©Copyright 2016
Natalia Nebulishvili

Scalable Propagation Algorithms for Node Labeling in Bipartite Graphs

Natalia Nebulishvili

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science in Computer Science and Systems

University of Washington

2016

Committee:

Martine De Cock, Chair

Matthew Tolentino

Jacob Nelson

Golnoosh Farnadi

Program Authorized to Offer Degree:
Computer Science and Systems

University of Washington

Abstract

Scalable Propagation Algorithms for
Node Labeling in Bipartite Graphs

Natalia Nebulishvili

Chair of the Supervisory Committee:
Associate Professor Martine De Cock
Institute of Technology

Bipartite graphs are graphs whose vertices can be partitioned into 2 different independent sets. Many interesting machine learning problems either present themselves naturally as problems of node labeling in bipartite graphs (like user modeling in social networks) or can be converted into this form (like document classification). In both cases the nodes of the graph represent different entities with specific labels. Some of the labels are known a priori while others have to be learned.

Label propagation (LPA) is an approach in which labeled nodes in a graph propagate their labels towards neighbors iteratively to assign labels to unknown nodes. While LPA is fast for node labeling, it doesn't distinguish between different kinds of nodes in the graph, while our proposed algorithm can distinguish between 2 different kinds of nodes in the graph, and adapts itself automatically to the underlying characteristics of the network without previous knowledge.

Given the large size of the typical datasets to which Label Propagation Algorithms are applicable, we pay attention to ensuring and demonstrating the scalability of the developed node labeling algorithms. To allow our proposed techniques to efficiently process graphs with millions of nodes and edges, we designed implementations in Grappa, a distributed shared memory system. We performed experiments to measure the scalability and the accuracy of our node labeling algorithms on different sizes of datasets from Facebook.

TABLE OF CONTENTS

Page

LIST OF FIGURES

ACKNOWLEDGMENTS

Foremost, I would like to express my sincere gratitude to Prof. Martine De Cock for the continuous support of the research, for her patience, motivation, enthusiasm, and feedback. Her guidance helped me in all the time of the research and writing of this thesis. I could not have imagined having a better advisor during the research.

I would also like to thank the members of my thesis committee for their dedication to helping me in making the most of this research, their insightful comments and encouragement. I am sincerely grateful to them for sharing their views on a number of issues related to this work. This accomplishment would not have been possible without them.

I would like to express my gratitude to everyone else who supported and encouraged me throughout the course of my Master's thesis.

Thank you,

Natalia Nebulishvili

Chapter 1

INTRODUCTION

Label propagation is an approach in which labeled nodes in a graph propagate their labels towards neighbors iteratively to assign labels to unknown nodes [19]. It is a fast and simple approach which is used for example for community detection in social networks [17], for document classification [14, 15], for topic extraction [4], and to find overlapping communities in real world and biological networks [6]. One could imagine the nodes to correspond to users of a social network, the edges to denote friendship links, and the labels to denote characteristics such as age and gender of the users. In another application scenario, the nodes could correspond to academic papers, the edges to citations (in this case we should consider directed edges), and the labels to scientific disciplines. In this case we can classify the documents based on the citation network.

The original Label Propagation Algorithm (LPA) [19] relies on the homophily assumption, i.e. the assumption that connected nodes have the same or similar labels. Not all networks exhibit this behavior. A nice example are social dating networks, in which connected users tend to have opposite genders instead of the same gender. Recently an adaptive version of LPA was introduced that extends the original algorithm by dynamically adapting to the underlying characteristics of the graph [5]. Just like in traditional LPA, to label a node, Adaptive LPA takes into consideration the labels of the neighboring nodes, but then uses conditional probabilities learned over the entire network to choose the most probable one. Both LPA and Adaptive LPA do not distinguish between *types* of nodes in the graph: they assume that all nodes are of the same type.

In this thesis, we propose *new algorithms for adaptive label propagation that can be applied to bipartite graphs.* Bipartite graphs allow for a more fine grained representation

of some machine learning problems than ordinary graphs. For instance, referring back to the social networks example described above, we could consider a *bipartite graph* where one type of nodes corresponds to users, as before, while another type of nodes denotes items (pages), and edges correspond to so-called "likes" (i.e. the user likes the page or not).

Our hypothesis is that a more fine grained representation of machine learning problems through bipartite graphs can allow for more accurate node labeling than traditional LPA performed on ordinary graphs. We therefore propose new algorithms for label propagation and adaptive label propagation on bipartite graphs, and experimentally compare these strategies to traditional LPA.

To allow our algorithms to process graphs with millions of nodes and edges efficiently, we implemented them in Grappa [10], a modern framework that provides shared memory abstraction for clusters. Performance scales up even for applications that have poor locality and an input-dependent load distribution. Grappa's Distributed Shared Memory (DSM), tasking system and communication layer remedy deficiencies of earlier DSM systems. Grappa also implements common programming abstractions for big data, including a GraphLab-like [9] vertex-centric API which is suitable for the graph structures we have. We evaluated the algorithms and their Grappa-implementations in terms of predictive accuracy as well as scalability on large datasets from Facebook.

We first give an overview of the traditional LPA algorithm as well as the DSM-system Grappa that we use for the development and implementation of our algorithms in Chapter 2. We then provide a detailed description of the different algorithms which we propose for label propagation in bipartite graphs in Chapter 3. This includes an overview and motivation of the choices of stopping criteria used for the iterative algorithms, as well as details on their implementations in Grappa. In Chapter 4 we describe the datasets we used for testing our proposed methods. We present the design and results of our experiments in Chapter 5. In the last chapter (Chapter 6) we summarize our observations and discuss suggestions for future work. We also provide the implementation for all the algorithms which are discussed

in this thesis (please follow the link).

Chapter 2

BACKGROUND

2.1 LPA

The label propagation algorithm (LPA), proposed by Zhu and Ghahramani [19], propagates the labels of a subset of vertices through the network to label other vertices. The authors use seed nodes $X_L = \{x_1, x_2, \ldots, x_l\}$ and their corresponding labels $Y_L = \{y_1, y_2, \ldots, y_l\}$ to infer the labels $Y_U = \{y_{l+1}, y_{l+2}, \ldots, y_{l+u}\}$ of previously unlabeled vertices $X_U = \{x_{l+1}, x_{l+2}, \ldots, x_{l+u}\}$. The labels are drawn from a predefined set $C = \{c_1, c_2, \ldots, , c_n\}$.

Consider the undirected graph $G = (V, E)$, where V is the set of all nodes, i.e. $V = X_L \cup X_U$ and E is the set of edges (connections between nodes).

Let us assume that we have the $(l + u) \times n$ matrix Y, where the i-th row Y_i represents the label distribution for vertex i, and the element Y_{ik} represents the probability that the correct label for vertex i is c_k.

The algorithm can be interpreted in two ways, which are discussed in Sections 2.1.1 and 2.1.2. Section 2.1.1 contains the matrix multiplication based description by the algorithm as originally proposed by Zhu and Ghahramani [19]. Our interpretation of the algorithm, presented in Section 2.1.2, leverages a vertex-centric model of the graph. It allows us to look at one vertex at a time and leads to a scalable implementation on large graphs in Grappa.

