NTNU
Norwegian University of
Science and Technology

TDT4506 Specialisation project

PLACEHOlder Title

Julian Lam

Fall 2016

Department of Computer and Information Science
Norwegian University of Science and Technology

Supervisor 1: Donn Alexander Morrison
Supervisor 2: Yaman Umuroglu

0.1 Assignment

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 modelling 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.

High-level synthesis (HLS) is becoming an important tool in the optimization/acceleration of algorithms in hardware. Starting with an algorithm written in a high-level language such as C or C++, HLS aids with hardware design by providing a methodology and tools that guide the developer through the design process.

This project should employ HLS as a design methodology for hardware accelerated seed selection in large graphs. The student will study seed selection for a given diffusion model, write a high-level model, and use HLS to implement a hardware design that exploits parallelism in the seed selection algorithm in order to improve performance over a GPCPU implementation. –

Abstract

PLACEHOLDER ABSRACT

This report is about how High level synthesis can be used to implement custom IP-core to solve specific problem, in this case, The seed selection of data diffusion. By using HLS we were able to test out different optimization schemes and compare the results of them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Information diffusion is a field of network research where a message, or data, is propagated through a network through edges. The message originates from a chosen set of nodes, known as *seed nodes* and propagates through the network according to different models,*Independent Cascade Model and Linear Threshold Model. High level synthesis take algorithms implemented in high level programming language such as C and C++ and and synthesise the design to low level designs[**?**].*

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. We will also have a look at High Level synthesis. This chapter will contain notations that we will 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 [**?**]. We will look at the independent cascade model, which is a special case of breadth first search [**?**]. By looking at how to improve BFS, we can apply such optimization to ICM and the seed selection algorithm. Another major part of the report is to utilize High Level Synthesis to synthesise the 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-

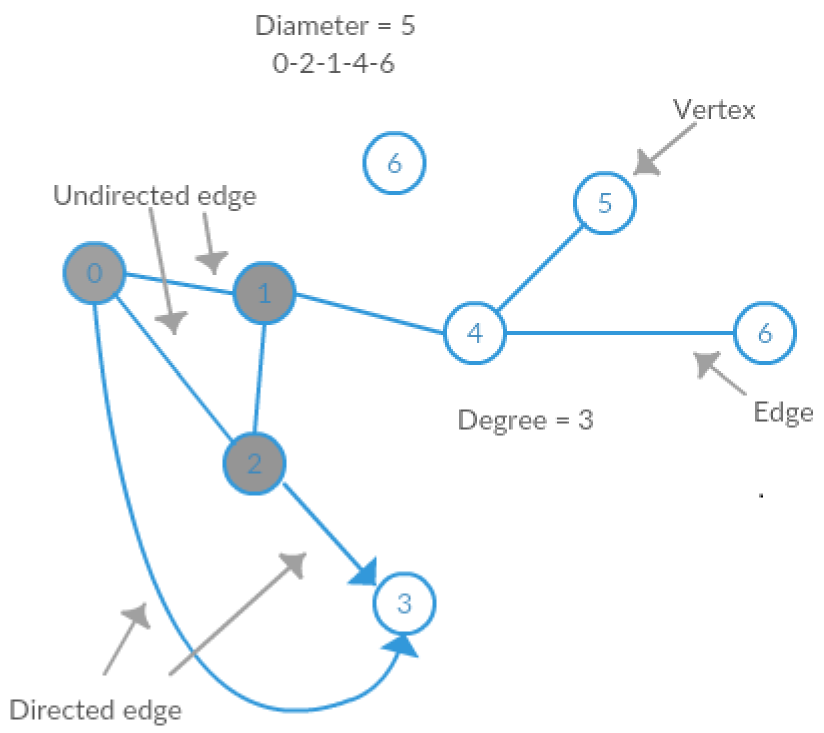


Figure 2.1: Simple network

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. 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 geodesic path 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 describe 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 [**?**]. 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 social network [**?**], information* *networks [**?**], technological networks [**?**] ,and biological network [**?**]. Different network have different properties and in this report 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 [**?**]. 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 [**?**]. Some notable experiments are the letter passing experiment [**?**], which we will discuss more in 2.3.1*

2.3 Network properties

Network properties are different characteristic properties that networks displays. We will focus on those that are more relevant to social networks 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 [**?**]. The experiment involved passing letters from person to person to reach a designated target with only small steps. For the published case, the chain was around six [**?**], meaning there were only six passes necessary for the letter to reach its destination. This shows us that for most pairs 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." [**?**]. 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 paper [**?**] asked even though 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.

