Universal Identifier: Who Are You?

Team Target

Niles Armstrong, An Do, Arezou Ghesmati,
Alex Happ, George Lankford, and Ding Zhao

Proposal for Universally Identifying Users using

Math to Industry Boot Camp (IMA), July 2016

1 Introduction

Effective business strategies always involve understanding their customers. A successful business should be cognizant of how their customers use the resources provided for them. While information can be gathered on a more general level, the variability of the demand of individuals can be very high. This is particularly important when trying to give shoppers that use Target's website a satisfying visit. Thus, by uniquely identifying customers, a more personalized shopping experience can be provided to the customer. The technological age has made the problem of identifying a distinct customer feasible with the amount of data being recorded and stored. Simultaneously, this problem is complex due to people interacting with businesses through multiple channels (store, website, coupon sites, etc.) and through multiple devices (desktop, mobile, phone). This work provides an algorithm that will provide a constrained solution to this problem. In Section 2, the assumptions and methodology for generating data to test the algorithm is provided. Section 3 gives the algorithm that will be used to uniquely identify customers from the data. Next, Section 4 provides data and the results from using the algorithm. Section 5 discusses the results and provide the pitfalls that will need to be addressed in future research. Finally, Section 6 gives concluding remarks.

2 Data Generation

The first step to identifying customers is understanding the data that is available about them. In order to keep the anonymity of customer information, synthetic data needed to be created. The first step is to decide what preliminary assumptions need to be made about the form and the content of the data.

The data that is generated comes from records of visits to Target's website. Information gained from store visits are not considered for this work. Customer website visits are described as one of four scenarios. A customer will come to the website via a device (desktop, laptop, cellphone or tablet) and

1. look at products with no purchase.
2. purchase an item with a credit card as a guest.
3. logs in as a member but makes no purchase.
4. logs in as a member but makes a purchase with a credit card.

The identifiers and their abbreviations that are considered from these scenarios are given in Table 2.1.

Identifier	Abbreviation
Browser Cookie	C
Login Username	Log
Email Address	Em
Credit Card Number (hashed)	CC

Table 2.1: The identifiers gained from website visits.

An email address being observed from the scenarios is considered a guest email address or combined with its username. The reasoning is that every username is associated with an email so they are considered the same. Table 2.2 shows the information that is gained from each scenario.

Scenario	C	Log	Em	CC
1	x	0	0	0
2	x	0	x	x
3	x	x	0	0
4	x	x	0	x

Table 2.2: The "x" and "0" means this information is or is not gained, respectively.

Many devices and web browsers (Safari, Google Chrome, Firefox, Internet Explorer, etc.) are used to visit the website. Thus, customers can be associated with a multitude of different cookies. For example, an individual could be linked to 2 different cookies where one is from their mobile phone through Safari and the other is from their desktop using the same browser. Another situation is a cookie's lifetime depends on how often a user clears their cookies. If cookies are cleared often then many cookies can be connected to one person.

Consider the following simple example to illustrate a graph of the data. A person goes to the website using their laptop, logs in, and purchases items. This gives a C, Log, and CC. A few days later, they come back to the site and purchase another product using their desktop as a guest with a different email than their Log, but uses the same credit card as the first time. This time a C, Em, and CC are gained. A few weeks later the person logs in to the website using their laptop just to browse with no purchase. This visit provides a C and Log. This example is represented by the graph described in Figure 2.1.

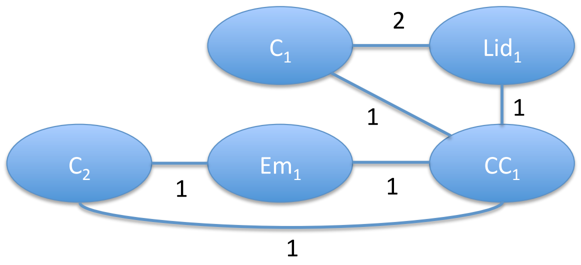

Figure 2.1: It is observed that the there are two different cookies to represent the visits from the laptop (C_1) and desktop (C_2). The two above the edge between C_1 and Log_1 (C_1Log_1) accounts for logging in from the desktop twice.

A node represents an identifier gathered when the website is visited. An edge indicates that the two nodes were gathered during one visit. The number on the edges is the frequency a link is formed between the 2 nodes. The graphs generated are k-partite graphs. Thus, no two identifiers of the same kind can have an edge between them. The graphs are generated by making random weighted adjacency matrices.