2.1.1 A matrix multiplication view of LPA

The original LPA is based on matrix multiplication. Let T be a $(l + u) \times (l + u)$ transition matrix, whose elements are defined as follows:

$$T_{ij} = P(j \to i) = \frac{1}{n_j}$$

where n_j is the number of neighbors of vertex x_j and $P(j \to i)$ represents the probability to jump from vertex x_j to vertex x_i. If there is no edge from j to i, then $T_{ij}0 =$. A node with less neighbors can propagate its label more easily than one with more neighbors. LPA is an iterative algorithm, where on convergence inferred labels are assigned to initially unlabeled nodes based on their most probable soft labels which are interpreted as distributions over labels. Convergence is reached when none of the elements of the matrix Y is changing. As the result of LPA is based on the matrix Y, we need to initialize it at the first step of the algorithm, using the information we have about the seed nodes. Recall that Y_{ik} is the probability that the label of node x_i is c_k. We know that the label of seed node x_i is y_i, for $i = 1, 2, \ldots, l$, so we initialize Y_ik to 1 if $y_i = c_k$, and 0 otherwise. As we do not know any information about other nodes' labels, we can fill their rows in matrix Y with uniform probabilities ($\frac{1}{n}$) or with 0's. Both strategies prevent to propagate anything from non-seed nodes. We used the second approach in our implementations. The details of the algorithm are presented in Algorithm 1.

Algorithm 1 Matrix-based LPA

Input: Graph G, set of seed nodes X_L and their corresponding labels Y_L
Output: Set of labels Y_U for previously unlabeled nodes

for $i = 1, 2, \ldots, l + u$ **do**
  for $k = 1, 2, \ldots, n$ **do**
    if $y_i = c_k$ **then**
      $Y_{ik} \leftarrow 1$
    else
      $Y_{ik} \leftarrow 0$
    end if
  end for
end for
while Y has not converged **do**
  $Y \leftarrow T \cdot Y$
  Row-normalize Y ▷ Normalize each row of Y to sum up the probabilities to 1
  Reset $Y_1, Y_2, \ldots, Y_l$ to their original values
end while
for $i = l + 1, l + 2, \ldots, l + u$ **do**
  $y_i \leftarrow \arg\max_{c_k \in C} Y_{ik}$
end for

To understand the details of Algorithm 1, we apply it to a small graph and simulate an execution of each step.

Example 2.1.1 *Let assume that we are given a graph G, which is presented in Figure 2.1, a set of seed nodes* $X_L = \{0, 3, 4, 7, 8\}$*, their corresponding labels* $Y_L = \{y_0 =$ *"female",*$y_3 =$ *"female",* $y_4 =$ *"male",* $y_7 =$ *"female",* $y_8 =$ *"male"*$\}$ *and a set of possible labels* $C = \{c_0 =$ *"male",* $c_1 =$ *"female"*$\}$*. We need to infer labels for the vertices from the set* $X_U = \{1, 2, 5, 6\}$*. Let us take a specific element from the transition matrix to calculate. As vertex 1 has 2 neighbors, 0 and 2, we get* $T_{01} = T_{21} = \frac{1}{2}$*. All elements of transition matrix T are determined in a similar way:*

$$T = \begin{bmatrix} 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 1 & 1 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{4} & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 \end{bmatrix} \tag{2.1}$$

Lines 1-9: *Initialization of matrix Y. First column corresponds to "male", while the second one is for "female". As we know labels for vertices 0, 3, 4, 7 and 8, we filled the*

corresponding elements in these rows with 1; all other elements were set to 0.

$$Y = \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{2.2}$$

Let us consider 2 iterations. We will get the following changes:

Iteration 1, Line 11: *Propagation using the product of T and Y*

$$Y = \begin{bmatrix} 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 1 & 1 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{4} & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{2.3}$$

Iteration 1, Line 12: *Row-normalization*

$$Y = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ \frac{1}{2} & \frac{1}{2} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{1}{2} & \frac{1}{2} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{2.4}$$

Iteration 1, Line 13: *Reset the rows of the seed nodes to their original values from (2.2)*

$$Y = \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ \frac{1}{2} & \frac{1}{2} \\ 0 & 1 \\ 1 & 0 \\ 0 & 0 \\ \frac{1}{2} & \frac{1}{2} \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{2.5}$$

As we can see from matrix Y, all vertices which were the neighbors of seed nodes got some information from them. As neighbors of vertex 5 are non-seed nodes, they could not propagate anything at the first iteration. Thus the corresponding line in matrix Y did not change.

Iteration 2, Line 11: *Propagation*

$$Y = \begin{bmatrix} 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 1 & 1 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{4} & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ \frac{1}{2} & \frac{1}{2} \\ 0 & 1 \\ 1 & 0 \\ 0 & 0 \\ \frac{1}{2} & \frac{1}{2} \\ 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{2} \\ \frac{1}{8} & \frac{9}{8} \\ 1 & \frac{3}{2} \\ \frac{1}{8} & \frac{1}{8} \\ \frac{1}{8} & \frac{1}{8} \\ \frac{7}{24} & \frac{7}{24} \\ 1 & 1 \\ \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{6} \end{bmatrix} \tag{2.6}$$

Iteration 2, Line 12: *Row-normalization*

$$Y = \begin{bmatrix} 0 & 1 \\ \frac{1}{10} & \frac{9}{10} \\ \frac{2}{5} & \frac{3}{5} \\ \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} \tag{2.7}$$

Iteration 2, Line 13: *Reset the rows of the seed nodes to the original values from (2.2)*

$$Y = \begin{bmatrix} 0 & 1 \\ \frac{1}{10} & \frac{9}{10} \\ \frac{2}{5} & \frac{3}{5} \\ 0 & 1 \\ 1 & 0 \\ \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{2.8}$$

As we can see after the second iteration all nodes got some information from neighbor ones. Some vertices also influenced by the nodes which were not involve during the first iteration. For example vertex 2 got information from vertex 1 too and this gave it opportunity to change the distribution over labels. Now the label of vertex 2 is more "female" than "male".

According to the stopping criterion which was used during our implementation, the

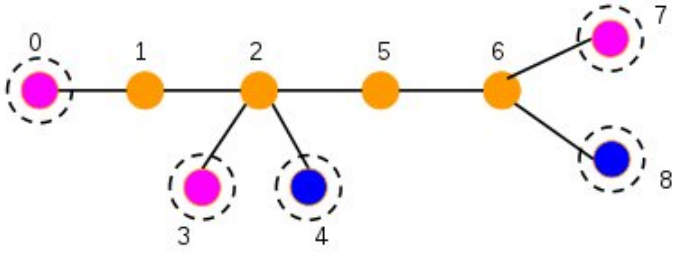


Figure 2.1: A small example graph. The seed nodes are circled.

iterative process stops after 5 iterations and the matrix Y takes form:

$$Y = \begin{bmatrix} 0 & 1 \\ \frac{74}{875} & \frac{801}{875} \\ \frac{1343}{3150} & \frac{1807}{3150} \\ 0 & 1 \\ 1 & 0 \\ \frac{566}{12254} & \frac{659}{1225} \\ \frac{173}{350} & \frac{177}{350} \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{2.9}$$

As we already know the matrix Y after iterative process is over, we can now go to Line 15 and infer the labels for non-seed nodes.

Lines 15-17 *As the second elements in the corresponding rows for all non-seed vertices in the matrix Y in (2.9) are greater than the first ones in the same rows, all non-seed nodes will get the label "female". In other words $y_1 = y_2 = y_5 = y_6 =$ "female".*

2.1.2 Vertex centric LPA

Unlike the matrix-based representation of LPA, which is discussed in Section 2.1.1, the vertex-centric approach is based on the information each vertex has. We will show that despite of this fact, it is exactly the same as matrix-based LPA. We just need to represent the matrix multiplication in terms of elements. For each Y_{ij} we get

$$Y_{ij} \leftarrow \sum_{k=1}^{l+u} T_{ij} Y_{kj}$$

By definition

$$T_{ik} = \frac{1}{n_k}.$$

Taking these facts into consideration the propagation step for each vertex will take the form:

$$Y_{ij} \leftarrow \sum_{k=1}^{l+u} \frac{1}{n_k} Y_{kj}. \tag{2.10}$$

This formulation gives us the possibility to collect the information for a vertex from its neighbors gradually without using the transition matrix. As we sum the information from neighbors step by step and also use the Y matrix from the previous iteration, we need some temporary memory for each node to accumulate information from neighbors in the correct way. The Label Propagation Algorithm in a vertex centric structure is presented in Algorithm 2. We use so-called GAS model, which includes three main phases: Gather, Apply and Scatter. During gather phase a node collects information from neighbors, Apply is executed on the node for which the information is gathered, and Scatter phase updates the information to neighbor vertices.