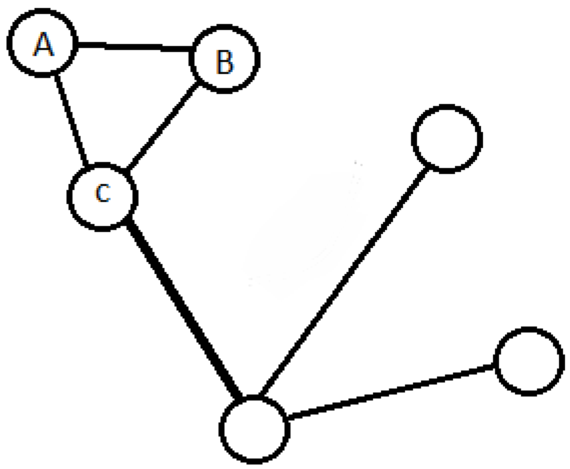


Figure 2.2: Examples of triangles in networks

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 are friends with 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 and those node would *likely be high prioritized node. 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 mentioned in section 2.3.3, 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 wether high degree nodes tends to connect to other high degree nodes, or if they prefer to connect to low degree nodes. It turns out that both incidents ar 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 a majority of high degree nodes 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

For 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 schemes, 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, in which 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 regarding 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 it would result in remove influential nodes and limit the spread of the data. Removing important nodes connectiong two different communities would result in isolation and no path towards other communities. 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 [**?**]. 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 communities would be smaller. This is of course dependent on how many intercommunity connections there are in the 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 continues until the queue is empty and all the nodes have been iterated over.

Breadth first search is a common graph iteration algorithm. The breadth first search is often limited by the irregular memory access where the algorithm have to find the data stored in different spaces in the memory. As mentioned earlier, independent cascade model is one special case of the Breadth first search. Breadth first search is the independent cascade model with a 100% chanche to activate the neighbores.