An adjacency matrix is a matrix where the rows and columns are graph vertices. In this case, they will be the identifiers. The weight of the edge connecting node i and node j is in the i, j entry of the weighted adjacency matrix (wam). The wam for Figure 2.1 is given in Matrix (2.1).

$$\begin{array}{c} \\ C_1 \\ C_2 \\ Log_1 \\ Em_1 \\ CC_1 \end{array} \begin{array}{c} \begin{array}{ccccc} C_1 & C_2 & Log_1 & Em_1 & CC_1 \end{array} \\ \begin{pmatrix} 0 & 0 & 2 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 2 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix} \end{array} \tag{2.1}$$

Wams can be made random or specific for each user. To test the data, wams are made like in Matrix (2.2).

$$\begin{bmatrix} \text{User 1} & * & * & \cdots \\ * & \text{User 2} & * & \cdots \\ * & * & \ddots & \cdots \end{bmatrix} \tag{2.2}$$

User i is generated and entries in $*$ represent spurious edges that arise from users interconnecting. This allows the true answer to be known while being able to devise random connections between users. The graphs given by these wams are used in the algorithm developed in the next section.

3 Methodology

Since the graphs being used are undirected weighted graphs, a minimum cut algorithm could be applied immediately. A simple minimum cut algorithm can partition a graph into disjoint sets based on edge weights. However, this would give rise to situations where isolated nodes, nodes with no edges, are created such as in Figure 3.1.

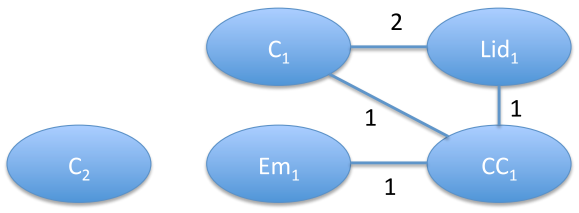

Figure 3.1: C$_2$ is an isolated node possibly created from a minimum cut algorithm implemented on the graph from Figure 2.1. Now, there is no way to use the information in C_2 when it could contain valuable details about a user. C$_2$ is called a *dangling cookie* and is not very useful in this context because it gives no link to a particular user.

Thus, the algorithm is an iterative method where each iteration is broken into three parts: Feasibility, Merge, and Cut. This section will give an overview of how each part works. The Feasibility part of the algorithm will decide if the other two parts are needed.

3.1 Feasibility

The Feasibility section is used to decide if there are more than one user in a graph. This will give understanding for if the graph needs to be cut. The principal components that determine this are the number of Ems, Logs, and CCs in the graph. After extensive research, the averages in Table 3.1 and Table 3.2 give percentages for the number of emails and credit cards per person used in this work, respectively.

Identifier	0	1	2	3	4	5+
Ems	0%	30.85%	38.29%	24.17%	6.06%	.63%

Table 3.1: The average number of Ems per person.

Identifier	0	1-2	3-4	5-6	7+
CCs	18%	48%	18%	9%	7%

Table 3.2: The average number of CCss per person.

The weak composition of the number of users and the average statistics are used to come up with a probability for if there are one or more users in a graph. Table 3.3 and Table 3.4 describe the different compositions for 2 users given there are 2 Ems/Logs and 2 CCs, respectively. Here, $\{a, b\}$ in the composition column describes the situation if one user has a Ems/Logs/CCs and the second user has b Ems/Logs/CCs.

Composition	Probability of Composition
{2,0}	$0\% * 30.85\% = 0\%$
{0,2}	$0\% * 30.85\% = 0\%$
{1,1}	$30.85\% * 30.85\% = 9.52\%$

Table 3.3: The weak composition for 2 users if there are 2 EMs/Logs in the graph.

Composition	Probability of Composition
{2,0}	$48\% * 18\% = 8.64\%$
{0,2}	$48\% * 18\% = 8.64\%$
{1,1}	$48\% * 48\% = 23.04\%$

Table 3.4: The weak composition for 2 users if there are 2 CCs in the graph.