To illustrate that the matrix multiplication view of LPA is exactly the same as the vertex centric view, we will show its effect on the example graph.

Example 2.1.2 *Let us consider that we are given the same graph and initial data, which were used in Example 2.1.1. We will apply Algorithm 2 to them and discuss all main changes*

Algorithm 2 Vertex-Centric based LPA

for $i = 1, 2, \ldots, l + u$ **do**
  for $k = 1, 2, \ldots, n$ **do**
    if $y_i = c_k$ **then**
      $Y_{ik} \leftarrow 1$
    else
      $Y_{ik} \leftarrow 0$
    end if
  end for
end for
for each vertex j **do**
  $Temp_j \leftarrow \overrightarrow{0}$ ▷ Define a vector $Temp_j$ with n zeroes
end for
Gather information for all vertices from adjacent ones in $Temp$ vectors using Equation 2.10
repeat
  for each vertex j **do**
    Apply:
    if vertex j is not a seed node **then**
      $Y_j \leftarrow Temp_j$
      Normalize Y_j
      $y_j \leftarrow \arg\max_{c_k \in C} Y_{jk}$
    end if
    $Temp_j \leftarrow \overrightarrow{0}$
  end for
  for each vertex j **do**
    Scatter:
    for $i \in Neighbors(j)$ **do**
      $Temp_i \leftarrow Temp_i + Y_j / n_j$
    end for
  end for
until no y_j is changing

through the first 2 iterations. If we can not get a label for a node (Line 16), because all elements in corresponding row are equal, we will assign "unecided" to it.

Lines 1-9: *The initialization of matrix Y is identical as before*

$$Y = \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Line 13: *Let us first discuss the Gather phase for vertices 0 and 1. As vertex 0 has just one neighbor, which is vertex 1, it can only get information from vertex 1. As vertex 1 does not have any label, $Temp_0$ will become $(0,0)$. Vertex 1 has 2 neighbors: 0 and 2. When we collect information from them, we also take into account the number of neighbors that each of them has. So for vertex 1 we get: $Temp_1 = (0,1) + \frac{1}{4}(0,0)$. Thus $Temp_1 = (0,1)$. For the other vertices we get: $Temp_2 = (1,1)$, $Temp_3 = (0,0)$, $Temp_4 = (0,0)$, $Temp_5 = (0,0)$, $Temp_6 = (1,1)$, $Temp_7 = (0,0)$, $Temp_8 = (0,0)$.*

We present the results of the "Apply" and the "Scatter" phases into blocks to show that the computations happen at a vertex level, and they can be executed in parallel, independent of each other.

Line 17: *$j = 0$, vertex 0 is a seed node and we go to Line 22 in this case.*

Line 22: $Temp_0 = (0,0)$

Line 17: $j = 1$, *vertex 1 is not a seed node and we go to Line 18.*

Line 18: $Y_1 = (0, 1)$

Line 19: $Y_1 = (0, 1)$

Line 20: $y_1 =$ "$female$"

Line 22: $Temp_1 = (0, 0)$

Line 17: $j = 2$, *vertex 2 is not a seed node and we go to Line 18.*

Line 18: $Y_2 = (1, 1)$

Line 19: $Y_2 = \left(\frac{1}{2}, \frac{1}{2}\right)$

Line 20: $y_2 =$ *"undecided"*

Line 22: $Temp_2 = (0, 0)$

Line 17: $j = 3$, *vertex 3 is a seed node and we go to Line 22.*

Line 22: $Temp_3 = (0, 0)$

Line 17: $j = 4$, *vertex 4 is a seed node and we go to Line 22.*

Line 22: $Temp_4 = (0, 0)$

Line 17: $j = 5$, *vertex 5 is not a seed node and we go to Line 18.*

Line 18: $Y_5 = (0, 0)$

Line 19: $Y_5 = (0, 0)$

Line 20: $y_5 =$ *"undecided"*

Line 22: $Temp_5 = (0, 0)$

Line 17: $j = 6$, *vertex 6 is not a seed node and we go to Line 18.*

Line 18: $Y_6 = (1, 1)$

Line 19: $Y_6 = \left(\frac{1}{2}, \frac{1}{2}\right)$

Line 20: $y_6 =$ *"undecided"*

Line 22: $Temp_6 = (0, 0)$

Line 17: $j = 7$, *vertex 7 is a seed node and we go to Line 22.*

Line 22: $Temp_7 = (0, 0)$

Line 17: $j = 8$, *vertex 8 is a seed node and we go to Line 22.*

Line 22: $Temp_8 = (0, 0)$

Line 27: $j = 0, i = 1$, *vertex* $j = 0$ *has just one neighboring node, namely* $i = 1$, *to which it has to propagate its information.* $Temp_1 = (0, 0) + (0, 1)\,/1$.

We do not discuss details of the "Scatter" phase for the other nodes, and instead just present the results:

Line 27: $j = 1, i = 0$, $Temp_0 = \left(0, \frac{1}{2}\right)$

Line 27: $j = 1, i = 2$, $Temp_2 = \left(0, \frac{1}{2}\right)$

Line 27: $j = 2, i = 1$, $Temp_1 = \left(\frac{1}{8}, \frac{9}{8}\right)$

Line 27: $j = 2, i = 3$, $Temp_3 = \left(\frac{1}{8}, \frac{1}{8}\right)$

Line 27: $j = 2, i = 4$, $Temp_4 = \left(\frac{1}{8}, \frac{1}{8}\right)$

Line 27: $j = 2, i = 5$, $Temp_5 = \left(\frac{1}{8}, \frac{1}{8}\right)$

Line 27: $j = 3, i = 2$, $Temp_2 = \left(0, \frac{3}{2}\right)$

Line 27: $j = 4, i = 2$, $Temp_2 = \left(1, \frac{3}{2}\right)$

Line 27: $j = 5, i = 2$, $Temp_2 = \left(1, \frac{3}{2}\right)$

Line 27: $j = 5, i = 6$, $Temp_6 = (0, 0)$

Line 27: $j = 6, i = 5$, $Temp_5 = \left(\frac{7}{24}, \frac{7}{24}\right)$

Line 27: $j = 6, i = 7$, $Temp_7 = \left(\frac{1}{6}, \frac{1}{6}\right)$

Line 27: $j = 6, i = 8$, $Temp_8 = \left(\frac{1}{6}, \frac{1}{6}\right)$

Line 27: $j = 7, i = 6$, $Temp_6 = (0, 1)$

Line 27: $j = 8, i = 6$, $Temp_6 = (1, 1)$

As the first iteration is already finished we can compare it to the results from Example 2.1.1. If we take a look to the Matrix Y from (2.6), we can see that $Temp$ vectors contain the same information.

