high. Meaning that most node can reach each other through a relatively small step.

2.3.2 Transitivity/Clustering

In graphs and networks, there would often have a special connection pattern called *triangles*. Triangles is where three Vertices : v_a, v_b, v_c is all connected to each other as shown in Figure:2.2. We can look at such a connection as person A is friends with B and C. There are a chance that B and C is friends with

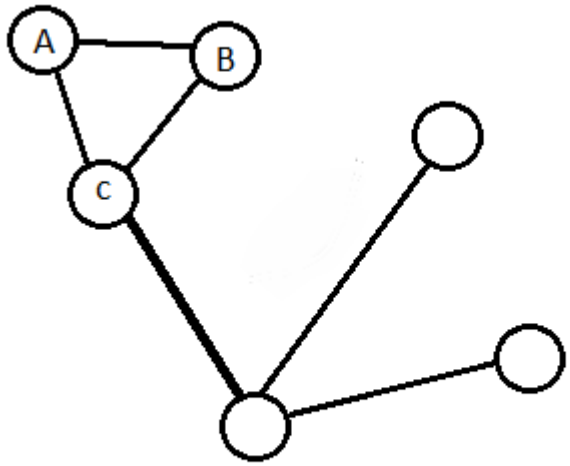


Figure 2.2: Examples of triangles in networks

each other too. Transitivity is used to determent how many such components is present in the graph.

For data diffusion and seed selection, this would mean that picking nodes that are neighbor to each other, would have a smaller spread then picking two not neighboring vertices. By picking two nodes that are connected to each other, they would likely share common neighbors and thus having a smaller reach.

2.3.3 Degree distribution

As mentioned earlier, each node have a degree. The degree distribution shows how the different degrees in the network is distributed. From the degree distribution we can see how many nodes have specific degree. Different networks have different shaped degree distribution.

For random graphs, the distribution would most likely be a Poiison distribution or binomial. For social graph, the degree distribution is often in the shape of exponential distribution. For the information diffusion, this distribution shows us how many high degree nodes there are in the network, those node would likely be high prioritized node and have a large impact. One of the algorithm uses the distribution to select the seed nodes. We will discuss this in later sections.

2.3.4 Degree correlations

As **??** show, the network have a degree distribution with few high degree nodes and many low degree nodes. One interesting properties is the degree correlations. Degree correlations is how the high degree and low degree nodes connects to each other. One question is if high degree nodes tends to connect to other high degree nodes, or do they prefer to connect to low degree nodes. It turns out that both incidents is found in networks [**?**]. Most social networks are assortative, meaning vertices have a selective linking, where high degree vertex tends to connects to other high degree vertex, while technological network and biological network are most likely disassortative [**?**].

For data diffusion, this kind of behavior would result in wasting a seed by picking majority of high degree node for starting seed, since most of them would be connected to each other. One solution is by mixing the selection, choose some percentage to be high degree, and some with lower degree.

2.3.5 Network resilience

Most network model, there is the need to remove nodes from the network. Removal of a node can have no effect on the network, or it can be devastating. Network resilience looks at how the network can resist to such a removal. There are two different removal scheme, the random removal where nodes are randomly picked and removed, or the targeted removal where specific nodes are removed depending on the criteria.

The experiment mentioned in [**?**] by Albert et al showed that for a subset of network representing the internet and the world wide web, targeted removal had a larger impact then random removal. The targeted removal removed the highest degree nodes from the network, and the random removal removed nodes randomly. The random removal had a minimal effect on the network, while the targeted removal had a much larger impact, the mean vertex-vertex distance increased. They proposed that the internet was highly resilient to random removal, while much more vulnerable to targeted removal.

There are other studies that proposed a different interpretation about the data found.

In an example of information diffusion, a targeted removal would result in massive change to the diffusion path. By removing high degree node would result in remove influential nodes and limit the spread of the data. Removing important node connection two different community would result in isolation and no path towards other community. For our case, we assume that the network is a static, unchanging network.

2.3.6 Community structure

One properties that is often observed in a social network, is the community structure. The community structure is where a group of vertices having high density of edges with each other, while having low density of edges to other

"community". We can see an example of the community structure clearly displayed from [11]. Where we can see the playground was divided into different groups.

This type of community structure would have a large impact on how the algorithm would select a seed for information diffusion. If all the seed would be selected in one community, the probability of spreading over to other community would be smaller. This is of course dependent on how many intercommunity connections there are in hte network.

2.4 Breadth First search

Breadth first search is an tree traversal algorithm. BFS start at the root node v_r. The algorithm then stores all v_rs children node in an *queue*. The algorithm then takes the first node from the queue, v_1 and stores all the children node to v_1 in the back of the queue, this process continuous until the queue is empty and all the nodes have been iterated over.

Breadth first search is common graph iteration algorithm. The breadth first search is often limited by the irregular memory access where the algorithm have too find the data stored in different space in the memory. As mentioned earlier, Independent cascade model is one special case of the Breadth first search.

Algorithm 1 Breadth First Search

```
1: dist[∀v ∈ V] = −1; currentQ, nextQ = ∅
2: step = 0; dist[root] = step
3: ENQUEUE(nextQ,root)
4: while nextQ ≠ ∅ do
5:     currentQ = newxtQ; nextQ = ∅
6:     step = step + 1
7:     while currentQ ≠ ∅ do
8:         u = DEQUEUE(currentQ)
9:         for v ∈ Adj[u] do
10:            if dist[v] == −1 then
11:                dist[v] = step
12:                ENQUEUE(nextQ, v)
   return dist
```

2.5 Data diffusion

Data diffusion is looking at how information is propagated through a network or a graph. An example would be how a new Internet meme, a new product or how a new disease is spread through a community. The process consist of a set of starter nodes, which we will call seed nodes, that are "infected". During each time-step, there are a percentage p_g that the "infected" nodes

would "infect" its neighbors. We would need to first pick out a set of initial starter node. These k seed nodes is a set of k node that in the initial time-step is infected. They will pass on the information/infection during each time-step and the information/infection will propagate through the network.