The feasibility probability (fp) is calculated by multiplying each row by each column and adding up the percentages. Equation (3.1) gives the fp from Table 3.3 and Table 3.4.

$$0(.0864 + .0864 + .2304) + 0(.0864 + .0864 + .2304) + .0952(.0864 + .0864 + .2304) = .0384 = 3.84\% \quad (3.1)$$

Thus, fp(2 users) with 2 Ems and 2 CCs is 3.84%. Here, fp(i) where $i \in \{1, 2, ..., k\}$ is computed where k is either the total number of Ems/Logs in the graph or the first number of users with a higher probability than one users. If f(1) $>$ f(i) for $i \neq 1$ then the graph has one user. Otherwise, the graph needs to be cut and the algorithm moves into the Merge section. Algorithm 3.1 describes the Feasibility function (feasfunc) used in this work.

Algorithm 3.1 feasfunc(Graph G)

```
1. Let $totEm$ = total # of Ems/Logs and $totCC$ = total # of CCs.
2. Let $i = 2$.
3. Let $oneuser$ = fp(1)
4. While $i \neq totEm + 1$
    (a) Compute fp(i users) for totEms and 2 CCs.
    (b) If $fp(i) > oneuser$ go to Merge.
    (c) Otherwise $i = i + 1$.
```

3.2 Merge

If the feasfunc determines the graph should be cut, nodes are merged together to avoid losing information as in Figure 3.1. Now it must be decided which nodes will absorb other nodes. To this end, nodes are separated into two types: core and data nodes. The core nodes are the principal identifiers of a user, namely the Ems and Logs. The CCs and Cs are data nodes because they contain information about the user. It is important to note that the core nodes *absorb* the data nodes during the Merge. This ensures no information about the graph is lost. Figure 3.2 gives an example of how the merge is completed. The Merge order is determined from the initial graph. The data nodes will be absorbed into the core node with the highest weight connected to it. After the Merge, then a Stoer-Wagner minimum cut is made.

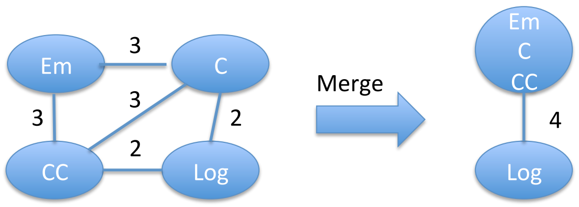

Figure 3.2: Note that the merged nodes contain the information that it absorbed including edge weights. If the edge is not absorbed then the remaining edge weights are summed. In this case edge CCC and CCLog are summed to make the final edge weight of 4.

3.3 Minimum Cut algorithm

A minimum cut algorithm finds a partition of the graph into disjoint subgraphs such that the sum of the weights of the edges connecting the two parts is minimum. The Merge allows this to be an easier task. The Stoer-Wagner algorithm is already a built-in function in the software being used so it was implemented. The reader is referred to [1] for details. After the cut is complete, the graph should be in two subgraphs. However, this does not guarantee that both subgraphs are distinct users.

3.4 Algorithm

Stoer-Wagner splits the graph once, so a subgraph could potentially have multiple users. This is why the algorithm is an iterative method. Each subgraph after a Stoer-Wagner cut is put throught the algorithm until all subgraphs are unique users. This is why it is important not to lose any information from the graph. Algorithm 3.2 gives a description of how the method is implemented.

Algorithm 3.2 unid(Graph G)

```
1. Let Uid be empty.
2. Call feasfunc($G$)
3. If feasfunc says there is one user, done. If not, Merge($G$).
4. Then MinCut(G). Let $S = \{S_1, S_2\}$ where $S_i$ are the subgraphs produced my MinCut($G$).
5. Let $i = 1$.
6. While $S$ is not empty
    (a) feasfunc($S_i$)
    (b) If $S_i$ is not a distinct user, then Merge($S_i$) and MinCut($S_i$). Add the two new subgraphs to the
        end of $S$ and $i = i + 1$.
    (c) Otherwise $S_i$ is deleted from $S$ and added to Uid. $i = i + 1$.
```

3.5 Example

To illustrate Algorithm 3.2, an example is presented. User 1 logs in on their desktop 10 times and purchases with their Mastercard five times. User 2 has a laptop that they have logged in on 7 times and purchased with their Visa four times. Sometimes the User 2 forgets their login information so they have purchased as a guest a couple times. There was once where User 1 logged into User 2's laptop and purchased something with their American Express. The wam for each user and the scenario are given in Matrix (3.2), Matrix (3.3), and Matrix (3.4). The graph for this scenario is given in Figure 3.3.