Line 17: $j = 0$, *vertex 0 is a seed node and we go to Line 22.*

Line 22: $Temp_0 = (0, 0)$

Step 17: $j = 1$, *vertex 1 is not a seed node and we go to Line 18.*

Step 18: $Y_1 = \left(\frac{1}{8}, \frac{9}{8}\right)$

Step 19: $Y_1 = \left(\frac{1}{10}, \frac{9}{10}\right)$

Step 20: $y_1 =$ *"female"*

Step 22: $Temp_1 = (0, 0)$

Line 17: $j = 2$, *vertex 2 is a seed node and we go to Line 18.*

Line 18: $Y_2 = \left(1, \frac{3}{2}\right)$

Line 19: $Y_2 = \left(\frac{2}{5}, \frac{3}{5}\right)$

Line 20: $y_2 =$ *"female"*

Line 22: $Temp_2 = (0, 0)$

Line 17: $j = 3$, *vertex 3 is a seed node and we go to Line 22.*

Line 22: $Temp_3 = (0, 0)$

Line 17: $j = 4$, *vertex 4 is a seed node and we go to Line 22.*

Line 22: $Temp_4 = (0, 0)$

Line 17: $j = 5$, *vertex 5 is not a seed node and we go to Line 18.*

Line 18: $Y_5 = \left(\frac{7}{24}, \frac{7}{24}\right)$

Line 19: $Y_5 = \left(\frac{1}{2}, \frac{1}{2}\right)$

Line 20: $y_5 =$ *"undecided"*

Line 22: $Temp_5 = (0, 0)$

Line 17: $j = 6$, *vertex 6 is not a seed node and we go to Line 18.*

Line 18: $Y_6 = (1, 1)$

Line 19: $Y_6 = \left(\frac{1}{2}, \frac{1}{2}\right)$

Line 20: $y_6 =$ *"undecided"*

Line 22: $Temp_6 = (0, 0)$

Line 17: $j = 7$, *vertex 7 is a seed node and we go to Line 22.*

Line 22: $Temp_7 = (0, 0)$

Line 17: $j = 8$, *vertex 8 is a seed node and we go to Line 22.*

Line 22: $Temp_8 = (0, 0)$

The "Apply" phase is already finished and now we are going into the "Scatter" phase.

Line 27: $j = 0, i = 1$, $Temp_1 = (0, 1)$

Line 27: $j = 1, i = 0$, $Temp_0 = \left(\frac{1}{20}, \frac{9}{20}\right)$

Line 27: $j = 1, i = 2$, $Temp_2 = \left(\frac{1}{20}, \frac{9}{20}\right)$

Line 27: $j = 2, i = 1$, $Temp_1 = \left(\frac{1}{40}, \frac{49}{40}\right)$

Line 27: $j = 2, i = 3$, $Temp_3 = \left(\frac{1}{40}, \frac{9}{40}\right)$

Line 27: $j = 2, i = 4$, $Temp_4 = \left(\frac{1}{40}, \frac{9}{40}\right)$

Line 27: $j = 2, i = 5$, $Temp_5 = \left(\frac{1}{40}, \frac{9}{40}\right)$

Line 27: $j = 3, i = 2$, $Temp_2 = \left(\frac{1}{20}, \frac{29}{20}\right)$

Line 27: $j = 4, i = 2$, $Temp_2 = \left(\frac{21}{20}, \frac{29}{20}\right)$

Line 27: $j = 5, i = 2$, $Temp_2 = \left(\frac{26}{20}, \frac{34}{20}\right)$

Line 27: $j = 5, i = 6$, $Temp_6 = \left(\frac{1}{4}, \frac{1}{4}\right)$

Line 27: $j = 6, i = 5$, $Temp_5 = \left(\frac{23}{120}, \frac{47}{120}\right)$

Line 27: $j = 6, i = 7$, $Temp_7 = \left(\frac{1}{6}, \frac{1}{6}\right)$

Line 27: $j = 6, i = 8$, $Temp_8 = \left(\frac{1}{6}, \frac{1}{6}\right)$

Line 27: $j = 7, i = 6$, $Temp_6 = \left(\frac{1}{4}, \frac{5}{4}\right)$

Line 27: $j = 8, i = 6$, $Temp_6 = \left(\frac{5}{4}, \frac{5}{4}\right)$

From the examples, which were discussed above, it can be seen that both interpretations (Algorithm 1 and Algorithm 2) are equivalent in terms of changes in the matrix Y. If we take a look at the last "Apply" phase and 2.8 after row-normalization from Example 2.1.1, we see that after normalization in the "Apply" phase we get the exactly the same results which are presented in Y. We have already discussed the results of the "Scatter" phase, which are the same as the ones after propagation. Therefore, it can be concluded that both interpretations give us the same results.

2.2 Grappa

Grappa is a modern view of distributed shared memory system for in-memory data-intensive applications [10]. It exploits concurrency in big data applications to cover network access latencies, enabling users to program a cluster as if it were a single, large shared memory machine. Grappa does so using three main components: a *user-level multitasking system* supporting thousands of concurrent threads per core in the system, a *communication layer* that combines independent messages with common destinations to increase network efficiency, and a *distributed shared memory (DSM)* layer that provides access to data anywhere in the system. Remote memory is accessed using *delegate operations*, which execute a small piece of code at the core where a value is stored; this allows efficient fine-grained memory operations, including both simple reads and writes and more complex synchronization operations. This allows Grappa's DSM to provide the same memory consistency model as is used in C/C++. Grappa is implemented as a C++ user library using MPI [7] for communication. Grappa's efficient shared memory programming model allows different frameworks to co-exist in the same application and to develop application-specific optimization. As it

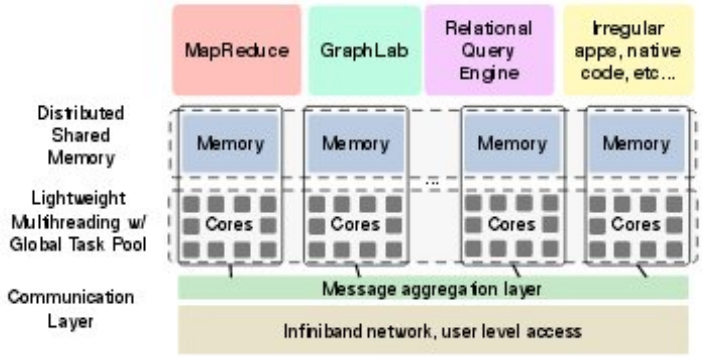


Figure 2.2: Grappa design overview (picture from [10])

was mentioned by the authors, a vertex-centric GraphLab-like API on Grappa performs better than GraphLab itself. For more details about the architecture of Grappa, please see Fig. 2.2

To take a deeper look into the Grappa's vertex-centric API we discuss main implementation details of LPA. A vertex-centric GraphLab-like API on Grappa gives us possibility to define our own structure for each vertex and add data, which we want to be included in the vertex. To implement the LPA we define a structure, which includes an array (*label_count*) of size n. It represents the label distribution for a vertex. The crucial role during propagation is played by the "Addition assignment" $(+=)$ operator. Grappa's vertex-centric API use it to collect the results from neighbors. So, we need to overload it in a desired way. More details about its implementation can be found in Fig. 2.3.

According to Algorithm 2, when information is collected we normalize a vector (array in the implementation) for each vertex and then find the most probable label for the node during the "Apply" phase. For implementation details please see Fig. 2.4.

