A Distributed Graph Algorithm for Discovering Unique Behavioral Groups from Large-Scale Telco Data

Qirong Ho[1] Wenqing Lin[2]* Eran Shaham[1] Shonali Krishnaswamy[1]
The Anh Dang[3] Jingxuan Wang[3] Isabel Choo Zhongyan[3] Amy Shi-Nash[4]

[1]Institute for Infocomm Research, Singapore [2]Qatar Computing Research Institute
[3]DataSpark†, SingTel, Singapore [4]Commonwealth Bank of Australia

{hoq,eran-shaham,spkrishna}@i2r.a-star.edu.sg wlin@qf.org.qa
{anhkeen,jingxuan,isabelc}@singtel.com amy.shi-nash@cba.com.au

ABSTRACT

It is critical for a large telecommunications company such as Singtel to truly understand the behavior and preference of its customers, in order to win their loyalty in a highly fragmented and competitive market. In this paper we propose a novel graph edge-clustering algorithm (DGEC) that can discover unique behavioral groups, from rich usage data sets (such as CDRs and beyond). A behavioral group is a set of nodes that share similar edge properties reflecting customer behavior, but are not necessarily connected to each other and therefore different from the usual notion of graph communities. DGEC is an optimization-based model that uses the stochastic proximal gradient method, implemented as a distributed algorithm that scales to tens of millions of nodes and edges. The performance of DGEC is satisfactory for deployment, with an execution time of 2.4 hours over a graph of 5 million nodes and 27 million edges in a 8-machine environment (32 cores and 64GB memory per machine). We evaluate the behavioral groups discovered by DGEC by combining other information such as demographics and customer profiles, and demonstrate that these behavioral groups are objective, consistent and insightful. DGEC has now been deployed in production, and also shows promising potential to extract new usage behavioral features from other data sources such as web browsing, app usage and TV consumption.

*Qirong Ho and Wenqing Lin contributed equally to this work. This work was done while the author was with the Institute for Infocomm Research, Singapore.

†DataSpark is a wholly owned subsidiary of SingTel Group, providing advanced insight service to the group and beyond.

CIKM'16 , October 24–28, 2016, Indianapolis, IN, USA

DOI: http://dx.doi.org/10.1145/2983323.2983354

Keywords

Behavioral groups; Graph; Edge-clustering; Large scale distributed implementation; Telecommunications; CDR

1. INTRODUCTION

The telecommunications industry is undergoing a significant digital transformation globally. The market is highly competitive, not only among the mobile operators themselves, but also with popular (and mostly free) Over The Top (OTT) service providers. Human communications, particularly among the younger generation, are moving away from traditional calls (especially international calls) and text messages to alternatives such as FaceTime, Google Hangout, Skype, instant messaging and social media. Mobile operators are therefore under real revenue threats as well as the danger of losing relevance in the consumers' mind, and urgently need to increase their capabilities in understanding customer needs, behavior patterns and preferences, in order to stay competitive, win customer loyalty and continue revenue growth in the long run.

Solving this real business problem would have significant impact to mobile operators, yet classic analysis methods such as demographic profiling, micro segmentation and churn prediction [4, 18] (to name a few) — as useful as they are — lack the ability to discover new/emerging and often very fragmented behavior patterns, in an unsupervised and automated fashion from hundreds of millions of service and network data events that are automatically registered on a telco network when mobile devices are connected. Moreover, while there has been substantial research into generating intelligence from alternative, feature-rich data sources [6, 15] — such as customer profiles collected during sign-up, social media, mobile applications — these sources tend to suffer from noise and missing data (such as when customers do not disclose, or provide misleading information). In contrast, service and network data events — which include call detail records (CDRs), signal strength and network type (4G/LTE, 3G, 2G), cell towers connected, application launched, data volume, connection speed, success or failure of the connection, etc. — are reliable due to being automatically registered, anonymized and encrypted by the network, where anonymization and encryption are carried out in streaming

mode at the mobile operators in order to protect data privacy and security. Furthermore, the data do not include any personal information such as registration details, product subscription or billing. The opportunity here is to find a scalable and unsupervised learning algorithm that can uncover behavior patterns from large amounts of service and network data.

Service and network data is frequently *relational*, i.e., graph-structured, because the data events usually involve two connected entities — e.g., caller and callee, or subscribers connecting to cell towers and internet hosts. In this work, we would like to extract *unique behavioral groups* (subscribers with similar behavior) from such relational data in a fully-unsupervised manner, which can later be combined with other data sources (customer profiles, etc.) to discern further insights about each behavioral group. To give an example, we may find a group made up of young dating men, characterized by calls placed during weekend nights (determined from service and network data), by male subscribers in the 20-30 age group (determined from other data such as customer profiles). Without the call behavior inferred from service/network data, we could still be finding "young men" from the gender and age recorded in customer profiles, but not necessarily "young *dating* men" (which is a *behavior*).

As a starting point to ground our work, we use Call Detail Records (CDRs) of Singtel subscribers as the empirical data in this study — while noting that our proposed method can be applied to other relational service and network data events. The CDR dataset is also highly representative of Singapore's entire population as Singtel is the largest telecommunications company in Singapore. Previous works [12, 7, 13] on CDR data analyzed macroscopic properties of CDR graphs, but did not directly address the problem of extracting behavioral groups. As a first attempt, given the CDR graph data, one might turn to community detection algorithms from the social networks literature [19]; however, we observe that the CDR graph communities generated by such algorithms are unlikely to correspond to behavioral groups at a useful resolution — for example, some algorithms only return 2-3 communities, while others return thousands of communities with less than 10 nodes. In our view, this issue stems from the mismatch between classical *graph community detection*, which finds sets of nodes with strong homophily and inter-connectivity, and our task of finding *groups of nodes with similar behavior*, which may not necessarily be well-connected in the graph. For example, the group of young dating men are unlikely to call each other, but potential dates instead. One might consider local community detection algorithms [10] that discover communities centered around a seed node, but this introduces a difficult alignment problem, where the outputs from different seed nodes need to be reconciled into coherent behavioral groups, which renders this approach impractical.

Motivated by this challenge, we develop an effective method to discover behavioral groups within service and network data, such as CDR graphs. Our method, which we call Distributed Graph Edge Clustering (DGEC), discovers behavioral groups by using not only the graph's connectivity, but also the *features* of each data event (which corresponds to an edge in the graph), such as timestamp, duration, location, connection speed, data volume, etc. The idea is that nodes within the same behavioral group exhibit similar behaviors, so their actions (corresponding to edges) should have similar feature values as well. However, simply clustering the edge attributes (e.g. via a typical clustering algorithm) will not solve the problem reliably, because many calls may simply happen to resemble a behavioral group by chance — consider how a business call occurring during a weekend night does not qualify as behavior of "young dating men", even though it may have similar timestamp, duration and location by chance. In order to ensure DGEC is robust against this issue, we also incorporate the graph's connectivity to choose only the most significant behavioral groups per subscriber, which typical clustering algorithms do not perform. This also has the benefit of making DGEC's output easier to visualize and interpret.