Algorithm 1 Breadth First Search

```
1:  $dist[\forall v \in V] = -1; currentQ, nextQ = \emptyset$
2:  $step = 0; dist[root] = step$
3:  ENQUEUE(nextQ,root)
4:  while $nextQ \neq \emptyset$ do
5:      $currentQ = newxtQ; nextQ = \emptyset$
6:      $step = step + 1$
7:      while $currentQ \neq \emptyset$ do
8:          $u$ = DEQUEUE(currentQ)
9:          for $v \in Adj[u]$ 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. Some 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. Seed nodes is a set of k nodes that in the initial time-step are 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

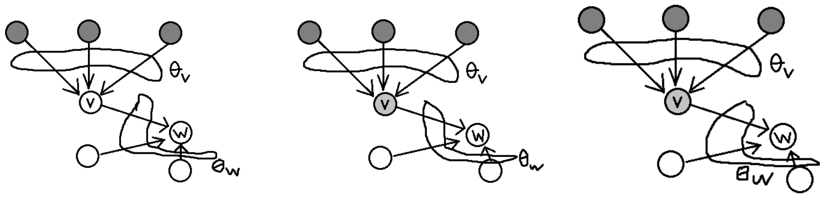
*When we talk about data diffusion, we can look at how diseases or technological innovations would spread through a social network. We can simulate those kind of behaviors with different diffusion models. There are two basic diffusion models used to simulate the propagation of information through a network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

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 friends 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 larger circle of friends then others(high degree), while others have larger impact on a person(large p_x). Some might be harder to promote too(weighted edges), while some users have no friends(singletons). The different models we will focus on, are 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, let's assume that the weight of all the nodes in this model are 0.5, meaning over half of its neighbors 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.

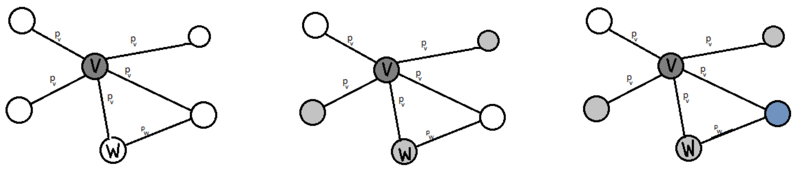


(a) Step 1 (b) step 2 (c) step 3

Figure 2.3: Linear Threshold mode

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 probability $p_v != p_w$. In 2.4a, the node v is activated, during the next time-step, node v have activated three neighbors. 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.

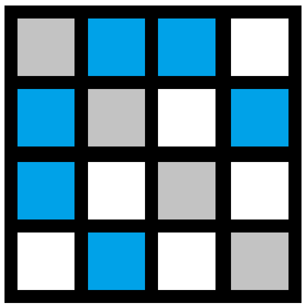


(a) step 1 (b) step 2 (c) step 3

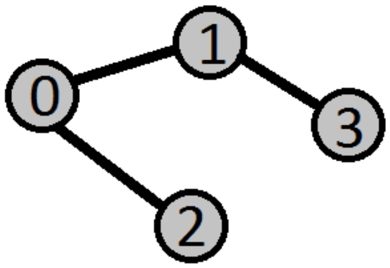
Figure 2.4: Independent cascade model

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 continue 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 child 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



(a) The adjacency matrix



(b) The graph corresponding to the adjacency matrix

Figure 2.5: Sparse matrix to graph

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.

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 matrices [**?**]. We can see that such an idea have been proposed in earlier literature [**?**]. By representing graphs as a 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 $N \times N$ 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 matricies only have few non-zero elements, by storing only the non-zero elements, we can have savings in memory.

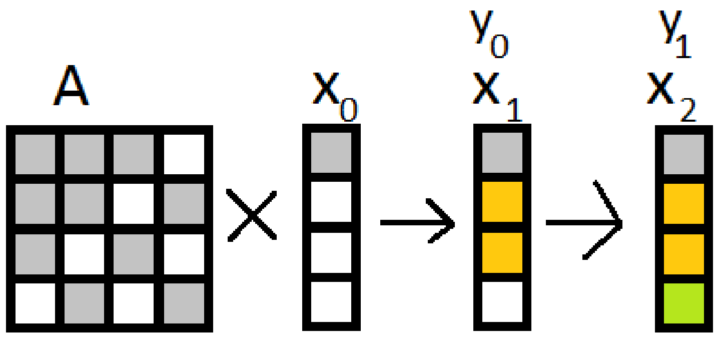


Figure 2.6: BFS on Boolean semiring

2.8.2 Breadth First Search as a matrix multiplication.

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

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. Each selected nodes is in the initial timestep activated. During each timestep, the seed nodes will propagate the activation along the network depend on what diffusion model is used. 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 viral marketing [**?**]. 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 [**?**] proposed by Kempe et al, is known to be the best algorithm according to the result from [**?**]. The greedy algotihm starts by itterate over the entire network and finds one node that have the largest coverage and stores that node in the set A. The algorith then activates that node from A, and computes the coverage of each node in the network with the previous choosen node. After the algorithm finds the second node, that node is stored in A with the previous node and we have now choosen two seed node. The algorithm continues until we have choosen k seed nodes, where each have been tested to have maximum coverage in relation with the previous choosen nodes. To compute the maximum coverage, the algorithm have to test every combination of nodes together, this results in heavy computation and the algorithm would therefore not scale well.*

We can look at the greedy algorithm as a special case of BFS with a transistion probability. Each node in the network will be choosen as the seed node for the

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 $1\delta$
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 [**?**]. Unlike the greedy algorithm, does not compute the coverage of node, the algorithm picks the top k* *nodes according to the degree distribution instead. The node chooses the top k nodes with the highest degree and stores them as the seed nodes. This approach benefits over the greedy algorithm by not having as much computation time as the greedy algorithm since only one itteration is needed to compute the degree to node. The disadvantage is that this algorithm does not take the 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 $1\delta$
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 picks 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

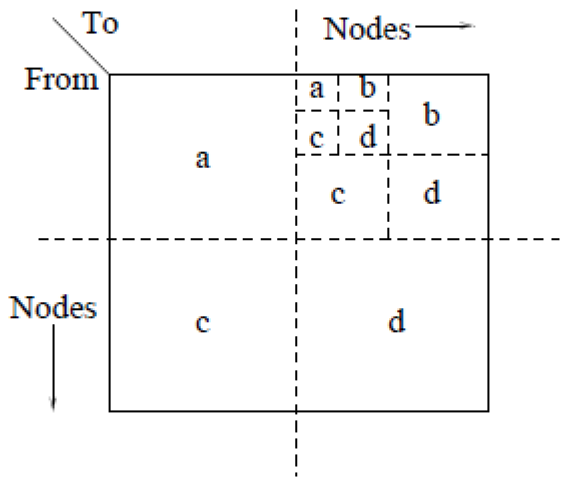


Figure 2.7: The R-mat model [**?**]