As we can see from Fig. 2.4, the implementation of the "Apply" mostly corresponds Algorithm 2. The only exception is the "activate()" function which is used for activating of the node which can not meet the criterion we use for stopping. If none of the vertices become active during the "Apply", the iterative process stops. Otherwise all nodes are

```
label_counter& operator+=(const label_counter& other) {

  CHECK(other.nadj > 0) << "Neighbor should have at least one other neighbor.";


  for (int i = 0; i < number_of_groups; i++) {
    label_count[i] += (other.label_count[i])/other.nadj;
  }

  return *this;
}
```

Figure 2.3: Snapshot of LPA implementation in Grappa, which illustrates how information is collected from each neighbor during "Gather" and "Scatter" phases.

```
void apply(Vertex& v,  Gather& total) {

   if(v->check_seed){
           for(int i=0; i<number_of_groups; i++){
               total.label_count[i]=0.0;
           }
           changelabel.nadj=v.nadj;
           changelabel.label_count[v->label]=1.0;
       return;
   }
  double maxCount = 0.0;
  label_counter tmp;
  for(int i=0; i<number_of_groups; i++){
      tmp.label_count[i]=changelabel.label_count[i];
      changelabel.label_count[i]=total.label_count[i];
      total.label_count[i]=0;
  }
  changelabel.normalize();
  double norm=0;
  for(int i=0; i<number_of_groups; i++){
      norm+=(tmp.label_count[i]-changelabel.label_count[i])*(tmp.label_count[i]-changelabel.label_count[i]);
  }
  if(sqrt(norm)>=threshold) v->activate();
  int maxLabel = v->label;
  changelabel.vId=v->vid;
  for (int i = 0; i < number_of_groups; i++) {
      if (changelabel.label_count[i] > maxCount) {
        maxCount = changelabel.label_count[i];
        maxLabel = i;
      }
  }
  if (maxLabel != v->label){
      if(maxCount>0){
             v->label = maxLabel;
      }
  }
}
```

Figure 2.4: Snapshot of LPA implementation in Grappa, which illustrates the "Apply" phase from the GraphLab's GAS engine

activated for next iteration. This state allows the vertices to propagate their information towards neighbors.

Taking into account the facts mentioned above about the “Addition assignement” operator and about the “activate()” function, we can conclude that both of them play important role for the implementation, if one uses Grappa’s GraphLab-like vertex-centric API.

Chapter 3

SCALABLE LABEL PROPAGATION ALGORITHMS FOR BIPARTITE GRAPHS

3.1 Introduction

The original LPA by Zhu and Ghahramani [19] assumes that all vertices are of the same kind. Bipartite graphs with two kinds of vertices arise naturally in many cases. A nice example is Facebook, where we have users and pages as nodes, while edges correspond to "likes"(i.e. the user is a fan of the page or not). Considering the vertices to be the same type might lead us to losing some information. Taking into account this fact, previously discussed algorithms may not be effective. Therefore, we propose novel label propagation approaches for bipartite graphs, which consist of the combined application of two algorithms: LPA and Adaptive-LPA.

Let us assume that we have a bipartite graph (i.e. a graph with two types of vertices). Consider the undirected graph $G = (V, E)$, where $V = V_1 \cup V_2$ is the set of all nodes of different types and E is the set of edges (connections between different types of nodes). Let us assume that we are given seed nodes from V_1, $X_L = \{x_1, x_2, \ldots, x_l\}$ and their corresponding labels $Y_L = \{y_1, y_2, \ldots, y_l\}$ to infer the labels $Y_U = \{y_{l+1}, y_{l+2}, \ldots, y_{l+u}\}$ of previously unlabeled vertices from V_1, $X_U = \{x_{l+1}, x_{l+2}, \ldots, x_{l+u}\}$. The labels are drawn from a predefined set $C = \{c_1, c_2, \ldots, c_n\}$.

Section 3.2 will cover the description of *user-item LPA with item-user Adaptive-LPA* with a small example, criteria we used to stop iterative process and details about our scalable implementation of the algorithm. We will discuss *user-item AdaptiveLPA with item-user LPA* with its description, stopping criteria and implementation details in Section 3.3.

3.2 User-item LPA with Item-user Adaptive-LPA

3.2.1 Algorithm Description

Let us assume that we have the $(l + u) \times n$ matrix Y, where the i-th row Y_i represents the label distribution for vertex x_i, and the element Y_{ik} represents the probability that the correct label for vertex x_i is c_k. Let us now define a variable $type_j$ for each node x_j as follows:

$$type_j = \begin{cases} 1, & \text{if label is needed for } x_j \\ 2, & \text{otherwise} \end{cases}$$

When type equals 1, it means that we are going to label this node using a predefined set of possible labels C or the node already has a label, because it is a seed node, while vertices of type 2 are the intermediaries between the vertices of type 1. Information may be given about the labels of vertices of type 2, but not necessarily. We use them during the propagation process to label the nodes of type 1. To be more explicit, we can take a look at a small bipartite graph in Figure 3.1, where seed nodes are the vertices of type 1. Our goal is to label nodes 2, 5 and 8, which are also included in type 1 vertices. Other nodes (3,6 and 9) represent intermediaries between type 1 vertices and $type_j$ variable for each of them equald to 2.

Let us consider that we are not given any information about the nodes of type 2. Let us assume that for each vertex x_j, a vector of size n, Y_j is the current label distribution for the given vertex. The element Y_{jk} represents the probability that the correct label for vertex x_j is c_k.

Next we define a relation matrix for bipartite graphs G where we use connections between just one type of nodes through intermediaries. By “conection” we mean a simple path of

length 2 in G. Let P be a $n \times n$ matrix:

$$P = \begin{bmatrix} p_{c_1|c_1} & p_{c_1|c_2} & \cdots & p_{c_1|c_n} \\ p_{c_2|c_1} & p_{c_2|c_2} & \cdots & p_{c_2|c_n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{c_n|c_1} & p_{c_n|c_2} & \cdots & p_{c_n|c_n} \end{bmatrix}$$

Each element $p_{c_k|c_{k'}}$ is a number between 0 and 1 that is particular to denote the relation between classes c_k and $c_{k'}$. The value of $p_{c_k|c_{k'}}$ is calculated using connections between seed nodes of type 1 through the vertices of type 2. First we pick the connections where one node is from class c_k, while another one is from class $c_{k'}$ and then we divide their quantity by the number of connections which nodes from class $c_{k'}$ have with all other seed nodes through intermediaries. More precisely $p_{c_k|c_{k'}}$ is defined as follows:

$$p_{c_k|c_{k'}} = \frac{\sum\limits_{\substack{(x_r,x_i)\in E' \\ (x_r,x_j)\in E'}} Y_{ik} \cdot Y_{jk'}}{\sum\limits_{\substack{(x_r,x_j)\in E' \\ (x_r,x_i)\in E'}} Y_{jk'}} \tag{3.1}$$

where E' is the set of all edges for which one node belongs to the seed nodes. x_r denotes a vertex of type 2; it plays a crucial role in connecting the nodes of type 1. As we include just seed nodes and connections between them through intermediaries, we still assume both x_i and x_j in the denominator. Otherwise we would sum over all the edges between seed nodes from class $c_{k'}$ with vertices of type 2, which will lead us to losing information on connections between seed nodes.

The algorithm itself is an iterative approach, where propagation occurs from every node during each iteration. Considering that we have a bipartite graph, which includes different types of nodes, we leverage these properties and use different propagation strategies for different types of nodes. In particular, we use LPA to propagate information from nodes of type 1, while Adaptive-LPA is applied for propagation from the nodes of type 2. It means

that we propagate the label distribution from type 1 nodes to intermediaries in the traditional way, while type 2 nodes propagate the label distribution based on a relation matrix that is learned beforehand from connections between seed nodes. As we have connections only between different types of nodes, the propagation strategies can be interpreted as the information a node receive from its neighbors. In this case if a recipient node is a type 1 node, it collects information from neighbors using Adaptive-LPA, while a vertex of type 2 uses LPA to gather information from neighbors. Taking into account these facts we will get the following formulation:

$$Y_{ik} = \begin{cases} \sum\limits_{j \in N(i)} \frac{1}{|N(j)|} Y_{jk}, & \text{if } type_i = 2 \\ \sum\limits_{j \in N(i)} \sum\limits_{c_{k'} \in C} \frac{P_{kk'}}{|N(j)|} Y_{jk'}, & \text{if } type_i = 1 \end{cases} \tag{3.2}$$

where $N(i)$ is the set of neighbors of the node x_i.