More formally, DGEC is an optimization model that discovers (1) K behavioral groups within service/network graphs (which are directed, multi-edge, and have features on each edge), and (2) which of the K behavioral groups each edge and node is affiliated with (where nodes can belong to multiple groups). We derive a stochastic proximal gradient algorithm to solve for the model, and an efficient data-parallel strategy that we implement on a distributed Machine Learning framework (Petuum [21]). The resulting implementation can easily process graphs with tens of millions of nodes and edges on a small cluster in a few hours; with this level of efficiency, we are able to analyze weeks to months worth of data in a single day. In addition, our method takes edge features as real-valued vectors, and therefore admits feature engineering to bolster its sensitivity to behavioral groups. Finally, we conclude with an application and visualization on CDR data, in which we combine our method's output with other data sources such as customer profiles, in order to discover and analyze interesting behavioral groups.

2. EXPERIMENTAL GRAPH DATA SET

To ground our method on a real telco service and network dataset, we consider data from Singtel that consists of all anonymized CDR records in January 2015. These CDR records span millions of subscribers and tens of millions of calls. Each raw CDR record contains 5 attributes: date, time of day, call duration, caller ID, and receiver ID. Before converting the CDR records into a graph, we performed basic outlier removal by removing calls whose duration exceeded 3 hours (2 orders of magnitude above the mean/median call duration). The pruned calls made up $< 0.1\%$ of the calls, and $< 0.01\%$ of the subscribers were removed as a result of the pruning (because they had no calls left).

We converted the CDR records into a directed simple graph (with attributes on the edges), by placing a directed edge e for each CDR record from caller u to receiver v; we permit multiple edges between a caller-receiver pair (u, v). Each edge e is associated with a feature vector, generated from the attributes of the corresponding CDR record. We performed light feature engineering to generate the following edge features: hour of day, day of week, duration of the call, time period of day (e.g., morning, afternoon, and evening), total number of calls between the two subscribers, and the degrees of the two subscribers on the CDR graph. The resulting graph exhibits the following notable properties:

- Node degrees, call frequencies and call durations follow a power-law distribution. Most calls were made during working hours (9am to 6pm); the mean call du-

ration was approximately 170s, while the median call duration was roughly 50s.

- The graph is very sparse — the median outdegree and indegree are 0 and 3 respectively, and the mean degree (indegree plus outdegree) is 4.9.
- The maximum outdegree is 1885, and the maximum indegree is 148160.
- The graph is well-connected: 25% of nodes belong to a giant strongly connected component, and 99% of nodes are in a giant weakly connected component.
- Node local clustering coefficients (which measure the community strength around a node) fall off quickly with increasing node degree. In other words, subscribers that send or receive many calls do not belong to clear communities, because they are hub-like structures in the graph (e.g. call centers).

2.1 Related Work: Community Detection

We ran a variety of community detection algorithms on our CDR graph, in order to see if they could detect interesting behavioral groups amongst the subscribers. All of the implementations are single-machine (non-distributed). In the process, we made the following observations:

Linkcomm [1] partitions edges by performing hierarchical clustering on the Jaccard Similarity computed between edges. Because Linkcomm needs to instantiate a size-M^2 matrix (where M is the number of edges), it was unable to run on graphs with tens of millions of edges (including our CDR graph), even on machines with 128GB memory.

METIS [9] is a large-scale graph partitioning algorithm that produces non-overlapping communities through a multilevel algorithm. METIS finished execution within half an hour on our graph, but produced K communities of roughly equal size. As such, we were not able to associate these communities with meaningful subscriber behaviors.

Louvain [5] is a non-overlapping community detection algorithm, based on a greedy bottom-up hierarchical approach that uses modularity to test for good communities. The algorithm finished execution in only 5 minutes, but generated over 23000 communities, of which 90% contained < 4 subscribers. Due to the large number of tiny communities, we were not able to easily interpret the output.

Com2 [2] is a tensor decomposition method that is able to incorporate edge attributes (different from most other algorithms described here). We ran Com2 under two scenarios: (1) using the original adjacency matrix (with no edge features); (2) using a tensor that includes the adjacency matrix as well as the call duration feature. In the former case, the algorithm outputs only 3 tiny communities (size < 50 nodes), while in the latter case, the algorithm outputs zero communities.

OSLOM [10] is a local expansion method that grows communities from seed nodes. OSLOM took over one week to finish execution, and returned zero communities.

SPAEM [14] is an expectation-maximization algorithm based on a probability model, which measures the strength of association between all nodes and all communities. Unfortunately, we were not able to run the algorithm as it exceeded our servers' available memory (128GB).

GANXiSw [20] is an agent-based algorithm that uses label propagation to find communities. GANXiSW took approximately one week to finish execution, and returned < 10 very small communities with ≤ 20 subscribers.

Combo [16] is an optimization-based algorithm that moves nodes between communities until a high objective value is achieved. Combo required more than one week to finish, and returned zero communities.

In general, we found that the algorithms were not able to return useful behavioral groups — either the algorithms failed to execute (due to running out of memory), or they returned tiny communities with at most tens of subscribers, or (in the case of METIS) returned communities of roughly equal size. These observations motivate us to design a new approach to detecting behavioral groups, which can take advantage of additional information in service and network data — such as multiple edges between nodes, and edge features (date, time, etc.) — that is neglected by classical community detection approaches. Such information greatly helps in discovering behavioral groups, because it allows us to more easily distinguish between (for example) calls placed during working hours, and calls placed late at night in a bar or nightclub.

3. A MODEL OF BEHAVIORAL GROUPS

Motivated by the challenges facing classical community detection algorithms, we propose our own model of behavioral groups, which we define as subgraphs (edges and their adjacent nodes) that share similar edge features. The intuition is that behavioral groups can be readily distinguished when we consider edge features — for example, in CDR data, office calls are expected to fall into the time range between 9AM-5PM, while home calls would take place outside office hours. However, different from classical clustering algorithms, we also use the graph's connectivity to improve our model's robustness to data outliers. More specifically, we wish to balance two criteria:

1. **Behavioral groups have similar edge features:** we want to group edge features around K well-separated centroids, with each edge being assigned to its closest centroid. We shall use "centroid" and "behavioral group" interchangeably, when the context is clear.
2. **Nodes have few behavioral groups:** at the same time, we use the local graph connectivity around each node to make the centroids robust to chance, isolated edges that happen to resemble a behavioral group (e.g. weekend business calls may happen to look like calls from young dating men, but are not part of that behavioral group). This is accomplished by penalizing nodes whose adjoining edges are assigned to a large number of centroids; the effect is that every node (and its adjoining edges) may only be associated with a small number of behavioral groups.[1]

Because of these criteria, our algorithm will output behavioral groups that are markedly different from the communities produced by classical community detection algorithms — Figure 1 provides an intuitive illustration.

3.1 Minimization Problem

We formally express the above criteria as a minimization

[1]We note that this idea can be further generalized to encompass each node's egonet or even neighbors-of-neighbors, though at the cost of increased computational complexity — such investigation is left as future work.