*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 [**?**]. 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. [**?**]. The algorithm to generate such a recursive matrix is as follows: The idea is to partition the adjacency matrix into four equally sized part branded A,B,C,D, as 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 continues 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 are 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.*

what is Information diffusion

what is the solution(bfs)

what is ICM

how to implement bfs as SpMV

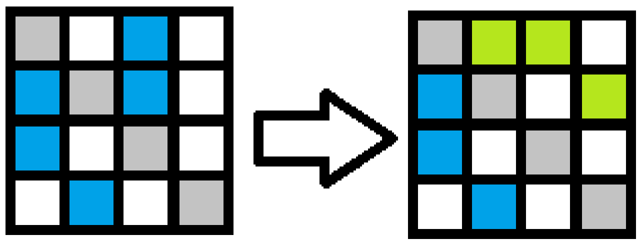


Figure 2.8: How the adjacency matrix is flipped on the diagonal

how to apply information diffusion to bfs and problem

2.11 High Level Synthesis

High Level Synthesis is a synthesiser that converts high level programing languages such as C, C++, SystemC. There are multiple different library and stuff

2.12 Memory Mapped axi interface

Something something mapped to memory.

2.13 Cache coherency

Cache coherency, which is where multiple cache reference the same area.

2.14

$$ \tag{2.1} $$

Chapter 3

Related works

Yamans paper, where there are some works that shows the solution i use

parallalization of the algorithm

maybe some examples of HLS to show that HLS is used.

showe that there are not many HLS implementation, recently matured.

show that there are not many hardware implementation for information diffusion.

need to look through Yamans paper and get some refrences from there.

might be good to look at how this type of sparce matrix multiplication can be used

show other implementation of SPmv

Show some examples where image processing is done through vivado HLS.

Chapter 4

Method

In this chapter, we will discuss how we designed the high level system and how the different core is connected. We will in later chapter discuss the choice of said designs. For this project, we have implemented a simple Linear-feedback Shift Register(LFSR) and our custom IP core which for this report will be called Data Diffusion Core(DD Core).

The LFSR is a commonly used random number generator(RNG)[CITATION NEEDED]. Different sized LFRS is able to generate pseudo random numbers dependent on how large the LFSR is. LFSR generates a pseudorandom number based on the previous number. The LFSR is a register containing 16 bit. we take the bit from, 16,14,13 and 11, resulting in (((16 XOR 14) XOR 13) XOR 11) = bit. The bit is pushed into bit-position 1 and the entire raegisters shifts towards left 1 bit.

We ran multiple test to prove that the LFSR is determenistic. As we can see from the table below:

[INSERT TABLE; 2 COL: both send in the same seed and we can see that we receive same number,](Can look at the master thesis Donn send me ages ago.)

The IP core we created, Takes in different signals, The address to the location to the matrix, the location to the result, and the x-buffer. Since we are working with ICM, a global probability, a random seed as the initial state for the LFSR. The core applies matrix vecotr mulitplication. For this implementation, a vector, vector multiplication is applied, to be able to handle graph that is larger then recomended cache size. The cache have not enough space to store the entire adjacency matrix.

We started by creating the pseudocode to our implementation. The code was a modified form of the shown code of sparse matrix vector multiplication over a boolean semiring.

For this project, we created a sparse matrix with a R-mat generator. The R-mat generator is an simple python program that creates a sparse matrix with total elements equal = power(2,n). The amount of edges that was created is $(total_amount_of_edge \times edgefactor).Thematrixisstoredasasingletextfilethatcontainsastringcontaining1sor0s.$

$matgeneratorpartitionsthematrixintodifferentpartswheretheelementisplaced.Eachparthaveaspecificperce$
$0.57, B = 0.19, C = 0.19, D = 0.05.Foraundirectedadjacencymatrix, Thematrixmustbediagonallyidentical.[F$

For our core, we used the architecture of memory mapped. We pass the address in the form of AXI4-lite. The core includes an dma-function, which the HLS have included for us.

We implemented a vector-vector multiplication. The multiplications applies the logical action AND with each element in the vector with the corresponding element in the row of the matrix.

- Start by creating pseudocode to illustrate how the Independent cascade model works and how to set up the code. - Implement the algorithm(ICM) in C code, since High level synthesis synthesis C/C++ code. - Create test to make sure the code is correct. (Test early to catch problem early.) - Port the code over to be high level synthesis compatiable. () - analyse the implementation. - implement on Hardware. - might want to implement wiwth interface, AXI 4 and stuff like that

NOTES: For our example, a AXI4 Stream would be best i think.

implemented a 16-bit LFSR(linear feedback shift register) to generate the random number, which is used to calculate if the activation will take place. each have a [5-25%] activation rate. implemented some different models, use a array instead of a matrix.

WE have created a generator following the R-mat generator for sparse graph for testing. We implemented the simulation as a sparse matrix vector multiplication we implemented that in vivado HLS. and synthesised it We changed the implementation to accomodate the requirements for HLS; - no recursion, unknown matrix length and allocation of memory.