As the result of LPA is based on the matrix Y, we need to initialize it at the first step of the algorithm, using the information we have about the seed nodes. Recall that Y_{ik} is the probability that the label of node x_i is c_k. We know that the label of seed node x_i is y_i, for $i = 1, \ldots, l$, so we initialize Y_ik to 1 if $y_i = c_k$, and 0 otherwise. As we do not have any label for other nodes, we do not want them to propagate anything, thus we assign zeroes to all the elements for their corresponding rows in matrix Y.

As we are given labels for seed nodes, we should not change them during propagation. Therefore to maintain consistency of the algorithm we consider 2 different clamping methods:

- Seed-Clamping: In this approach, we clamp the seed nodes as proposed by [19], which means that the labels of seed nodes never change. The algorithm converges if no changes occur in the whole graph.

- Iteration-Clamping: In this approach, in addition to clamping the seed nodes, we clamp other nodes as well if their labels do not change after i consecutive iterations

in a row. When all the nodes are clamped, the iterations are stopped.

We present details and a full description with seed-clamping in Algorithm 3.

Algorithm 3 User-item LPA with Item-user Adaptive-LPA with Seed-Clamping

```
for i = 1, 2, ..., l + u do
    for k = 1, 2, ..., n do
        if y_i = c_k then
            Y_ik ← 1
        else
            Y_ik ← 0
        end if
    end for
end for
Initialize matrix P using Equation (3.1)
Collect information from every node with appropriate strategy using Equation (3.2)
Row-normalize Y
Reset the rows of the seed nodes in Y
Repeat from step 3 until convergence
y_j ← arg max_{c_k ∈ C} Y_jk, j = l + 1, l + 2, ..., l + u
```

We will calculate a prospective label for each node during every iteration except for the seed nodes. If y_j is the same during i consecutive iterations we will clamp it. More details of the algorithm with Iteration-Clamping approach are presented in Algorithm 4.

To take a deeper look into the details of Algorithm 3, we apply it to a small example graph, which is presented in Figure 3.1. This gives us the possibility to discuss all the changes which can happen during the first 2 iterations of the algorithm.

Example 3.2.1 *Let us assume that we are given the graph from Fig. 3.1 with seed nodes* $X_L = \{0, 4, 7, 10\}$*, their corresponding labels* $Y_L = \{y_0 =$ *"female",* $y_4 =$ *"male",* $y_7 =$ *"female",* $y_{10} =$ *"male"*$\}$ *and a predefined set of labels* $C = \{c_0 =$ *"male",* $c_1 =$ *"female"*$\}$*.*

Let us consider 2 iterations of Algorithm 3. When we apply Algorithm 3 to the graph with the initial data described above, we get the following changes:

Steps 1-9 *The* Y *matrix has 10 rows, where initially the elements are only 0 or 1. The first column is for "male" while the second one corresponds to "female". As we know*

Algorithm 4 User-item LPA with Item-user Adaptive-LPA with Iteration-Clamping

for $i = 1, 2, \ldots, l + u$ **do**
  for $k = 1, 2, \ldots, n$ **do**
    if $y_i = c_k$ **then**
      $Y_{ik} \leftarrow 1$
    else
      $Y_{ik} \leftarrow 0$
    end if
  end for
end for
Initialize matrix P using Equation (3.1)
Propagate from every node to every non-clamped neighbor with appropriate propagation strategy using Equation (3.2)
Row-normalize Y
Reset the rows of the seed and clamped nodes in Y
$y_j \leftarrow \arg\max_{c_k \in C} Y_{jk}$, $j = l + 1, l + 2, \ldots, l + u$
Clamp a node x_j *if* y_j has not changed during i iterations, $j = l + 1, l + 2, \ldots, l + u$
Repeat from step 3 until convergence

labels for vertices 0,4,7 and 10, their corresponding rows are filled with 0 and 1 in the appropriate places. For example, a label for node 0 is "female", thus in row 0 the first element is 0 and the second one is 1.

$$Y = \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} \tag{3.3}$$

Step 10 *Initializing matrix P is based on the connections that the seed nodes have. In our example, the set of edges including seed nodes is $E' = \{(0,1),(0,3),(4,3),(4,6),(7,1),(7,3),(7,6),(7,9),(10,6),(10,9)\}$. We discuss how to calculate P_{00}, which is $p_{\text{“}male\text{”}|\text{“}male\text{”}}$. Using Equation 3.1 will lead us to calculate connections between “male” users through intermediaries. According to the example we have two nodes, namely 4 and 10, which are labeled as “male”. As they are connected with each other through vertex 6, it means that each of them has 1 connection, and the total number of connections becomes 2. Now we divide it by the total number of connections all “male” nodes have with seed nodes of any class. Vertex 4 has two connections (0 and 7) through node 3 and two other connections (7 and 10) through vertex 6, while vertex 10 has two connections (4 and 7) through node 6 and one more connection with vertex 7 through node 9. Hence we have 7 connections in total and $P_{00} = \frac{2}{7}$. Using the same calculations for other relations, the elements of matrix P are:*

$$P = \begin{bmatrix} \frac{2}{7} & \frac{5}{9} \\ \frac{5}{7} & \frac{4}{9} \end{bmatrix}$$

Iteration 1, Step 11 *Collecting information using the appropriate propagation strategy. Let us discuss both strategies for different nodes. We can select vertices 0, 1 and 2. First we examine how propagation works for vertex 1. As it is a node of type 2 we use LPA for propagation from its neighbors. It has just 2 seed nodes as neighbors: 0 and 7. Others will just propagate 0's, so they do not have any influence. We check row 0 in (3.3) and row 7 in matrix Y and divide their elements by the corresponding number of neighbors. As vertex 0 has 2 neighbors we get $\frac{1}{2}$ for the element corresponding to “female”, while “male” one is 0. Taking into account that the number of neighbors for vertex 7 is 4, we get $\frac{1}{4}$ for “female”, and 0 for “male”. When we sum $\frac{1}{2}$ and $\frac{1}{4}$, the result is $\frac{3}{4}$ for “female”, while we get 0 as a result for “male” for row 1 in (3.4).*

As none of the nodes of type 2 has any label, their corresponding rows in matrix Y in

(3.3) are filled with 0s. When we use Adaptive-LPA for propagation from them, all elements in the sum become 0. As both vertices (0 and 2) are nodes of type 1, they collect information from only type 2 vertices. Hence the elements of the corresponding rows in Y become 0 in (3.4).

$$Y = \begin{bmatrix} 0 & 0 \\ 0 & \frac{3}{4} \\ 0 & 0 \\ \frac{1}{2} & \frac{3}{4} \\ 0 & 0 \\ 0 & 0 \\ 1 & \frac{1}{4} \\ 0 & 0 \\ 0 & 0 \\ \frac{1}{2} & \frac{1}{4} \\ 0 & 0 \end{bmatrix} \tag{3.4}$$