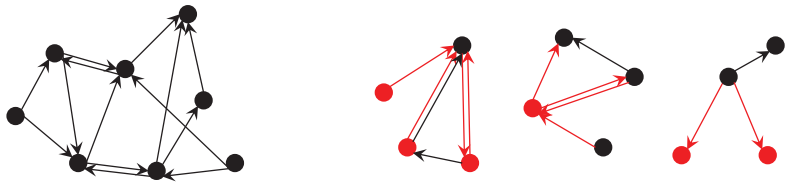

Figure 1: Difference between classical graph community detection and our method (behavioral group detection). Whereas communities are well-connected subgraphs, behavioral groups are sets of nodes and edges that share similar edge features — such as call timestamps and durations, which are indicative of behavior. In the example on the right, the red edges have similar features (e.g. calls placed on Friday night), and the red nodes are part of this behavioral group (e.g. dating men). Behavioral groups do not have to be strongly or weakly connected.

Variable	Type	Description
N	Input	Number of nodes. Nodes are indexed as $i \in \{1,\dots,N\}$
M	Input	Number of edges. Edges are indexed as $j \in \{1,\dots,M\}$
D	Input	Number of features in each edge
$u_1,\dots,u_M \in \mathbb{R}^D$	Input	Feature vectors for each of the M edges
$(x_1,y_1),\dots,(x_M,y_M)$	Input	Source node x_j and destination node y_j for each of the M edges
$\mathcal{E}_1,\dots,\mathcal{E}_N$	Input	Set of incoming and outgoing edges (integers $j \in \{1,\dots,M\}$) touching node i
K	Parameter	Desired number of behavioral groups. Each group is indexed as $k \in \{1,\dots,K\}$
α,β,γ	Parameter	Tuning parameters that control the algorithm's sensitivity
$c_1,\dots,c_K \in \mathbb{R}^D$	Output	Feature centroids for each of the K behavioral groups, in D-dim edge feature space
$a_1,\dots,a_M \in \{1,\dots,K\}$	Output	(Integer-valued) behavioral group assignments for each of the M edges
$\theta_1,\dots,\theta_N \in \Delta^{K-1}$	Output	Behavioral group participation vectors (probability distributions) for each node

Table 1: Inputs, parameters, and outputs for our DGEC, our behavioral group detection algorithm.

problem, whose quantities are defined in Table 1:

$$\min_{c,a,\theta} \sum_{j=1}^{M}\sum_{k=1}^{K} \mathbb{I}(a_j = k)||u_j - c_k||_2^2 + \alpha \sum_{i=1}^{N} \|\theta_i - \frac{1}{|\mathcal{E}_i|}\sum_{j\in\mathcal{E}_i} \text{Vector}(a_j)\|_2^2 + \sum_{i=1}^{N}\sum_{k=1}^{K}(1-\beta)\ln(\theta_{ik})$$
$$\text{s.t.} \quad \theta_i \in \Delta^{K-1}, \tag{1}$$

where $\mathbb{I}(x)$ is the indicator function equal to 1 when condition x is true (or 0 otherwise), and the function $\text{Vector}(x)$ converts an integer $x \in \{1,\dots,K\}$ into a K-dimensional indicator vector, i.e.

$$[\text{Vector}(x)]_k = \begin{cases} 1 & \text{if } k = x \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

In Eq. 1, the first term corresponds to criteria 1 (behavioral groups have similar edge features), and the second and third terms correspond to criteria 2 (nodes have few behavioral groups) — Figure 2 provides helpful visual intuition. Criteria 1 is a least-squares penalty that requires each behavioral group centroid c_k to be close to its assigned edge features u_j (where a_j is the variable assigning edges to behavioral groups). Criteria 2 is composed of two terms: the first term, which involves α, θ_i, a_j, tightly associates every node i with the behavioral groups assigned to adjacent edges (note that $\mathcal{E}_i$ is the set of edges adjacent to i). This association is represented by the probability vector θ_i, where element θ_{ik} denotes how much node i is associated with behavioral group k. The second term encourages θ_i to be sparse (few non-zeros), by penalizing elements θ_{ik} as they approach zero (observe that $\ln(1) = 0$, and $\ln(0) \to -\infty$) — the effect is to zero out elements θ_{ik} that are already near zero. Finally, the parameters α, β, γ are used to control the relative importance of the three terms; in our experiments, we found that the optimization was insensitive to the exact values, and setting $\alpha = 1, \beta = \gamma = 0.1$ sufficed to output good-quality behavioral groups.

3.2 A Tractable Algorithm via Relaxation

The minimization Eq. 1 involves discrete variables $a_1,\dots,a_M$ with an exponentially large state space, and is therefore computationally intractable. In order to achieve a practical solution, we relax each discrete variable a_j to a real-valued simplex vector $z_j \in \Delta^{K-1}$ (i.e. probability vectors over K choices[2]). For example, if $K = 3$, then $a_j = 2$ would be relaxed to the vector $z_j = [0,1,0]^\top$ — but note that z_j can also take values that do not correspond to any value of $a_j \in \{1,\dots,K\}$, such as $z_j = [0.1, 0.8, 0.1]^\top$. By applying this relaxation, we obtain a modified minimization problem

$$\min_{c,z,\theta} \quad L(c,z,\theta) = \sum_{j=1}^{M}\sum_{k=1}^{K} z_{jk}||u_j - c_k||_2^2 + \alpha\sum_{i=1}^{N}\|\theta_i - \frac{1}{|\mathcal{E}_i|}\sum_{j\in\mathcal{E}_i} z_j\|_2^2 + \sum_{i=1}^{N}\sum_{k=1}^{K}(1-\beta)\ln(\theta_{ik}) + \sum_{j=1}^{M}\sum_{k=1}^{K}(1-\gamma)\ln(z_{jk})$$
$$\text{s.t.} \quad z_j \in \Delta^{K-1}, \quad \theta_i \in \Delta^{K-1}, \tag{3}$$

where we have also added a new penalty term (the fourth term involving γ, z_{jk}) that encourages z_j to be sparse and thus close to a legitimate value of a_j. If we ignore this fourth term for the moment, we see that Eq. 3 is indeed a relaxation of Eq. 1, because they take the same objective value when

[2]That is, $z_j \in \mathbb{R}^K$ such that $z_{jk} \geq 0$ and $\sum_{k=1}^{K} z_{jk} = 1$.