$$\text{User 1} = \begin{array}{c|cccc} & C_1 & Log_1 & CC_1 & CC_3 \\ \hline C_1 & 0 & 0 & 5 & 0 \\ Log_1 & 10 & 0 & 5 & 1 \\ CC_1 & 5 & 5 & 0 & 0 \\ CC_3 & 0 & 1 & 0 & 0 \end{array} \tag{3.2}$$

$$\text{User 2} = \begin{array}{c|cccc} & C_2 & Log_2 & Em_1 & CC_2 \\ \hline C_2 & 0 & 7 & 2 & 4 \\ Log_2 & 7 & 0 & 0 & 4 \\ Em_1 & 2 & 0 & 0 & 2 \\ CC_2 & 4 & 4 & 2 & 0 \end{array} \tag{3.3}$$

$$\text{Scenario} = \begin{array}{c|cccccccc} & C_1 & Log_1 & CC_1 & CC_3 & C_2 & Log_2 & Em_1 & CC_2 \\ \hline C_1 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 \\ Log_1 & 10 & 0 & 5 & 1 & 1 & 0 & 0 & 0 \\ CC_1 & 5 & 5 & 0 & 0 & 0 & 0 & 0 & 0 \\ CC_3 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ C_2 & 0 & 1 & 0 & 1 & 0 & 7 & 2 & 4 \\ Log_2 & 0 & 0 & 0 & 0 & 7 & 0 & 0 & 4 \\ Em_1 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 2 \\ CC_2 & 0 & 0 & 0 & 0 & 4 & 4 & 2 & 0 \end{array} \tag{3.4}$$

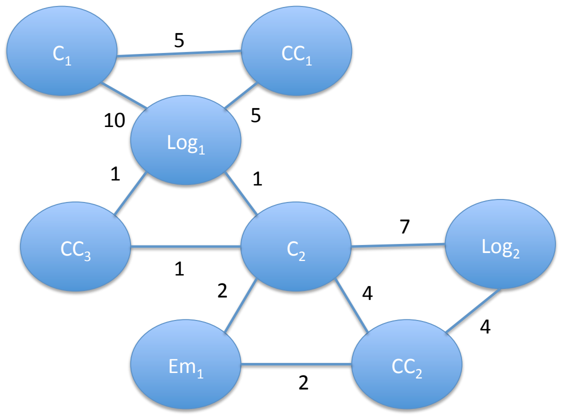

Figure 3.3: This is the graph generated from Matrix (3.4).

The algorithm starts by letting the universal identifier set (Uid) be empty. Now determine if the graph G (Figure 3.3) has more than one user. Table 3.5 gives the fp that i users are in G with 3 Logs/Ems and 3 CCs.

Users	fp(Users)
1	4.35%
2	12.42%

Table 3.5: Since there the fp for two users is higher than one user, the graph is merged.

Here, fp(2) > fp(1) so G needs to be cut. Before it is cut, it is merged. Recall that the Merge order is determined by the initial graph. The list of merges to give G in Figure 3.4 is as follows:

- C_1, CC_1, and CC_3 is absorbed into Log_1 since Log_1 is the only core node they are connected to.
- Edge weight CC_3C_2 is added to edge weight Log_1C_2 since CC_3 was absorbed into Log_1.
- C_2 and CC_2 are absorbed into Log_2 since that is their greatest core node edge.
- Edge weight Em_1C_2 is added to edge weight Em_1CC_2 since CC_2 and C_2 were absorbed into Log_2.

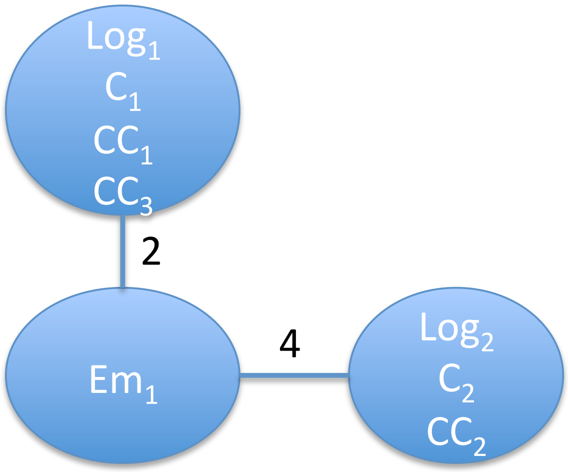

Figure 3.4: Graph G after the merge.