2.6 Basic Diffusion Models

When we talk about data diffusion, we can look at how disease or technological innovations would spread through a social network. We can simulate those kind of behavior with different diffusion models. There are two basic diffusion models used to simulate the propagation of information through a network [4], the *linear threshold model*(LTM) and the *independent cascade model*(ICM) [4].

This process is similar to how a new product is promoted via social media. Each node is a person that can either buy the new product(activated), or ignore it(inactive). Each person will then see their friend promote the new product and potentially buy the promoted product. There are several different criteria for each person to buy the product(activates). They can either have a percentage chance to be affected by the advertisement(ICM), or they will only be interested if a percentage of his or hers friends have promoted it(LTM). Some people might have a bigger friend circle then other(high degree), while other have bigger impact on a person(large p_x). Some might be harder to promote too(weighted edges), while some user have no friend(singletons). The different model we will focus on, is the independent cascade model and the linear threshold model.

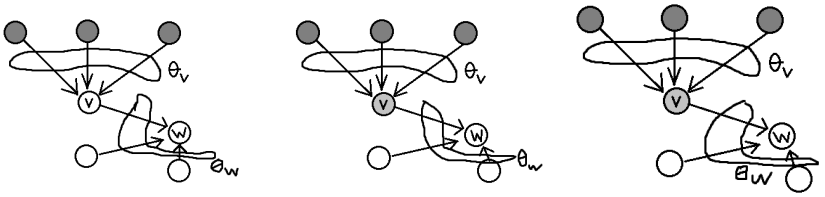
2.7 Linear threshold model

The linear threshold model uses a threshold θ_v between the interval [0,1], which represent the fraction of $v's$ neighbors that need to be active to activate node v, this is known as the weight of v, b_v. In this case, lets assume that the weight of all the nodes in this model is 0.5, meaning over half of its neighbor must be activated for the node to be activated. As we can see in Figure 2.3a, current node v will get activated when $b_v <= \theta_v$.In figure **??** we can see that v is now activated. The next time-step, Figure 2.3c, node w is checking if it will too, be activated. We can see here that$b_w! <= \theta_v$, so node w will not be activated.

We can look at the linear threshold model as a cosmetic company trying to promote their new product via social media. Each users of the product would display the new cosmetic product through social media. Each users would then be exposed to the product through their friends update. Each user would adopt the new product after seeing a percentage of their friends using the product.

2.7.1 Independent cascade model

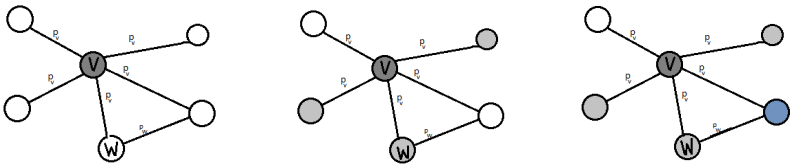
The independent cascade model(ICM) have a local or global probability for determining of activation, p_v. In figure 2.4, we have private probability. The



(a) Step 1 (b) step 2 (c) step 3

Figure 2.3: Linear Threshold mode

probability $p_v! = p_w$. In 2.4a, the node v is activated, during the next time-step, node v have activated three neighbor. In the next time-step 2.4c node w was able to activate the blue node. For a ICM with global probability, each node would then have the same probability to activate its neighbor. As we can see from Figure 2.4, each neighbor to v is activated individually. As in 2.4b, only three nodes were activated. In ICM, each node can only try to activate the neighbor once. In figure 2.4b, the node that was not activated by v, got activated in 2.4c by the node w.



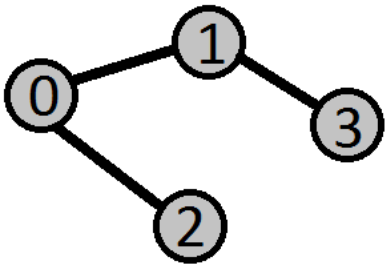
(a) step 1 (b) step 2 (c) step 3

Figure 2.4: Independent cascade model

We can compare the ICM with the same example we have been using, the cosmetic company promoting a product. Each new user have a chance(pp_v) to promote the product to a new user. The new user would then continuous promoting the new product to his/hers friends.

As mentioned earlier, the ICM is a special case of the breadth first search. If we set the transmisison probability to 1.0, meaning there are 100% chance to activate the neighbores, the ICM is equal to BFS. BFS iterates through the graph by appending the children node not examined into the queue. Then examines a new node from the queue and repeating the process until all nodes have been examined. The main difference between the ICM and BFS, is that in ICM, there is a chance that the child node is not added into the queue. During each step, a random "coin toss" is tested to determent if the child node would be added into the queue. Other then that, both algorithms iterates through the network via edges.

(a) The adjacency matrix



(b) The graph corresponding to the adjacency matrix

Figure 2.5: Sparse matrix to graph

2.8 Matrix notations

Nodes and edges are not the only way to present a graph, graphs and networks can be represented as sparse adjacency matrix [6]. We can see that such an idea have been proposed in earlier literature [12]. By representing graphs as sparse matrix, we can often discover different ways to optimize the algorithm, we can have a different structure to store data. The adjacency matrix in particular, is a interesting way to represent the graph. A graph $G = (V, E)$, G have N vertices and M edges, this correspond to a N×N adjacency matrix called A. If A(i,j)=1, then there is an edge from v_i to v_j. Otherwise its 0. In Figure:**??**, we can see how a undirected graph can be represented as an adjacency matrix. To generate a undirected graph as a adjacency matrix, the matrix must be mirrored diagonally, meaning if A(i,j)=1, then A(j,i)=1, if this is not true, then the matrix would be representing a directed graph.