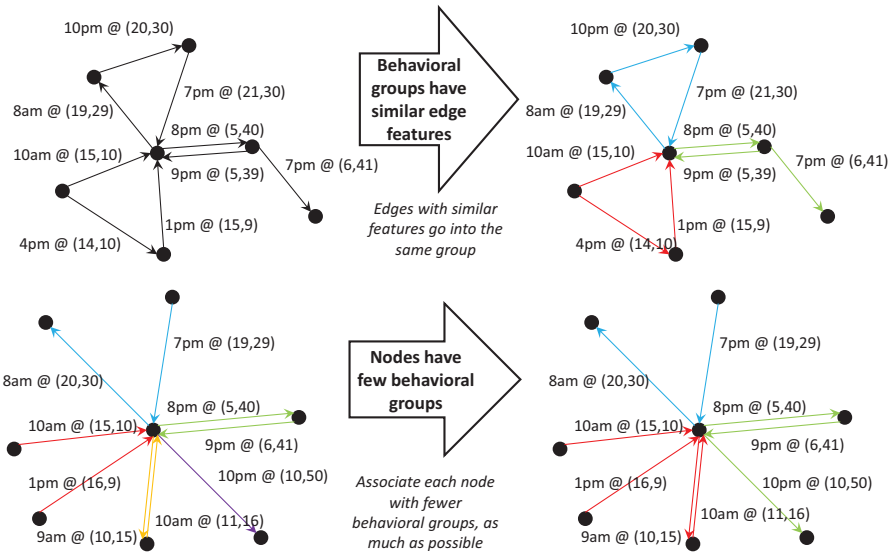

Figure 2: Two criteria for our behavioral group model: "behavioral groups have similar edge features" and "nodes have few behavioral groups". In this example, each edge j represents a phone call in a CDR graph, and has a 2-attribute feature vector u_j: "hour of day" and "location". Edge colors denote assignment to different behavioral groups. The first criteria assigns edges with similar feature values to the same behavioral group; the second criteria prevents nodes from associating with too many behavioral groups — this means that behavior must be *significantly repeated* (i.e. many similar calls) before a node and its edges can be assigned to a behavioral group.

$z_j = \text{Vector}(a_j)$ for all j (i.e. where every z_j corresponds to a legitimate value of a_j).

Having obtained a real-valued constrained minimization problem Eq. 3 in the variables c, z, θ, we may now solve it via the proximal gradient method. The algorithm proceeds by iterating the following update equations until convergence:

$$\begin{aligned} c_k &\leftarrow c_k - \eta \nabla_{c_k} L(c,z,\theta) \\ &= c_k - \eta \left[-\sum_{j=1}^{M} 2 z_{jk} (u_j - c_k) \right] \qquad (4) \\ \tilde{z}_{jk} &\leftarrow z_{jk} - \eta \frac{\partial}{\partial z_{jk}} L(c,z,\theta) \\ &= z_{jk} - \eta \left[\|u_j - c_k\|_2^2 \right. \\ &\quad \left. -\alpha \left(\sum_{i \in \{x_j, y_j\}} \frac{2}{|\mathcal{E}_i|} \left(\theta_{ik} - \theta_{ik}^{edge} \right) \right) + \frac{1-\gamma}{z_{jk}} \right] \\ z_j &\leftarrow \text{prox}_{\text{simplex}(\cdot)} \left([\tilde{z}_{j1}, \ldots, \tilde{z}_{jK}]^\top \right) \qquad (5) \\ \tilde{\theta}_{ik} &\leftarrow \theta_{ik} - \eta \frac{\partial}{\partial \theta_{ik}} L(c,z,\theta) \\ &= \theta_{ik} - \eta \left[2\alpha \left(\theta_{ik} - \theta_{ik}^{edge} \right) + \frac{1-\beta}{\theta_{ik}} \right] \\ \theta_i &\leftarrow \text{prox}_{\text{simplex}(\cdot)} \left([\tilde{\theta}_{i1}, \ldots, \tilde{\theta}_{iK}]^\top \right) \qquad (6) \\ \text{where } \theta_{ik}^{edge} &= \frac{1}{|\mathcal{E}_i|} \sum_{j' \in \mathcal{E}_i} z_{j'k}, \qquad (7) \end{aligned}$$

and where η is the gradient descent step size. The function $\text{prox}_{\text{simplex}(\cdot)}$ is the proximal operator for the simplex (i.e. probability vector) constraint — it projects θ, z to the closest point within the probability simplex Δ^{K-1}; we use the implementation in [17]. Note that the summation $\sum_{i \in \{x_j, y_j\}}$ simply selects the two nodes i touching the j-th edge. Once the proximal gradient descent has converged, we convert each vector z_j back to a discrete-valued a_j (which is one of the outputs of our method), by simply choosing the maximal element of z_j.

The proximal gradient equations Eqs. 4, 5, 6 are efficient and enjoy time complexity that is linear in N, M (i.e., number of nodes and edges respectively), because the quantity θ_{ij}^{edge} can be pre-computed at the beginning of each iteration. This ensures that, given a suitable distributed implementation over a cluster, our algorithm can scale to graphs of arbitrary size. The overall time complexity for the proximal gradient equations is $\mathcal{O}(M(KD + K\log(K)) + NK\log(K))$, where the $K\log(K)$ terms come from the proximal operator $\text{prox}_{\text{simplex}(\cdot)}()$ (which requires a sort). For practical values of K, the $\log(K)$ cost is negligible.

4. DISTRIBUTED IMPLEMENTATION

In order to efficiently and quickly process graphs with tens of millions of nodes and edges (if not more), we want to parallelize the proximal gradient steps Eqs. 4, 5, 6, with a distributed implementation that scales to any number of machines. The distributed setting introduces new considerations, such as how best to partition the input data u_j, as well as how to partition and efficiently synchronize the variables c_k, z_j, θ_i across the network. In particular, we note that the graph's connectivity may need to be considered in the data and variable partitioning strategy — observe that the second term in Eq. 3, as well as Eq. 7, are dependent on the edge neighborhood of node i.

While sophisticated partitioning strategies are certainly possible, we found that a *data-parallel* strategy suffices to achieve good performance. In this strategy, each paral-

lel worker $p \in \{1, \ldots, P\}$ is assigned a disjoint subset of edges[3] $E_p \subset \{1, \ldots, M\}$, as well as a disjoint subset of nodes $V_p \subset \{1, \ldots, N\}$ — that is to say, we partition the edges and nodes across the parallel workers.[4] On each parallel worker with edges E_p and nodes V_p, we split the proximal gradient computations into 2 main loops per iteration: EdgeLoop(), which iterates over edges to update c, z via Eq. 4, 5, and NodeLoop(), which iterates over nodes to update θ via Eq. 6; Algorithm 1 provides detailed pseudocode. In order to compute the objective function Eq. 3 in parallel, the implementation maintains a set of global variables $L_1, \ldots, L_p$, which store each machine p's contribution to the objective.

Recent work has shown that ML algorithms can exhibit better performance when running in a "bounded-asynchronous" fashion [21], as contrasted with synchronous execution in MapReduce or Spark [22]. We choose to implement our algorithm using the bounded-asynchronous JBösen system [21] (http://petuum.org), while noting that MapReduce or Spark implementations are also feasible (though we do not explore them). JBösen automatically synchronizes model variables between machines, via a distributed shared memory (DSM) programming interface that resembles single-machine multi-core programming.[5] Specifically, in Algorithm 1, the variables c_k, θ_i are treated as global and accessed through the JBösen API,[6] while the variables z_j are stored locally on each machine (via standard arrays). Compared to alternative distributed ML systems such as [11], an additional benefit to JBösen is convenience — it uses the Java programming language, and packages all code and dependencies into a single `jar` file that can be readily deployed to any machine with a Java Runtime Environment installation or the Hadoop YARN scheduler.