It is clear that Stoer-Wagner going to cut the edge with weight 2 as shown in Figure 3.5. Now there are two subgraphs $\{S_1, S_2\}$ given in Figure 3.6 and Figure 3.7 that need to be considered, respectively.

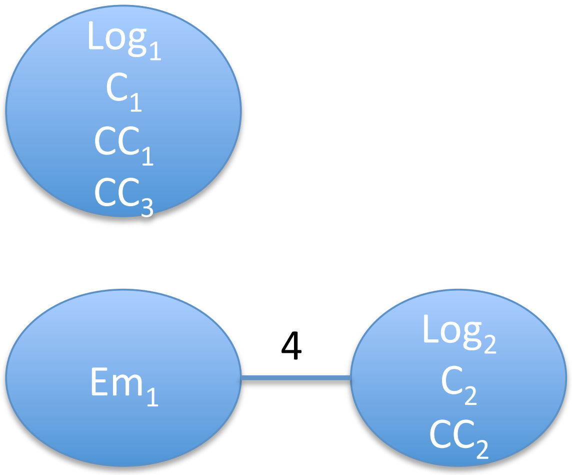

Figure 3.5: The graph G after Stoer-Wagner makes the cut.

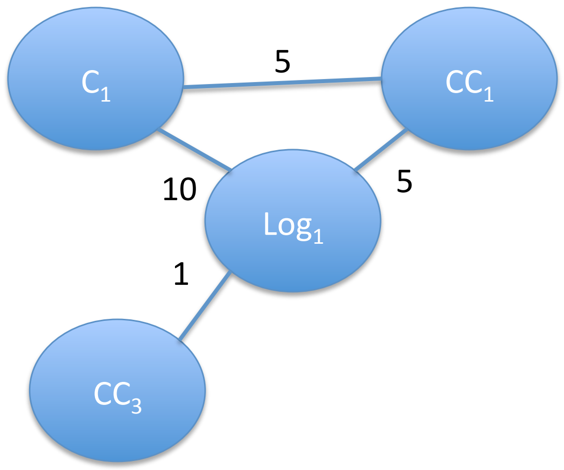

Figure 3.6: This graph shows the expanded nodes for S_1.

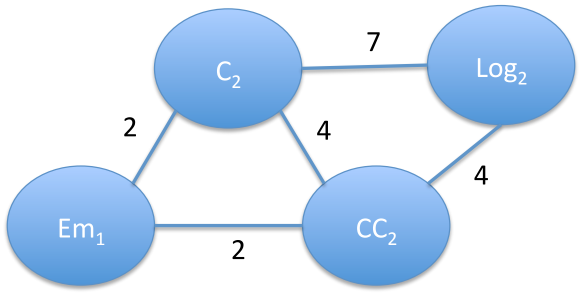

Figure 3.7: The graph shows the expanded nodes for S_2.

Since S_1 has only one Log/Em then it can be at most one user. S_2 has 2 Logs/Ems and 1 CC so fp must be checked. Table 3.6 shows that S_2 is only one user. Thus, the algorithm correctly distinguishes the two users where Uid $= \{S_1, S_2\}$.

Users	fp(Users)
1	18.37%
2	2.67%

Table 3.6: There can be at most 2 users since there are only 2 Logs/Ems in S_2. It is obtained that fp(1) > fp(2).

3.6 Results

The algorithm was tested on 4 sets of 500 clusters of n people for $n \in \{2, 3, 4, 5\}$. Due to time constraints, the algorithm was evaluated based on how many times it was able to determine the correct number of users. Table 3.7 gives the results of the tests for how many times the exact number of users were chosen correctly or within one of the correct answer. In the future, the accuracy of the information gathered for each user should be investigated.

	$n = 2$	$n = 3$	$n = 4$	$n = 5$
Comp $= n$	56% – 75%	63% – 68%	67% – 69%	65% – 69%
Comp $\in [n-1, n+1]$	98%	93%	98%	97%
Mean(Comp)	1.7	2.8	3.9	4.9

Table 3.7: The results for the number of components (comp).

4 Complexities

A few difficult situations arose while finding a solution to this problem. One situation that creates complexities is the *Public Library Problem* shown in Figure 4.1. The *Public Library Problem* describes a very busy cookie with many different emails and log ins attached to it. It is hard to decipher who should own the cookie.