2.8.1 Sparse Matrix

A sparse matrix is a matrix containing few nonzero. Social graph with few edges would often be represented as a sparse matrix. Since sparse matrix only have few non-zero elements, by storing only the non-zero elements, we can have savings in memory.

2.8.2 Breadth First Search as a matrix multiplication.

From [6], we can see that BFS can be recast as algebraic operations. BFS can be performed by applying matrix-vector multiplication over Boolean semirings [2]. The graph is represented as a adjacency matrix A, then for the root node, a

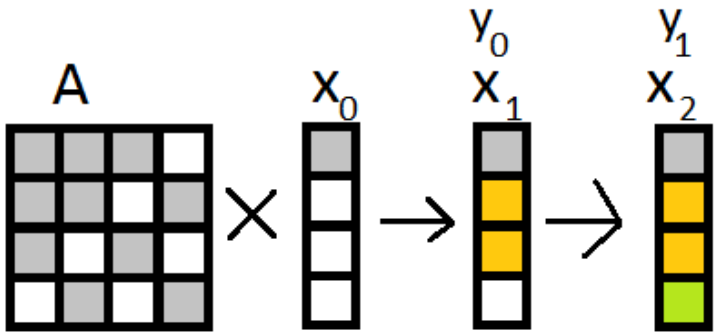


Figure 2.6: BFS on Boolean semiring

vector x(root)=1 is multiplied with the matrix A. $A \times x_0 = y_0$. y_0 is the result after the first matrix-vector multiplication and in the next iteration, $x_1 = y_0$. We can see from the Figure2.6

2.8.3 Semiring

A *semiring* is a set of elements with two binary operations. The two operations are often known as "addition"(+) and "multiplication"(×). As we shown in previous section, the algorithm perform matrix multiplication uses the two operations, multiplications and addition. In [2], the AND and OR operator was chosen instead of the normal addition and multiplication.

2.8.4 BFS to data diffusion

As mentioned before, ICM is a special case of the breadth first search. By modifying the algorithm proposed earlier, we can in theory perform ICM with matrix-vector multiplication. We will discuss more how such an algorithm is in later chapter.

2.9 Seed selection algorithm

The seed selection algorithm, is the algorithm used to select the initial k seed nodes to be chosen at the start of the information diffusion. We can compare it to a new gadget or a cosmetic company trying to promote a new product. By selecting a few influential persons to give a free sample. The new trend would

most likely spread through a *viral marketing* [13]. The seed selection algorithm would be the algorithm to select the few influential individuals to receive this free sample. There are multiple different scheme to choose from, in this section, we will focus on four different algorithms, greedy algorithm, degree algorithm, random algorithm and the independent greedy algorithm.

2.9.1 The greedy algorithm

The greedy algorithm [5] proposed by Kempe et al, is known to be the best algorithm according to the result from [5]. The algorithm computes the maximum coverage each node would result in, then chooses the set of k nodes that would have the largest coverage.

Algorithm 2 Greedy Algorithm

1: Start with $A = \emptyset$
2: **while** $|A| \leq l$ **do**
3: For each node x, use repeated sampling to approximate $\sigma(A \cup x)$ to within $(1 \pm \varepsilon)$ with probability 1δ
4: Add the node with largest estimate for $\sigma(A \cup x)$ to A.
5: Output the set A of nodes.

2.9.2 The degree algorithm

Another popular algorithm is the degree algorithm [5]. Unlike the greedy algorithm, the degree sort all the node according to their degree distribution. The algorithm picks the top k nodes according to the degree distribution. This approacah benefits over greedy algorithm by not having as much computation time as the greedy algorithm. The disadvantage is that this algorithm does not take teh degree correlation into acount. As mentioned in section 2.3.4, High degree nodes would often have common node as neighbor. This would result in multiple overlapping activated node choosen.

Algorithm 3 Degree Algorithm

1: Start with $A = \emptyset$
2: **while** $|A| \leq l$ **do**
3: For each node x, use repeated sampling to compute DegreeMax(x).
4: Add the node with largest degree to A.
5: Output the set A of nodes.

2.9.3 Independent algorithm

Another algorithm is the independent greedy algorithm. The algorithm iterates through the network, computing the spread of each node. The algorithm then

chooses the vertex with the largest coverage independent of the other previous chosen nodes. This algorithm is a special case of the greedy algorithm mentioned above.

Algorithm 4 Independent Algorithm

1: Start with $A = \emptyset$
2: **while** $|A| \leq l$ **do**
3: For each node x, use repeated sampling to approximate $\sigma(A \cup x)$ to within $(1 \pm \varepsilon)$ with probability 1δ
4: Add the node with largest estimate for $\sigma(x)$ to A.
5: Output the set A of nodes.

2.9.4 Random algorithm

The last one is the random algorithm. The random algorithm just pick a random seed node. This approach is the simplest to implement and easiest. The downside is that this is random and there are no strategic choosing of seed node.