4.1 Initialization Strategies for c_k, z_j, θ_i

It is commonly accepted that optimization-based algorithms may be sensitive to the choice of initial values. In order to ensure our algorithm outputs consistent and reproducible results, we adapt the Kmeans++ initialization procedure [3] to our needs — essentially, we initialize the behavioral group centroids c_k to randomly-chosen edge feature vectors u_j, selected such that they are (with high probability) far apart from each other. Our initialization procedure follows three high-level steps:

1. Generate K well-spaced initial centroids c_k, by running the Kmeans++ initialization strategy on the edge attributes u_j.
2. Assign each z_j to its closest cluster centroid c_k, plus a small amount of random noise.
3. Initialize $\theta_i = \frac{1}{\mathcal{E}_i} \sum_{j \in \mathcal{E}_i} z_j$, plus a small amount of random noise.

Empirically, we observed that this initialization procedure yields reproducible outputs; furthermore, it greatly reduced

[3] E_p is not to be confused with $\mathcal{E}_i$ (edges touching node i).

[4] In our current implementation, the partitions are random and of equal size. One might fairly ask if pre-partitioning the graph via a graph cut algorithm (or similar method) may lead to improved performance, and we intend to investigate this in future work.

[5] A DSM interface allows the programmer to read/write to distributed model variables as if they are local arrays.

[6] The JBösen API allows global variables x to be either *read*, or *incremented* by a value y. In Algorithm 1, we represent *increments* via the notation $x \leftarrow x + y$ (i.e. "add y to x").

Algorithm 1 Distributed algorithm for solving Eq. 3

Inputs: $N, M, K, D, u_j, x_j, y_j, \mathcal{E}_i$ and P (num. parallel workers)
Outputs: c_k, z_j, θ_i
Declare global variables $c_k, \theta_i, \theta_i^{edge}, L_p$ on JBösen system
Partition M edges into P disjoint subsets $E_1, \ldots, E_P$, and assign E_p to parallel worker p
Partition N nodes into P disjoint subsets $V_1, \ldots, V_P$, and assign V_p to parallel worker p
Declare local variables z_j, z_j^{old} on worker p, for all edges $j \in E_p$
Initialize global variable c_k
Initialize local variables z_j on each worker
for each worker $p \in \{1, \ldots, P\}$ in parallel **do**
    // Initialize θ_i^{edge} with the contribution from each edge j
    for each edge $j \in E_p$ **do**
        $\theta_{x_j}^{edge} \leftarrow \theta_{x_j}^{edge} + \frac{z_j}{|\mathcal{E}_{x_j}|}$
        $\theta_{y_j}^{edge} \leftarrow \theta_{y_j}^{edge} + \frac{z_j}{|\mathcal{E}_{y_j}|}$
        $z_j^{old} \leftarrow z_j$
    end for
    `clock()` // JBösen API function to signal next iteration
    // Begin Stochastic Proximal Gradient (SPG) algorithm
    for iterations $t = 1$ to T **do**
        $L_p \leftarrow L_p - L_p$ // Zero out objective function variables
        EdgeLoop()
        NodeLoop()
        for each edge $j \in E_p$ **do**
            // Update contribution of z_j to θ_i^{edge}
            $\theta_{x_j}^{edge} \leftarrow \theta_{x_j}^{edge} - \frac{z_j^{old}}{|\mathcal{E}_{x_j}|} + \frac{z_j}{|\mathcal{E}_{x_j}|}$
            $\theta_{y_j}^{edge} \leftarrow \theta_{y_j}^{edge} - \frac{z_j^{old}}{|\mathcal{E}_{y_j}|} + \frac{z_j}{|\mathcal{E}_{y_j}|}$
            $z_j^{old} \leftarrow z_j$
            for each community $k \in \{1, \ldots, K\}$ **do**
                // update objective fn: add 1st term from Eq. 3
                $L_p \leftarrow L_p + z_{jk} \|u_j - c_k\|_2^2$
            end for
        end for
        `print`$(\sum_{p=1}^{P} L_p)$ // Print objective function
        `clock()` // JBösen API function to signal next iteration
    end for
end for

function EdgeLoop()
    // Iterate over edges to update c, z
    for each edge $j \in E_p$ **do**
        for each community $k \in \{1, \ldots, K\}$ **do**
            // update c: send gradient terms for edge j to c_k
            $c_k \leftarrow c_k + \eta\,[2 z_{jk}(u_j - c_k)]$
            // update z: perform gradient step on z_{jk}
            $\tilde{z}_{jk} \leftarrow z_{jk} - \eta \|u_j - c_k\|_2^2$
                $+ \sum_{i \in \{x_j, y_j\}} \frac{2\eta\alpha}{|\mathcal{E}_i|}(\theta_{ik} - \theta_{ik}^{edge}) - \eta \frac{1-\gamma}{z_{jk}}$
        end for
        // update z: apply simplex prox operator
        $z_j \leftarrow z_j + \left(-z_j + \text{prox}_{\text{simplex}(\cdot)}\left([\tilde{z}_{j1}, \ldots, \tilde{z}_{jK}]^\top\right)\right)$
    end for
end function

function NodeLoop()
    // Iterate over nodes to update θ
    for each node $i \in V_p$ **do**
        for each community $k \in \{1, \ldots, K\}$ **do**
            // update θ: perform gradient step on θ_{ik}
            $\tilde{\theta}_{ik} \leftarrow \theta_{ik} - 2\eta\alpha(\theta_{ik} - \theta_{ik}^{edge}) - \eta \frac{1-\beta}{\theta_{ik}}$
        end for
        // update θ: apply simplex prox operator
        $\theta_i \leftarrow \theta_i + \left(-\theta_i + \text{prox}_{\text{simplex}(\cdot)}\left([\tilde{\theta}_{i1}, \ldots, \tilde{\theta}_{iK}]^\top\right)\right)$
        // update objective fn: add 2nd, 3rd terms from Eq. 3
        $L_p \leftarrow L_p + \alpha \|\theta_i - \theta_i^{edge}\|_2^2 + \sum_{k=1}^{K} (1-\beta) \ln(\theta_{ik})$
    end for
end function

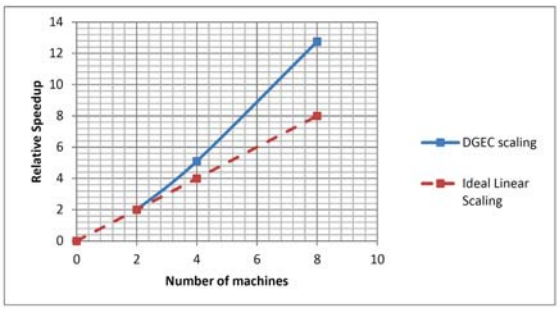

Figure 3: Scalability of DGEC implementation versus ideal linear scaling, from 2 through 8 machines, on a CDR graph with 5 million nodes and 27 million edges. DGEC required 15.3, 6, 2.4 hours to complete on 2, 4, 8 machines respectively (using $K = 20$ behavioral groups and 20 iterations, which were sufficient for convergence), demonstrating super-linear scalability going from 2 to 8 machines.

the number of iterations required for convergence — we found that 20 iterations sufficed to get c_k, z_j, θ_i to stabilize to 3 significant figures.