Iteration 1, Step 12 *Row-normalization:*

$$Y = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ \frac{2}{5} & \frac{3}{5} \\ 0 & 0 \\ 0 & 0 \\ \frac{4}{5} & \frac{1}{5} \\ 0 & 0 \\ 0 & 0 \\ \frac{2}{3} & \frac{1}{3} \\ 0 & 0 \end{bmatrix} \tag{3.5}$$

Iteration 1, Step 13 *Reset the rows of seed nodes:*

$$Y = \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 0 & 0 \\ \frac{2}{5} & \frac{3}{5} \\ 1 & 0 \\ 0 & 0 \\ \frac{4}{5} & \frac{1}{5} \\ 0 & 1 \\ 0 & 0 \\ \frac{2}{3} & \frac{1}{3} \\ 1 & 0 \end{bmatrix}$$

Iteration 2, Step 11 *Collecting information using appropriate strategy:*

$$Y = \begin{bmatrix} \frac{316}{945} & \frac{314}{945} \\ 0 & \frac{3}{4} \\ \frac{5}{27} & \frac{4}{27} \\ \frac{1}{2} & \frac{3}{4} \\ \frac{114}{525} & \frac{166}{525} \\ \frac{107}{1575} & \frac{208}{1575} \\ 1 & \frac{1}{4} \\ \frac{1859}{3150} & \frac{2446}{3150} \\ \frac{107}{1575} & \frac{208}{1575} \\ \frac{1}{2} & \frac{1}{4} \\ \frac{71}{378} & \frac{118}{378} \end{bmatrix} \tag{3.6}$$

Iteration 2, Step 12 *Row-normalization:*

$$Y = \begin{bmatrix} \frac{158}{315} & \frac{157}{315} \\ 0 & 1 \\ \frac{5}{9} & \frac{4}{9} \\ \frac{2}{5} & \frac{3}{5} \\ \frac{57}{140} & \frac{83}{140} \\ \frac{107}{315} & \frac{208}{315} \\ \frac{4}{5} & \frac{1}{5} \\ \frac{1859}{4305} & \frac{2446}{4305} \\ \frac{107}{315} & \frac{208}{315} \\ \frac{2}{3} & \frac{1}{3} \\ \frac{71}{189} & \frac{118}{189} \end{bmatrix}$$

Iteration 2, Step 13 *Reset the rows of the seed nodes:*

$$Y = \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ \frac{5}{9} & \frac{4}{9} \\ \frac{2}{5} & \frac{3}{5} \\ 1 & 0 \\ \frac{107}{315} & \frac{208}{315} \\ \frac{4}{5} & \frac{1}{5} \\ 0 & 1 \\ \frac{107}{315} & \frac{208}{315} \\ \frac{2}{3} & \frac{1}{3} \\ 1 & 0 \end{bmatrix}$$

According to the stopping criterion which is used during the implementation, iterative process stops after the third iteration. Y matrix takes the following form:

$$Y = \begin{bmatrix} 0 & 1 \\ 0.31746 & 0.68254 \\ 0.555556 & 0.444444 \\ 0.4 & 0.6 \\ 1 & 0 \\ 0.339683 & 0.660317 \\ 0.516728 & 0.483272 \\ 0 & 1 \\ 0.339683 & 0.660317 \\ 0.666667 & 0.333333 \\ 1 & 0 \end{bmatrix} \tag{3.7}$$

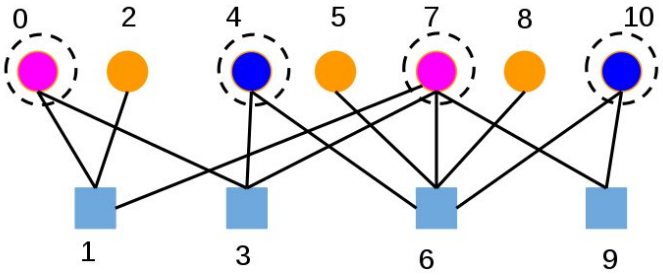


Figure 3.1: Bipartite example graph: circles represent the vertices of type 1 and squares denote the vertices of type 2 (i.e. the intermediaries between the vertices of type 1). Circled nodes correspond to seed nodes.

We now can go to Step 15.

Step 15 *Let us discuss details of choosing a label for vertex 1. As the second element of row 1 in Y in (3.7) is greater than the first one of the same row, $y_1 =$ "female". The labels for other vertices are inferred in the same way. Therefore, $y_2 =$ "male", $y_3 =$ "female", $y_5 =$ "female", $y_6 =$ "male", $y_8 =$ "female" and $y_9 =$ "male".*

3.2.2 Stopping Criteria

As Algorithm 3 is based on an iterative approach, without any stopping criterion, termination will never happen. To avoid an infinite loop, we propose an approach which is based on the labels of the nodes. In particular, we stop the process when none of the nodes changes its most probable label during the iteration. An important part in this stopping criterion is played by the clamping strategy. If the algorithm allows every node to keep changing its label, termination will never happen. Thus we used clamping approaches which were discussed in Section 3.2.1.

3.2.3 Scalable Implementation

We implemented Algorithm 3 using Grappa's vertex-centric API. This API implements a subset of the techniques developed in the GraphLab project [9] to support iterative, vertex-centric computation over graphs. The main concept in GraphLab lies in the Gather, Apply, Scatter (GAS) model, which describes the computation performed at each vertex in the graph using data from neighboring vertices as well as its own one. Unlike in the Graphlab version, in Grappa the "Gather" operation is completed only once at the very first step. After that, an iterative process includes only an interchange of "Apply" and "Scatter"' operations. As a vertex-centric model provides gradual accumulation of the information from neighbors, we need to have temporary memory for each vertex where information can be gathered. Using the temporary memory also ensures that a vertex will not lose the information, which should be propagated during the iteration. For temporary memory we use a vector for each node. To gather information from neighbors in a proper way, we assign 0 to all of its elements (Algorithm 5, line 3) when "Apply" is almost done. During the "Apply" phase, we do normalization and then choose the most probable label for a vertex. The "Scatter" phase provides an implementation of information propagation from a vertex to its neighbors. The propagation strategy is based on the type of node from where the Scatter occurs. More details for an iteration of Algorithm 3 in Grappa-like pseudocode can be seen in Algorithm 5. When we use the notation $Temp_j$ with just one subscript in Algorithm 5, we apply vector operations, while using two subscripts refers to individual element of the vector.

Grappa's Graphlab implementation, gives us the possibility to use the activate() function for a node to make it active. We activate only the nodes which change labels during the "Apply" phase. Grappa's vertex-centric API ensures to activate all the nodes if we have at least one active node. This makes all the nodes propagate their information. Otherwise, only the nodes which became active during the "Apply" phase are able to propagate their data. If we do not activate anything during "Apply", none of them become active and

Grappa's API stops the iterative process.

Algorithm 5 An Iteration of User-item LPA with Item-user Adaptive-LPA

```
1: for each vertex j do
2:     Apply:
3:         if vertex j is not a seed node then
4:             Y_j ← Temp_j
5:             Normalize Y_j
6:             y_j ← arg max_{c_k ∈ C} Y_jk
7:         end if
8:         Temp_j ← 0⃗
9: end for
10: for each vertex j do
11:     Scatter:
12:         for i ∈ Neighbors(j) do
13:             if type_j = 1 then
14:                 Temp_i ← Temp_i + Y_j / n_j
15:             else
16:                 for s = 1, 2, ..., n do
17:                     for k = 1, 2, ..., n do
18:                         Temp_is ← Temp_is + P_sk Y_jk / n_j
19:                     end for
20:                 end for
21:             end if
22:         end for
23:     end for
```

In Algorithm 5, line 3 refers to Seed-Clamping. If a node is a seed node we change neither its label nor the corresponding row in matrix Y. For the Iteration-Clamping approach we added two attributes for each node. One of them calculates the number of iterations during which the label does not change for a vertex. If the desired number of consecutive iterations is reached, another attribute, which checks if a node is already clamped, is made to become true. Using the latter attribute during "Apply" gives us the possibility to decide if we should change label for node or not.