2.10 RMat

One problem during graph analyzation and calculation is finding suitable graphs to analyses. Generate graphs with desired properties is not easy to do. One solution proposed by Chakrabartiy et al is to use the "recursive matrix" or R-mat model. The R-mat model generates graph with only a few parameters, the generated graph will naturally have the small world properties and follows the laws of normal graphs, and have a quick generation speed [1]. The R-mat models goal is to generate graphs that matches the degree distribution, exhibits a " community " structure and have a small diameter and matches other criteria. [1]. The algorithm to generate such a recursive matrix is as follow: The idea is to partition the adjacency matrix into four equally sized part branded A,B,C,D, like shown in Figure2.8. The adjacency matrix starts by having all element set to 0. Each new edge is "dropped" onto the adjacency matrix. Which section the edge would be placed in, is chosen randomly. Each section have a probability of a, b, c, d, and $a + b + c + d = 1$. After a section is chosen, the partition that was chosen is partitioned again. This continuous until the chosen section is a 1x1 square and the edge is dropped there. From the algorithm, we can see that the R-mat generator is capable to generate graphs with total numbers of node $V = 2^x$. Since the algorithm partitioned the matrix into four part. This is approach would only generate a directed graph. To generate undirected graph, $b = c$ and the adjacency matrix must make a "copy flip" on the diagonal elements, like Figure 2.8.

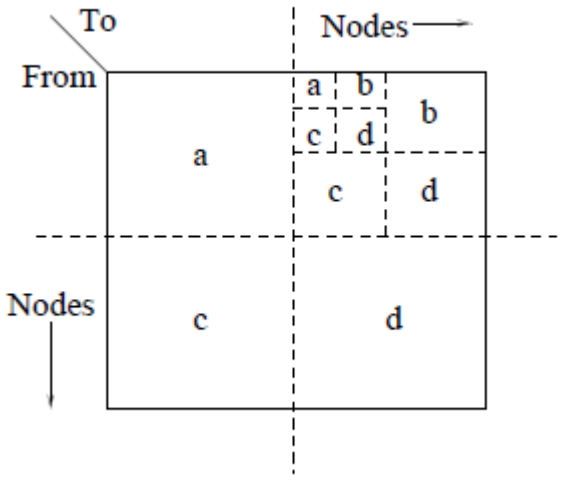


Figure 2.7: The R-mat model [1]

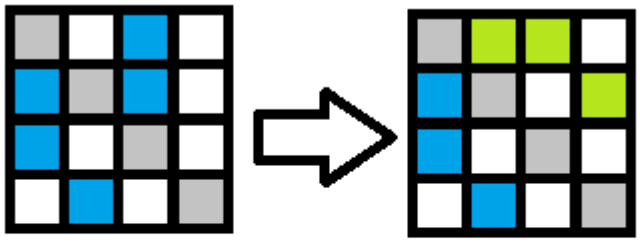


Figure 2.8: How the adjacency matrix is flipped on the diagonal

2.11 Cache oblivious model

A cache oblivious model ignores the cache size and line and designs the algorithm to be cross platform and optimized. An algorithm is *cache aware* if it contains parameter that can be tuned to optimize the cache complexity for the cache size [14]. Such algorithm have the disadvantage where a adaptation is needed for a new architecture [15]. The cache oblivious model is designed for two level of memory, and assumes that such an optimization is optimized for multiple levels as well. There are multiple algorithm designed that shows that such a model actually improves the algorithm. One of such is an cache oblivious sparse matrix multiplication. Sparse matrix multiplication is notoriously inefficient use of the cache system, it forces the user to jump through data in main memory [15]. The sparse matrix approach proposed [15] focus on reordering the matrix and row in an cache oblivious manner. There are results that shows that a cache oblivious approach have given improvement to computation time for some matrices during sparse matrix multiplication, but for cache-friendly structure, this is shown to be a minor improvement and even small loss.

Chapter 3

Related works

There are many studies studied the Maximization of influence in graph [4]. Kempe et al proposed the greedy algorithm that this report have used. There are different studies that have looked at this problem, among those are [16], where the scalability of the greedy algorithm was discussed. From [16] and our experiments, the greedy algorithm does not scale well with larger graphs. Domingos and Richardson studied how information diffusion can serve as viral marketing [13]

Some potentially improvement with hardware design include [2], that proposed the hybrid FPGA-CPU architecture. Where a FPGA-CPU hybrid BFS architecture was designed. They also showed how to view BFS as matrix-vector multiplication on Boolean semirings.

As we can see, there are many different research around this subjects, even tho there aren't many that looks at how to accelerate seed selection and the Independent cascade model in hardware.

The optimization to graph traversal is a heavily researched field. There are multiple different studies that have proposed different hardware solution for optimizing BFS [17]-[18]. Different strategies of parallise BFS have been proposed here [19].

[17] proposed a BFS for advanced multi-core processor that are scalable , enhances memory locality and cache utilization with an innovative data layout.

One optimization aspect, the high level synthesis have been getting some attention lately. [20] shows that HLS might be a viable optimization for FPGA design.

Chapter 4

Method

For this report, we wish too create some simulation of the independent cascade model and compare different seed selection algorithms to compare the result we got with previous work on similar subject [5]. We created a python simulation to simulate the data diffusion and the seed selection. We utilized the Pycx libraries to create the GUI, and applied the R-mat generator to create the different network. Our diffusion simulation finds k seed node, and for each k, runs the simulation 50 times. Each run runs until there are no more vertex to be activated. Since each node can only try to activate their neighbor once, regardless of the result, that node can't try to activate the same neighbors again. For our simulation, when no more nodes can be tested, the simulation stops.