4.2 Scalability

Our implementation of DGEC processes our CDR graph (to be analyzed in Section 5), using $K = 20$ behavioral groups[7] in just 2.4 hours on 8 machines, each of which has 32 CPU cores and 64GB RAM, while Figure 3 shows that our implementation enjoys good scalability as more machines are added. In particular, we observed super-linear scalability going from 2 to 8 machines[8]; this is because JBösen automatically partitions the global model variables c, θ across the machines, and having more machines provides more total bandwidth to communicate variables and their updates.

5. APPLICATION AND VISUALIZATION

5.1 Statistics and differences with K-means

DGEC discovered $K = 20$ distinct behavioral groups on our sample data of approximately 5 million nodes and 27 million edges. Figure 4 plots the size of each group, ranging from the low hundreds of subscribers to about 2 million. We note that only 15 groups had significant mass ($> 500,000$ subscribers), and this suggests that $K = 20$ groups is sufficient for this dataset. In Figure 5, we also plot a histogram of the number of groups each subscriber belongs to, which ranges from 1 to 8 groups per subscriber; that is to say, each subscriber exhibits up to a maximum of 8 distinct calling patterns (with the majority having 5 or fewer).

The penalty terms in Eq. 3 distinguish DGEC from clustering algorithms that do not consider graph structure, such as K-Means. On the same dataset with $K = 20$ groups, we compare the statistics of K-means (using the K-means++ initialization [3]) against DGEC: Figure 4 shows that groups output by DGEC have more similar sizes than K-means, and Figure 5 shows that DGEC assigns subscribers to fewer groups than K-means — whereas K-means assigns 26% of subscribers to more than 5 groups, DGEC assigns only 12% of subscribers to more than 5 groups. These observations align well with the modeling criteria for DGEC in Section 3; in particular, they confirm that DGEC is robust to noisy or insignificant group assignments (Figure 2), making the results more reliable and easier to interpret.

[7]We also tried using more behavioral groups, e.g. $K = 100$, but found the output to be qualitatively similar to $K = 20$, likely because we used mainly time and duration features. In future work, we intend to add more service and network data features, such as location information, which should allow more specific behavioral groups to be discovered.

[8]We do not include results for 1 machine, because it had insufficient memory to handle our experimental graph.

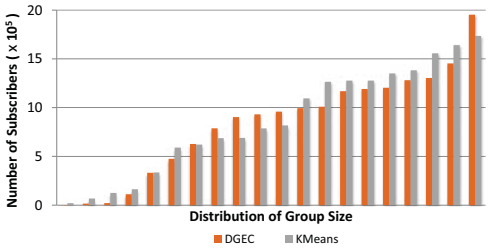

Figure 4: Size of behavioral groups.

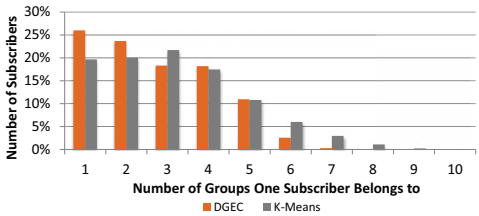

Figure 5: Histogram of the number of behavioral groups each subscriber belongs to.

5.2 Interpretation of Behavioral Groups

A particular benefit of DGEC is that it is able to capture different behavioral patterns of a single individual under varying scenarios, e.g. at work vs. social gatherings. One node (person) can thus belong to multiple behavioral groups depending on the circumstance. In explaining and validating these behavioral groups, we first selected the unique characteristics of an edge group, extracted the nodes (subscribers) belonging to the group, and then extracted other existing node attributes such as demographics, profiling and frequent locations. This allows us to construct a complete picture of a group in terms of their characteristics and behavioral patterns. Consistency in results across the various types of data extracted serves as validation of a behavioral group. For demonstration purposes, we selected 3 behavioral groups, referred to as Groups 3, 8, and 12.

The first example, Group 8, displays a unique calling pattern in terms of both time of day and days of the week which calls are most frequently made (Figure 6). Majority of calls were made by this group on Friday and Saturday nights, peaking from 9PM till midnight. The degree of incoming and outgoing connections is among the lowest (mean = 5) of all groups discovered. These findings indicate social activity among a small group of people.

We then studied in Figures 7 and 8 the demographics and customer profile of Group 8 to gain a deeper understanding of their behavioral pattern. Comparison of age distribution against a random sample of the general population indicated a much higher proportion of 20-29 year old subscribers in Group 8. Comparison of other subscriber characteristics,

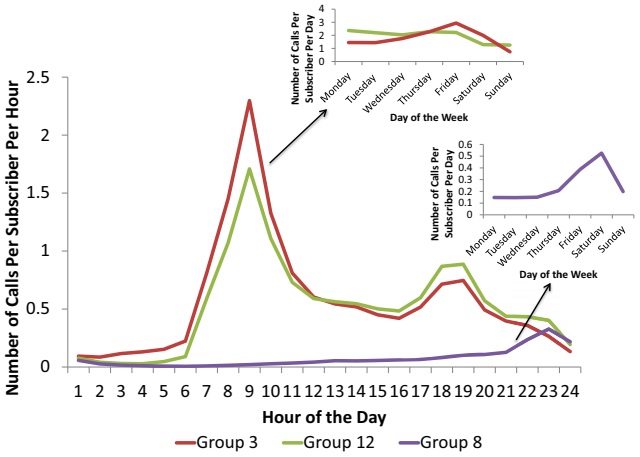

Figure 6: Number of calls per hour for behavioral Groups 3, 8, and 12.

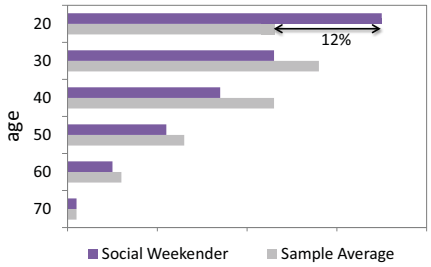

Figure 7: Age distribution of Group 8.

such as gender, race and nationality indicates that Singaporean, Chinese males have a higher likelihood of exhibiting this behavioral pattern. Due to commercial sensitivity of the data, this paper only displays indexed differences between the behavioral group against the general population.

Moreover, spatial information of this behavioral group captured during the peak call activity period further strengthens the assumption that calls made by this group serve a more social function as opposed to fulfilling business or family commitments. Figure 9 shows a concentration of the subscribers at late-night hotspots in Singapore such as Beach Road, Clarke Quay, Chinatown and River Valley. The supporting evidence leads us to draw the conclusion that subscribers in Group 8 largely make calls for leisure purposes, especially during weekends, and we thus name them "Social Weekenders".