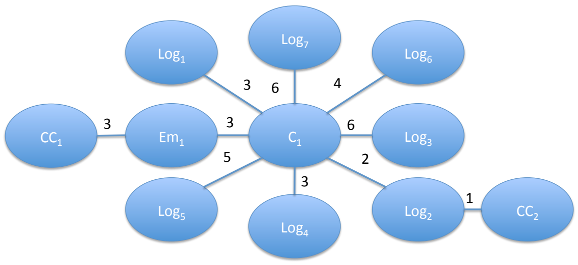

Figure 4.1: This is an illustration of the *Public Library Problem*.

Another issue arises when the Feasibility function does not recommend to cut when the edge weight suggests that there are multiple users. In Figure 4.2, it seems that this is two people where the a user logged in on someone else's computer once maybe to track a package. However, fp(1) = 18.38% > fp(2)= 3.84% so this graph would be called one person according to the algorithm. A possible solution is to have an additional filter that takes into account the edge weights as some ratio to each other.

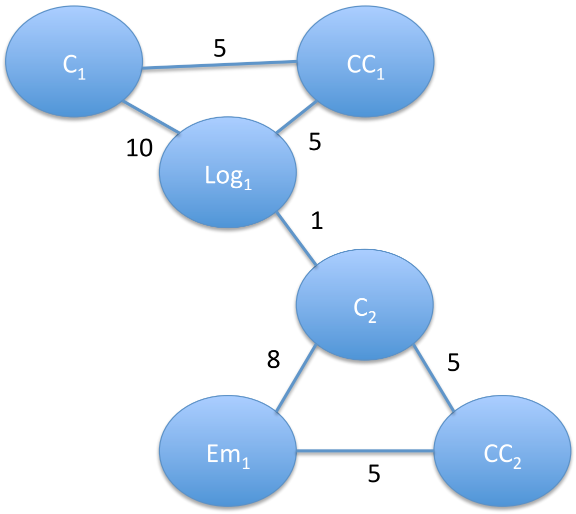

Figure 4.2: The edge weights in this graph give the impression that this is multiple people. If so, edge Log$_1$C$_2$ should be cut.

A final issue that should be considered is when there is a tie in edge weights after the end of the Merge

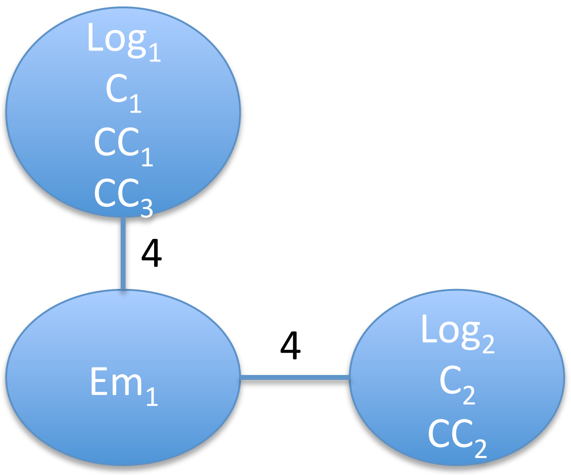

Figure 4.3: In the case of tie as seen here, the cut is made randomly.

as illustrated in Figure 4.3. In this case, Stoer-Wagner will not know which edge to cut. This means, that this cut is random and could be wrong. A solution would be to consider the information contained in the core nodes after the Merge. The edge weights that are absorbed could be a determining factor.

5 Conclusion

In conclusion, this paper introduces an algorithm that can universally identify users based on the information that obtained from their online shopping purchases. A method for generating synthetic data is implemented involving using blocks of weighted adjacency matrices. The algorithm creates a graph based on the matrices, and feeds them to the algorithm. Then the algorithm iteratively cuts the graph until unique users and their information can be identified.

In the future, the data will be in disconnected graphs with millions of nodes. The computers at Target Headquartera, however, will be able to parallelize the algorithm on all of the disconnected graphs to make the algorithm much faster. Also, the statistics used in this work will be made more robust from the resources that can be accessed by Target scientist.

References

[1] Mechthild Stoer and Frank Wagner, *A simple min-cut algorithm*, Journal of the ACM **44** (1997July), no. 4, 585–591. ↑5