4.1 PyCx

Pycx an libraries that help python to generate an GUI [**?**]. The pycx have a clear structure, initialize, observe and update. The initialize part, the graph is generated, the starter seed is found and the position to the graph is generated. The observe part is where python generate graphic for our simulation. for each step, the observe is called to generate a new frame. the update section i called every step, for our program, the diffusion is calculated as each step we can see how the data is diffused. The simulation allow the information and data to be displayed

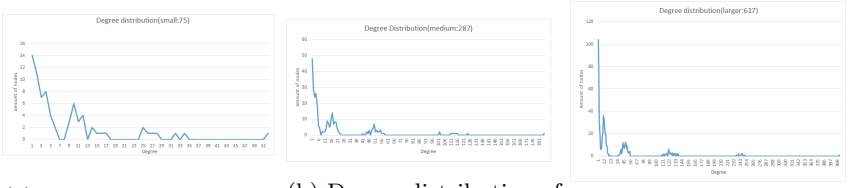
4.2 R-mat

For our generator, we choose to use the benchmark proposed at Graph500, $:A = 0.57, B = 0.19, C = 0.19, D = 1 - A - B - C = 0.05$. For a undirected graph, the matrix was mirrored diagonally. 3 graphs of different size were generated, the smaller one was with 128 nodes with approximately 400 edges, the medium sized graph was 512 node with approximately 3000 edges, and the largest was

generated with 1024 nodes and 10000 edges. The resulted graphs is proposed in next section.

4.3 Adjacency matrices to graphs

For this report, three different sized adjacency matrix was created. One of the restriction to the R-mat generator is that the size of the adjacency matrix have to be 2^n. This resulted in that our adjacency matrix was originally of the size, 128×128, 512×512 and 1024×1024. The resulted graphs had multiple singletons and unconnected nodes, for the sake of the simulation, those nodes were discarded and resulted in a graph with 75 nodes and 307 edges, 287 nodes with 2415 edges and 617 nodes and 8374 edges. This was not surprising consider that for a larger graph, those singletons would most likely result in the outskirt and small community. The degree distribution of the graphs is shown in Figure 4.1



(a) Degree distribution of small graph

(b) Degree distribution of medium graph

(c) Degree distribution of large graph

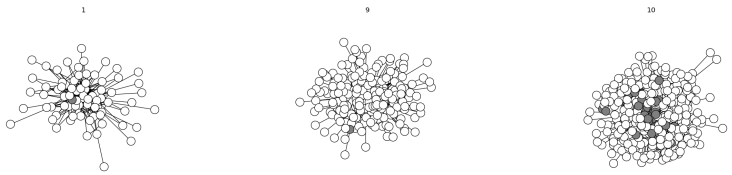
Figure 4.1: Degree distribution of the graph

4.4 The simulation

We wish to see how each algorithm scales with the increase of the graph size and compare the result from each algorithm to each other. We implemented the four different algorithm and ran the simulation on three different sized network. The simulation would first choose a set of k seed nodes as initial activated nodes. For each step, the activated nodes would propagate the activation to their neighbors. Each node can only try to activate their neighbor once. Since ICM have a global probability, to reduce the random effect of the propagation, for each new k, we would run the simulation 50 times with the same seed and find the average spread of that seed node. We created two different stop criteria for the simulation. The first simulation would stop only when there are no more nodes to activate, either since all the node that can be activated have been tested or somewhere in the network, the activation died out. The other criteria was after a specific amount of step, the simulation would stop.

For the second simulation, we choose to find the average "Step" the greedy algorithm would take before no more nodes can be activated and set the limit

to 70% of that value. For the small graph, we set the limit to 15 step, for the medium graph the limit was at 125, and the large graph was 190 step. We choose to have a global probability of 0.05 to activate the neighbors. The resulted graph can be seen in Figure 4.2.



(a) The small graph (b) The medium graph (c) The large graph

Figure 4.2: The different graphs we ran our simulation through

Chapter 5

Result

The result from the multiple runs we have gotten is as shown in the different figures. Figure 5.1 shows the spread and coverage from the simulation of the small graph. As we can see, the four different algorithm had little difference. The y-akses shows us the average coverage after 50 runs, and the x-akses gives us how many seed node was selected. We can see from figure 5.1, figure 5.2 and figure 5.3 that the resulted coverage was around 0.3-0.45. The smaller graph would result in a smaller spread, while larger graph results in large spread. Figure 5.1 have a more linear increase as the seed nodes increased. Figure 5.2 and Figure 5.3 on the other hand, had an massive increase in coverage after the addition of the second seed node, then the increase slowed down drastically.

The network that we ran our simulation on, resulted in a degree distribution as Figure 4.1a, Figure 4.1b, and Figure 4.1c. We can see that there are one vertex with high degree, while the majority of the nodes have a small degree.

We can see that the greedy algorithm performed better then the other algorithm for the large and medium graph, except the random algorithm for the large graph, where it was best.

Our result as a table:

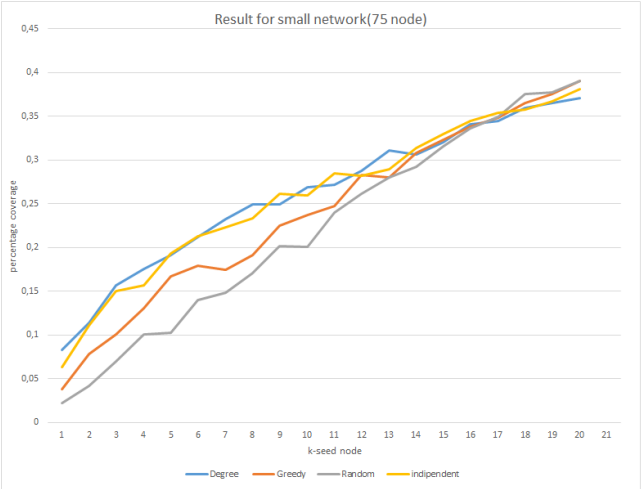


Figure 5.1: The result from the small graph

k	Degree	Greedy	Random	Indipendent
1	0.0832	0.0381333333333	0.0224	0.634666666667
2	0.114133333333	0.0778666666667	0.0413333333333	0.110933333333
3	0.157066666667	0.100533333333	0.0696	0.149866666667
4	0.175733333333	0.130666666667	0.100266666667	0.156266666667
5	0.190933333333	0.166666666667	0.102933333333	0.193066666667
6	0.212	0.178933333333	0.139466666667	0.212266666667
7	0.232533333333	0.174666666667	0.148533333333	0.2232
8	0.2488	0.1912	0.1704	0.233333333333
9	0.249066666667	0.225066666667	0.201333333333	0.261066666667
10	0.2688	0.237333333333	0.200266666667	0.2592
11	0.271466666667	0.246933333333	0.239466666667	0.285066666667
12	0.287733333333	0.2832	0.261066666667	0.281866666667
13	0.310933333333	0.280266666667	0.279733333333	0.2896
14	0.306133333333	0.308	0.292533333333	0.313866666667
15	0.320266666667	0.322666666667	0.3152	0.329066666667
16	0.3408	0.338133333333	0.336	0.3448
17	0.344533333333	0.348266666667	0.3496	0.354133333333
18	0.359733333333	0.348266666667	0.375466666667	0.3576
19	0.3648	0.375733333333	0.377333333333	0.366933333333
20	0.370933333333	0.389866666667	0.3904	0.381333333333

Table 5.1: Result from small graph

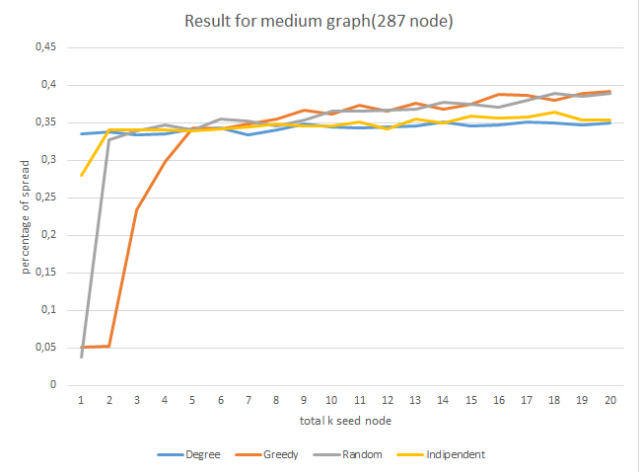


Figure 5.2: The result from the medium graph

k	Degree	Greedy	Random	Indipendent
1	0.334773519164	0.0506620209059	0.0384668989547	0.279790940767
2	0.337421602787	0.0529616724739	0.327456445993	0.340069686411
3	0.334216027875	0.23393728223	0.339442508711	0.340975609756
4	0.335400696864	0.298048780488	0.34668989547	0.341324041812
5	0.341742160279	0.342926829268	0.340557491289	0.339930313589
6	0.34362369338	0.341672473868	0.354773519164	0.341463414634
7	0.334076655052	0.348571428571	0.352055749129	0.345156794425
8	0.341324041812	0.355191637631	0.346550522648	0.348780487805
9	0.347944250871	0.366968641115	0.353867595819	0.345644599303
10	0.344390243902	0.362369337979	0.365156794425	0.345365853659
11	0.343554006969	0.373449477352	0.365783972125	0.350871080139
12	0.34425087108	0.366271777003	0.36668989547	0.341881533101
13	0.346132404181	0.375609756098	0.368432055749	0.354912891986
14	0.351358885017	0.368710801394	0.377979094077	0.349337979094
15	0.345714285714	0.374285714286	0.37456445993	0.359442508711
16	0.347804878049	0.38850174216	0.371567944251	0.355818815331
17	0.351010452962	0.386620209059	0.379721254355	0.357909407666
18	0.350383275261	0.380766550523	0.389128919861	0.364320557491
19	0.347386759582	0.388989547038	0.385714285714	0.354285714286
20	0.349268292683	0.391986062718	0.389477351916	0.353658536585