The Social Weekender group is comprised largely of digitally-native Millennials, typically students or young adults; they are fluent in technology and spend a large amount of time on their mobile devices. They are more comfortable communicating through digital mediums like social media and messaging apps, as opposed to using phone calls, and this is evident in the low average call volume of this group. Phone calls are made mostly by a Social Weekender to establish more immediate plans or to contact friends upon their arrival at a pre-determined location. This pattern of behavior coincides with the peak call activity recorded for the group, which is a common timeframe within which late night weekend plans are typically made.

As for Groups 3 and 12, they exhibit clear similarities in their calling patterns: subscribers in both groups tend to make calls between 8-9AM and 5-6PM, which are just

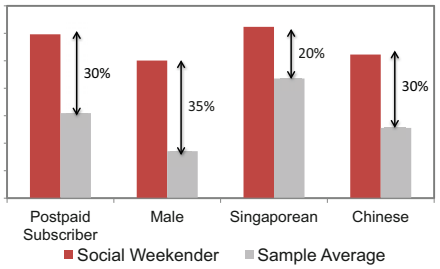

Figure 8: Demographic comparison for Group 8.

Figure 9: Group 8 hangouts on Saturday evening.

before and after typical working hours in Singapore(Figure 6). Subscribers in both clusters made calls mostly during weekdays, although a distinctly higher call volume was observed in Group 12 on Fridays and Saturdays. The variance observed could be attributed to differences in job requirements or individual sociability.

Similarly, subscriber characteristics were mapped against a random sample. Comparisons indicated that people in both groups were largely spread over the 20-29 and 30-39 year age groups(Figure 10), engaged in more routine work patterns and had a higher proportion of females and condominium dwellers than the sample. These findings lead us to conclude that individuals in Groups 3 and 12 are mostly young professionals of slightly higher income, who utilize their commuting time to make family arrangements. This is supported by the high proportion of Postpaid subscribers (93%) that make up these two groups, as Postpaid subscriptions are characteristic of higher income individuals as well as corporate mobile subscriptions(Figure 11). Spatial mapping of their work locations also indicated a high concentration in Singapore's Central Business District and surrounding areas, where most major financial and legal institutions are situated(Figure 12).

By combining these results with our internal "traffic monitoring module", we were able to discern the regular mode of transport utilized by 70% of individuals in Groups 3 and 12 [8]. Results indicate that three quarters of these individuals frequently utilized the Mass Rapid Transit (i.e. subway) system, further corroborating our earlier findings as this mode of transport allows individuals to easily make calls while commuting. Consequently, we jointly termed Groups 3 and 12 the "Productive Commuters" based on their calling patterns identified.

The behavioral groups can also be visualized within the local neighborhood of each node. Figure 13 shows a randomly chosen seed node from Groups 3 and 12, as well as its neighbours and neighbors-of-neighbors. The color of each

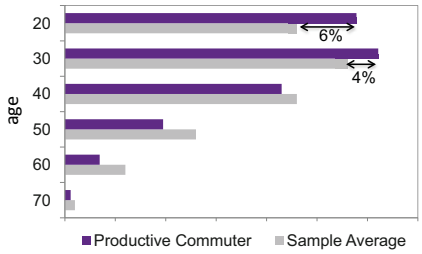

Figure 10: Age distribution of Groups 3 & 12.

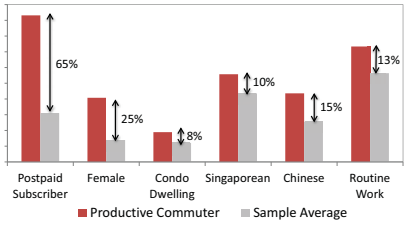

Figure 11: Demographics of Group 3 versus 12.

node v corresponds to its dominating behavioral group (i.e., the group with the maximum value in its behavioral group participation vector), and we color each edge e by the behavioral group that e belongs to. The size of a node reflects the number of neighbors it connected to, and the width of an edge represents the volume of calls between the two nodes. As shown in Figure 13, the seed node is connected to two large communities and a couple of smaller communities. The interesting observation is that, these two communities have different behavioral patterns, dominated by Group 12 (green) and Group 3 (light blue) respectively. Based on the behavioral description of Groups 3 and 12 shown earlier, these two groups are both likely to be work colleagues of the seed node, with the Group 12-dominated community exhibiting more weekend calls. We observe few other colors in this plot (save a few red nodes), indicating that the local neighborhood around the seed node is made up of similar types of subscribers.

5.3 Applications

By incorporating CDR data into customer profile analytics, mobile operators are able to identify more complete and robust customer profiles (including behavioral patterns), as well as to dynamically monitor the changing landscape. This allows for sharper targeting of customers, based on a more holistic understanding of customer preferences and needs. Such analysis adds value to mobile operators by providing the insight required for creation of products and services better suited to specific customer needs, as well as for a more targeted marketing and communications strategy based on customers' lifestyle patterns (by identifying optimal methods, timings and locations to engage customers with increased effectiveness).

For example, Social Weekenders are best engaged through experiential, relevant content over digital mediums such as app and web based platforms, due to the large proportion of Millennials that make up this group. Weekday afternoons are an ideal time to digitally target this group as they are less likely to be socially occupied and hence more receptive to digital mobile content. Understanding the preferences and needs of this group is especially applicable to Mobile Operators in maximizing revenue and preventing churn, as Millennials place high priority on product relevance and represent the group most likely to discover OTT services that cater better to their needs.

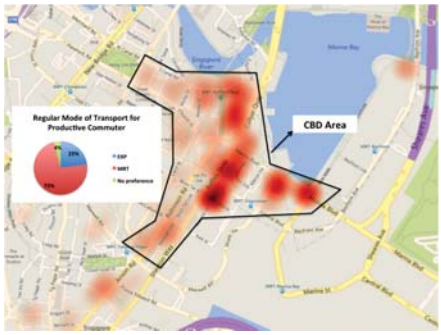

Figure 12: Working locations for Groups 3 & 12.

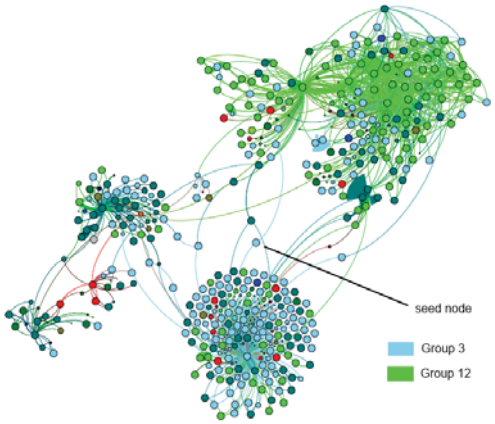

Figure 13: Local neighborhood of a random subscriber, tagged with behavioral groups.