pseudocode section: $matrix_vector_multiplication(matrix, x\_vector, result\_vector, coin\_toss, total_node):$

for(row-¿$total_node)do : for(col- > total_node)do : if(x_vector[col]matrix[row][col])do :$
$if(row = col)do : local\_result = 1; elsedo : local\_result = coin\_tosslocal_result; result[row] =$
$local\_result; local\_result = 0;$ ....

4.1 Parallization

The function matrix_vector_multiplication() performes a single matrix vector multiplication. From the pseudocode, we can see that there is a room for parallization of the SPmV. The outer for loop from the pseudocode, can be parlized since thatfoor loop is not dependent on the variable from the inner for-loop.

another paralization is during the simulation, after the spmv, the frontier needs to be calculated. And a converged() function is called in the end to determen if the simulation is finished. The frontier calculation and converged can pe run in parallel.

THe IP-core (currently) is only using around 2% of the resource available on the FPGA(Zedboard). This gives us a lot of room for parallization of different core. Implemented 2 bus, one for input stream, one for output stream. The output stream consist of the result$_vector$

THe input vector is the seed$_vectorandthematrix.TheseedandtheglobalprobabilityisusedaAXI4litesincethe$

IP Core Structure: For the second option, where the RNG core is not implemented into the IP CORE, we would have to have teh random number set as AXI4 stream from the buffer, since we would need to call a stream of random number from the core.

(global vs local probability) For this experiment, we have set a standard global probability, that signef that each activation of nodes is at a specific probability. THe local probability is that there are a loccal probability to each edge. Each edge can have a specific probability.

4.2 High Level Synthesis

HLS have many predefined protocol, axi4lite, srteam etc etc. These can be implemented in vivado HLS easily. The program needs to be predefined so that it knows when the streaming is done. The axi stream is best suited for stream of data. The work flow to the HLS is to first generate the C or C++ code. Then the create a testbench. then run the co-simulation. The co-simulation runs the test on both on the C-code and the simulated core. After that, run the core to the vivado to implement and design the core.

Chapter 5

Result

as we can see, the algorithm was able to finish a

Chapter 6

Discussion

6.1 problem that was encountered

One problemt that I encoutered during this project was that the output signal from the synthesiser was not in the correct direction. The output signal was often set as input signal. The HLS would automaticlly set the values as output signal or input signal.The reurn value from a funciton would be set as a the output signal, while the variable that the function takes, would be set as the input signal. Another way wo specify that something is the output signal would be to explicitly set them as pointer arguments. This will in set the signal to be output signal.

6.2 AXI4

To use a standard protocol to transport data, Xilinx have created what is known as AXI4, there are different standards to implement, axi4 lite, stream, master, etc: each have a different usage and standards. The axi4 lite is more suitable for smaller and easier IP-core[need to explain what IP core is]

2.0000e+09 2000 500 222.2222 125 80 55.5556 40.8163 31.2500 24.6914

2000000000.00000, 2000, 500, 22.222222222222, 125, 80, 55.5555555556,40.8163265306122,31.2500000000 24.6913580246914

6.3 implementaion

The vivado implementation on the Zedboard was problemtatic.

Chapter 7

Future work

something something

Interesting future work would be explore how the different architecture would improve the performance. For this implementation, we used a core that contained the LFSR, an interesting design would be to have one shared RNG for multiple cores that shares the RNG. The LFSSR would contionously generate random numbers, and each core reads from the random number buffer. This design would free up space and might be able to implement multiple cores in the Zedboard. The only problem would be that for each read from the buffer would result in high overhead. The size and rate that the RNG can be genarated will also be a limiting factor.

for this design, there are multiple optimization scheme that this report have not explored. We have not yet designed a more general scheme for computing.

For this project, a tiny graph was used, refering to graph 500 benchmarks, the smallest used graph is a toygraph, which is on the scale at around $2^2 6$

The community of the HLS is very active and frequently responds to forum post seeking help. Not many work that uses HLS[CITATION NEEDED]. Recently was free, used to cost money.

Learn more about HLS so it can be better utilized. different scheme to further work: - implement parallel - different scheeme, rng on the outside - Use other data structure. - larger graph - compare this solution to other solution, - implement more efficient memory storage - use other storage method since its sparse. - would be interesting to gather information on energy consumption. - Implement a more general architecture to handle more total nodes.

Chapter 8

Conclusion