Table 5.2: Result from medium graph

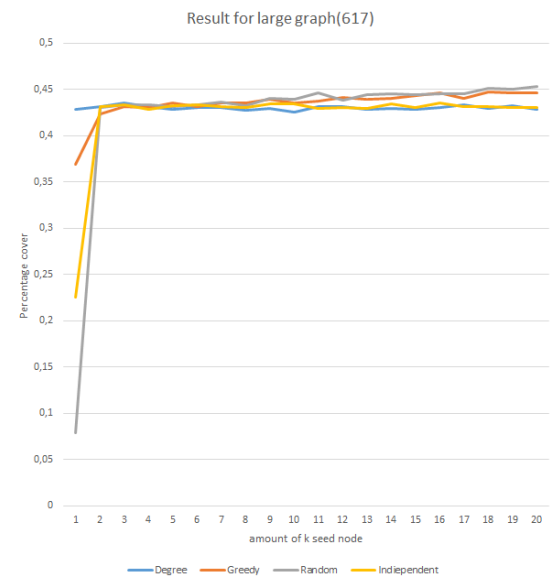


Figure 5.3: The result from the large graph

k	Degree	Greedy	Random	Indipendent
1	0.428363047002	0.368881685575	0.0788330632091	0.225705024311
2	0.431993517018	0.423857374392	0.430696920583	0.431928687196
3	0.435364667747	0.431831442464	0.433873581848	0.433387358185
4	0.431150729335	0.430470016207	0.433614262561	0.428849270665
5	0.42829821718	0.435818476499	0.431863857374	0.432544570502
6	0.430275526742	0.432025931929	0.433614262561	0.433776337115
7	0.430210696921	0.435559157212	0.436369529984	0.431377633712
8	0.427487844408	0.435170178282	0.432414910859	0.430923824959
9	0.429951377634	0.439643435981	0.440713128039	0.434424635332
10	0.425380875203	0.435170178282	0.439189627229	0.435072933549
11	0.431280388979	0.437471636953	0.446612641815	0.429692058347
12	0.431993517018	0.441847649919	0.438995137763	0.431118314425
13	0.428200972447	0.439384116694	0.444278768233	0.429627228525
14	0.429529983793	0.440842787682	0.445316045381	0.434586709887
15	0.428719611021	0.443824959481	0.444829821718	0.431118314425
16	0.431118314425	0.446126418152	0.4456726094	0.43510534846
17	0.433938411669	0.440097244733	0.445510534846	0.43209076175
18	0.42982171799	0.447066450567	0.451280388979	0.431345218801
19	0.432350081037	0.446677471637	0.450437601297	0.431021069692
20	0.428784440843	0.446839546191	0.453452188006	0.430632090762

Table 5.3: result from large graph

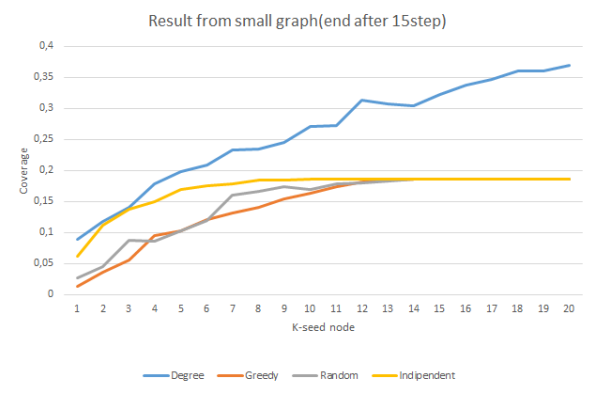


Figure 5.4: The result from the small second simulation

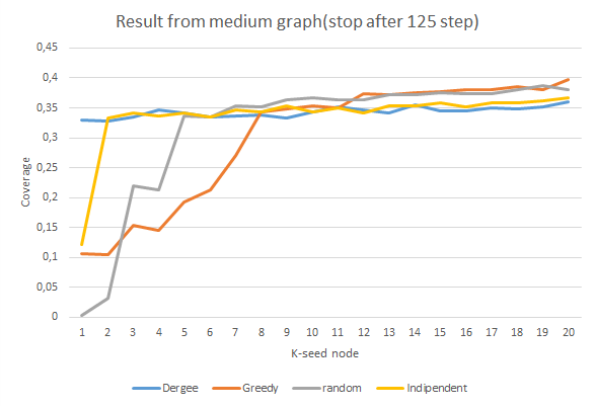


Figure 5.5: The result from the medium second simulation

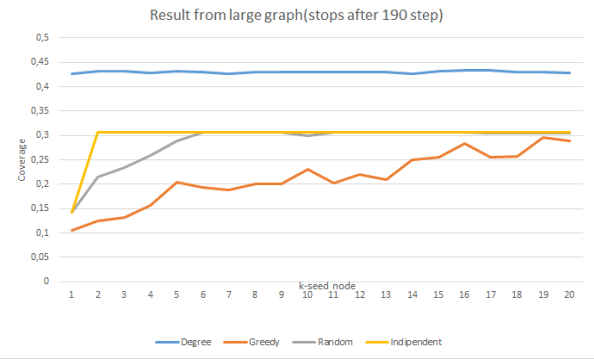


Figure 5.6: The result from the large second simulation

Chapter 6

Discussion

The result was somewhat surprising, the greedy algorithm that I thought would be the best algorithm in this simulation was not a huge improvement over the other algorithm. The random algorithm performed better then i expected and the degree algorithm was in some stages better then the greedy algorithm. The independent coverage algorithm performed better or equally good as most of the other algorithm. Toward the end where we choose twenty seed nodes, most of the algorithm performed somewhat the same.

These results can be explained by looking at our simulated social networks. The networks we ran the simulation over was a small graph compared to real world graphs, by choosing at most 20 seed nodes, a large percentage of the graph would be covered. Since the graphs was all connected in a network, the small world effect would result in that every vertex would be subjected to activation. Our global percentage was 0.05.

One paper showed how implementing a palatalized BFS algorithm on a distributed system reduced the communication times by a factor of 3.5, compared to a common vertex based approach [19]. In scientific application, ,graph based computation is pervasive and often data-intensive, this especially for distributed systems.

Our simulation runs until there are no more nodes too test. This can explain some of the result we got. Since the simulation runs until no more nodes can be activated, the coverage after 50 runs would most likely activate most of the node. This can result in no distinctive better algorithm.

For the second simulation, where we stop the simulation after 70% of the avrage step for greedy algorithm to finish, we got some interesting results. The degree algorithm performed much better then the other algorithms. With the exception of the medium graph, the degree algorithm was clearly better then all of the other algorithms. I suspect that there is some kind of problem I have yet to taken into account. If we ignore the degree algorithms result for a moment, we can see that for the small and medium graph, the greedy algorithm is performing as well as the other algorithms, in the larder graph, the greedy algorithm actually perform worse then the other.

Chapter 7

Future work

One interesting aspect that does not have huge previous work on is a hardware approach to speed up the seed selection and the information diffusion. We have seen different algorithms to select the most appropriate seed, the problem is that for larger graph, the seed selection algorithm is very taxing for the system. For this report, the graphs generated is still small, yet the largest of the used graph proved to be quite slow for the system. It would be interesting to see how an CPU-FPGA heterogeneous platform like [2] would improve the information diffusion.