Apart from commercial applications, DGEC can also serve as an additional resource for profiling customers internally. When subscriber information is collected during signup, certain fields pereceived as insignificant or sensitive tend to suffer from missing data, resulting in an incomplete customer profile. The extraction of behavioral clusters in this paper has enabled the prediction of missing data across several demographic categories. In the Productive Commuters group, the proportion of "NA" values was reduced from 47% in the random sample to 7% after clustering in the "Subscriber Type" category and from 69% to 25% in the "Gender" category. For example, if subscribers with incomplete profiles are identified to be Productive Commuters from their CDRs and available demographics, we can assume with high confidence that they are Postpaid subscribers due to the high proportion of Postpaid subscribers within the group (93%).

5.4 Deployment

We deploy DGEC in our analytic platform production environment as part of the "people module". This module aims to understand the unique behaviors of people and groups, based on their "digital footprints" from telco data, such as geo-spatial data, social data, and content. Other functionalities in the people module include home/work detection, top hang outs, inbound trips, outbound trips extraction, demographics predictions, web profiling, to name a few.

DGEC is run monthly as an offline batch job to extract customer behavioral groups from CDR data. As mentioned in Section 2, we perform light feature engineering by con-

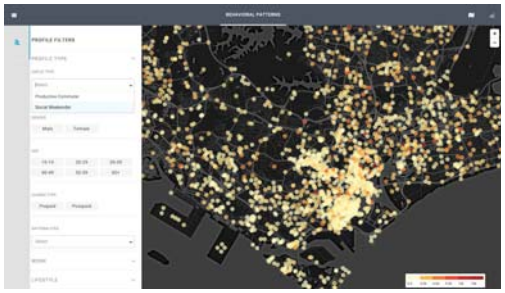

Figure 14: Snapshot of application interface.

verting raw CDR attributes into the following features: hour of day, day of week, duration of the call, time period of day (e.g., morning, afternoon, and evening), total number of calls between the two subscribers, and the degrees of the two subscribers on the CDR graph. Each customer is tagged with his/her behavioral group IDs, and the result is stored in *SOLR*, our internal big data store. Our Application Programming Interface (API) interacts directly with the database, and allows business intelligence users to make queries to generate new insights. Figure 15 shows the general workflow of the deployment.

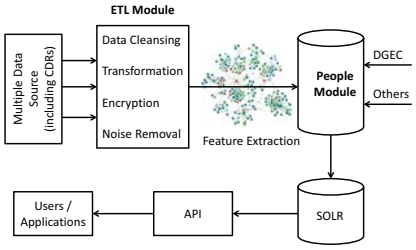

Figure 15: Deployment workflow including DGEC.

6. CONCLUSION

Graph approaches are fundamental to improving our understanding of consumer relationships and behaviors. We believe the combination of graph connectivity (between nodes) and behavior (through edge features) offers a significant improvement in discovering meaningful behaviour as well as relationships. The next step would be to expand the method to other data sources: for example, locations are another key aspect by which people form significant relationships, and telcos indeed have rich location information in service and network data. DGEC can be applied to identify different visiting patterns (edges) between people and places (nodes), and we expect that it can reveal features such as "purpose of visit", "likelihood to stay", "predicted next place", etc. Another valuable extension would be to hierarchically organize the behavioral groups, enabling business intelligence units to "drill down" from coarser groups to finer-grained ones. The ultimate goal is to have an unified and scalable learning method that can discover unique behavioral characteristics through the graph edge clustering method, in any service and network data. The business benefit of this capability could be a game changer to mobile operators.

7. REFERENCES

[1] Y.-Y. Ahn, J. P. Bagrow, and S. Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, 466:761–764, 2010.

[2] M. Araujo, S. Papadimitriou, S. Günnemann, C. Faloutsos, P. Basu, A. Swami, E. E. Papalexakis, and D. Koutra. Com2: Fast automatic discovery of temporal ('comet') communities. In *PAKDD*, pages 271–283, 2014.

[3] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *SODA*, pages 1027–1035, 2007.

[4] W.-H. Au, K. C. Chan, and X. Yao. A novel evolutionary data mining algorithm with applications to churn prediction. *Evolutionary Computation*, 7(6):532–545, 2003.

[5] V. D. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of community hierarchies in large networks. *CoRR*, abs/0803.0476, 2008.

[6] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *UAI*, pages 43–52, 1998.

[7] K. Dasgupta, R. Singh, B. Viswanathan, D. Chakraborty, S. Mukherjea, A. A. Nanavati, and A. Joshi. Social ties and their relevance to churn in mobile telecom networks. In *EDBT*, pages 668–677. ACM, 2008.

[8] T. Holleczek, D. T. Anh, S. Yin, Y. Jin, S. Antonatos, H. L. Goh, S. Low, and A. Shi-Nash. Traffic measurement and route recommendation system for mass rapid transit(mrt). In *KDD*, pages 1859–1868. ACM, 2015.

[9] G. Karypis and V. Kumar. Multilevel algorithms for multi-constraint graph partitioning. In *SC*, page 28, 1998.

[10] A. Lancichinetti, F. Radicchi, J. J. Ramasco, and S. Fortunato. Finding statistically significant communities in networks. *CoRR*, abs/1012.2363, 2010.

[11] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su. Scaling distributed machine learning with the parameter server. In *OSDI*, pages 583–598, 2014.

[12] A. Nanavati, R. Singh, D. Chakraborty, K. Dasgupta, S. Mukherjea, G. Das, S. Gurumurthy, A. Joshi, et al. Analyzing the structure and evolution of massive telecom graphs. *TKDE*, 20(5):703–718, 2008.

[13] V. Pandit, N. Modani, S. Mukherjea, A. Nanavati, S. Roy, A. Agarwal, et al. Extracting dense communities from telecom call graphs. In *COMSWARE*, pages 82–89, 2008.

[14] W. Ren, G. Yan, X. Liao, and L. Xiao. Simple probabilistic algorithm for detecting community structure. *Phys. Rev. E*, 79:036111, Mar 2009.

[15] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, pages 285–295. ACM, 2001.

[16] S. Sobolevsky, R. Campari, A. Belyi, and C. Ratti. A general optimization technique for high quality community detection in complex networks. *CoRR*, abs/1308.3508, 2013.

[17] W. Wang and M. A. Carreira-Perpinán. Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application. *arXiv:1309.1541*, 2013.

[18] C.-P. Wei and I.-T. Chiu. Turning telecommunications call details to churn prediction: a data mining approach. *Expert systems with applications*, 23(2):103–112, 2002.

[19] J. Xie, S. Kelley, and B. K. Szymanski. Overlapping community detection in networks: The state-of-the-art and comparative study. *ACM Comput. Surv.*, 45(4):43, 2013.

[20] J. Xie and B. K. Szymanski. Towards linear time overlapping community detection in social networks. In *PAKDD*, pages 25–36, 2012.

[21] E. Xing, Q. Ho, W. Dai, J.-K. Kim, J. Wei, S. Lee, X. Zheng, P. Xie, A. Kumar, and Y. Yu. Petuum: A new platform for distributed machine learning on big data. *Big Data*, PP(99):1–1, 2015.

[22] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *NSDI*, pages 2–2, 2012.