Another interesting branch to explore would be to see how a paralyzed independent cascade model would measure against the non-parallel version. There are studies exploring the paralyzed BFS [19]

For this report, we created the simulation by storing arrays and "pointer chasing", as mentioned in Chapter 2, by applying sparse matrix-vector multiplication, we could have explored other potentially beneficial optimization.

The cache oblivious sparse matrix-vector multiplication is one aspect we have yet to touch upon. It would be interesting to further investigate how much of an improvement this would potentially give compare to a standard sparse matrix multiplication.

One solution that i would like to explore, is the FPGA-CPU heterogeneous platform proposed in [2]. There we saw how to perform BFS on Boolean semirings. One modification to the algorithm, is after each matrix-vector multiplication, we compute each new activate cells in y_n have a percentage to be deactivated. After deactivation, depending on how we implement the next step, if we only calculate newly activated cells, then each node would just activate their neighbor once, the other solution is to remove the edge between the old node and the new node. This preventing the node from activating a node the second time.

Another interesting aspect that would be interesting to explore is the high level synthesis. One such HLS is the Vivado high-level synthesis, where normal C code can be directly targeted into Xilinx all programmable devices. We can see from [21], that the HLS can achieve 11-31% FPGA recourse usage reduction.

Chapter 8

Conclusion

As we have shown in this report, there are multiple different algorithms to select the seed. Information diffusion can simulate different aspect of real life. The problem is very taxing on a normal home computer. BFS can be optimized, ICM is a special case of the BFS. There are different optimization, matrix multiplication, maybe something with hardware.

The result we got from the simulation was somewhat surprising, the result from a similar simulation [4]. From [4] we see that the greedy algorithm is better then the other algorithm and the random algorithm performed the worst. For the simulation where the simulation runs until there are no more nodes to activate, the greedy algorithm performed the same as the other. This can be resulted by the stop criteria.

For the future work, it would be interesting to see if the matrix-vector multiplication can improve the seed selection algorithm or the information diffusion. One interesting aspect is to see if there are possible to improve it via some form of hardware solution.

Proposed solution, the modified matrix vector multiplication.

have a hardware design for it.

can compare with the cache oblivious sparce matrix multiplication.

Bibliography

[1] Deepayan Chakrabarti, Yiping Zhan, and Christos Faloutsos. R-mat: A recursive model for graph mining. 4:442–446, 2004.

[2] Y. Umuroglu, D. Morrison, and M. Jahre. Hybrid breadth-first search on a single-chip fpga-cpu heterogeneous platform. pages 1–8, Sept 2015.

[3] Stephen Eubank, Hasan Guclu, VS Anil Kumar, Madhav V Marathe, Aravind Srinivasan, Zoltan Toroczkai, and Nan Wang. Modelling disease outbreaks in realistic urban social networks. *Nature*, 429(6988):180–184, 2004.

[4] Éva Tardos David Kampe, Jon Klein. Maximizing the spread of influence through a social network. pages 137–146, 2003.

[5] Éva Tardos David Kampe, Jon Klein. Maximizing the spread of influence through a social network. 2003.

[6] Jeremy Kepner and John Gilbert. *Graph algorithms in the language of linear algebra*, volume 22. SIAM, 2011.

[7] M. E. J. Newman. *The Structure and Function of Complex Networks*, volume 45. 2003.

[8] Stanley Milgram Jeffrey Travers. An experimental study of the small world problem. *Sociometry*, 32(4):425–443, 1969.

[9] Stanley Milgram. The small world problem. *Psychology today*, 2(1):60–67, 1967.

[10] Jeffrey Travers and Stanley Milgram. An experimental study of the small world problem. *Sociometry*, pages 425–443, 1969.

[11] James Moody. Race, school integration, and friendship segregation in america1. *American journal of Sociology*, 107(3):679–716, 2001.

[12] MH McAndrew. On the product of directed graphs. *Proceedings of the American Mathematical Society*, 14(4):600–606, 1963.

[13] Pedro Domingos and Matt Richardson. Mining the network value of customers. pages 57–66, 2001.

[14] Matteo Frigo, Charles E Leiserson, Harald Prokop, and Sridhar Ramachandran. Cache-oblivious algorithms. pages 285–297, 1999.

[15] AN Yzelman and Rob H Bisseling. Cache-oblivious sparse matrix-vector multiplication by using sparse matrix partitioning methods. *SIAM Journal on Scientific Computing*, 31(4):3128–3154, 2009.

[16] Behrouz Derakhshan et al. Influence maximization in large scale graphs. 2015.

[17] Virat Agarwal, Fabrizio Petrini, Davide Pasetto, and David A. Bader. Scalable graph exploration on multicore processors. In *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, Washington, DC, USA, 2010. IEEE Computer Society.

[18] Scott Beamer, Krste Asanović, and David Patterson. Direction-optimizing breadth-first search. *Scientific Programming*, 21(3-4):137–148, 2013.

[19] Aydin Buluç and Kamesh Madduri. Parallel breadth-first search on distributed memory systems. page 65, 2011.

[20] Gordon Inggs, Shane Fleming, DavidB. Thomas, and Wayne Luk. Is high level synthesis ready for business? an option pricing case study. In Christian De Schryver, editor, *FPGA Based Accelerators for Financial Applications*, pages 97–115. Springer International Publishing, 2015.

[21] Jason Cong, Bin Liu, Stephen Neuendorffer, Juanjo Noguera, Kees Vissers, and Zhiru Zhang. High-level synthesis for fpgas: From prototyping to deployment. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 30(4):473–491, 2011.