To illustrate the equivalence of Algorithm 3 and Algorithm 5 we walk the reader through one iteration of the same example that was discussed in Section 3.2.1.

Example 3.2.2 *Let us consider that we are given the same graph and initial data as in Example 3.2.1. When we go to the first "Apply" phase, "Gather" is already done and the information for every node is already accumulated from neighbor vertices in* $Temp_j$ *vectors. Thus we have:* $Temp_0 = (0,0)$, $Temp_1 = \left(0, \frac{3}{4}\right)$, $Temp_2 = (0,0)$, $Temp_3 = \left(\frac{1}{2}, \frac{3}{4}\right)$, $Temp_4 = (0,0)$, $Temp_5 = (0,0)$, $Temp_6 = \left(1, \frac{1}{4}\right)$, $Temp_7 = (0,0)$, $Temp_8 = (0,0)$, $Temp_9 = \left(\frac{1}{2}, \frac{1}{4}\right)$, $Temp_10 = (0,0)$. *We will discuss the steps from Algorithm 5 and changes that occur by their influence:*

Line 3 *Apply:* $j = 0$. *Vertex 0 is a seed node and we are going to Line 7.*

Line 7 $Temp_0 = (0,0)$.

Line 3 *Apply:* $j = 1$. *Vertex 1 is not a seed node and we continue to Line 4.*

Line 4 $Y_1 = \left(0, \frac{3}{4}\right)$

Line 5 *Normalization:* $Y_1 = (0,1)$

Line 6 $y_1 =$ *"female"*

Line 7 $Temp_1 = (0,0)$

Line 3 *Apply:* $j = 2$. *Vertex 2 is not a seed node.*

Line 4 $Y_2 = (0,0)$

Line 5 *Normalization:* $Y_2 = (0,0)$

Line 6 $y_2 =$ *"undecided"*

Line 7 $Temp_2 = (0,0)$

Line 3 *Apply:* $j = 3$. *Vertex 3 is not a seed node.*

Line 4 $Y_3 = \left(\frac{1}{2}, \frac{3}{4}\right)$

Line 5 *Normalization:* $Y_3 = \left(\frac{2}{5}, \frac{3}{5}\right)$

Line 6 $y_3 =$ *"female"*

Line 7 $Temp_3 = (0, 0)$

Line 3 *Apply:* $j = 4$. *Vertex 4 is a seed node.*

Line 7 $Temp_4 = (0, 0)$

Line 3 *Apply:* $j = 5$. *Vertex 5 is not a seed node.*

Line 4 $Y_5 = (0, 0)$

Line 5 *Normalization:* $Y_5 = (0, 0)$

Line 6 $y_5 =$ *"undecided"*

Line 7 $Temp_5 = (0, 0)$

Line 3 *Apply:* $j = 6$. *Vertex 6 is not a seed node.*

Line 4 $Y_6 = \left(6, \frac{1}{4}\right)$

Line 5 *Normalization:* $Y_6 = \left(\frac{4}{5}, \frac{1}{5}\right)$

Line 6 $y_6 =$ *"male"*

Line 7 $Temp_6 = (0, 0)$

Line 3 *Apply:* $j = 7$. *Vertex 7 is a seed node.*

Line 7 $Temp_7 = (0, 0)$

Line 3 *Apply:* $j = 8$. *Vertex 8 is not a seed node.*

Line 4 $Y_8 = (0, 0)$

Line 5 *Normalization:* $Y_8 = (0, 0)$

Line 6 $y_8 =$ *"undecided"*

Line 7 $Temp_8 = (0, 0)$

Line 3 *Apply:* $j = 9$. *Vertex 9 is not a seed node.*

Line 4 $Y_9 = \left(\frac{1}{2}, \frac{1}{4}\right)$

Line 5 *Normalization:* $Y_9 = \left(\frac{2}{3}, \frac{1}{3}\right)$

Line 6 $y_9 =$ *"female"*

Line 7 $Temp_9 = (0, 0)$

Line 3 *Apply:* $j = 10$. *Vertex 10 is a seed node.*

Line 7 $Temp_3 = (0, 0)$

The "Apply" phase is already over and we are going now into the "Scatter" phase.

Line 13 $j = 0$. *$type_0 = 1$. We go to Step 14.*

Line 14 $j = 0$, $i = 1$*: As* $Y_0 = (0, 1)$ *and the number of neighbors of vertex 0 is 2,* $\left(0, \frac{1}{2}\right)$ *is added to* $Temp_1$*. Thus we get* $Temp_1 = \left(0, \frac{1}{2}\right)$*. The same will happen to vertex 3.*

Line 14 $j = 0$, $i = 3$*:* $Temp_3 = \left(0, \frac{1}{2}\right)$

Line 13 $j = 1$. *$type_1 = 2$. We go to Line 16.*

Line 16 $s = 1$.

Line 17 $k = 1$.

Line 18 $j = 1$, $i = 0$, $s = 1$, $k = 1$*: Considering that* $Y_1 = (0, 1)$ *and using the same relation matrix* P *which was used in the example from Section 3.2.1, we get* $Temp_{01} = 0 + \frac{2}{7} \cdot 0/3$*. We get that* $Temp_{01} = 0$

Line 18 $j = 1$, $i = 0$, $s = 1$, $k = 2$*: we get* $Temp_{01} = 0 + \frac{5}{9} \cdot 1/3$*. We get that* $Temp_{01} = \frac{5}{27}$*. We will do the same for* $Temp_{02}$.

Step 19 *When we are at this line, we have* $Temp_{02} = \frac{4}{27}$*. We will not go into a detailed explanation for the other nodes, but we will present important results.*

Line 13 $j = 1$. *$type_1 = 2$. We go to Line 16.*

Lines 16-20 $j = 1$, $i = 2$*. Result is:* $Temp_2 = \left(\frac{5}{27}, \frac{4}{27}\right)$

Line 13 $j = 1$. *$type_1 = 2$. We go to Line 16.*

Lines 16-20 $j = 1$, $i = 7$*. Result is:* $Temp_7 = \left(\frac{5}{27}, \frac{4}{27}\right)$

Line 13 $j = 2$. $type_3 = 1$. *We go to Line 14.*

Line 14 $j = 2$, $i = 1$. *Result is:* $Temp_1 = \left(0, \frac{1}{2}\right)$

Line 13 $j = 3$. $type_3 = 2$. *We go to Line 16.*

Lines 16-20 $j = 3$, $i = 0$. *Result is:* $Temp_0 = \left(\frac{316}{945}, \frac{314}{945}\right)$

Line 13 $j = 3$. $type_3 = 2$. *We go to Line 16.*

Lines 16-20 $j = 3$, $i = 4$. *Result is:* $Temp_4 = \left(\frac{47}{315}, \frac{58}{315}\right)$

Line 13 $j = 3$. $type_3 = 2$. *We go to Line 16.*

Lines 16-20 $j = 3$, $i = 7$. *Result is:* $Temp_7 = \left(\frac{316}{945}, \frac{314}{945}\right)$

Line 13 $j = 4$. $type_4 = 1$. *We go to Line 14.*

Line 14 $j = 4$, $i = 3$. *Result is:* $Temp_3 = \left(\frac{1}{2}, \frac{1}{2}\right)$

Line 13 $j = 4$. $type_4 = 1$. *We go to Line 14.*

Line 14 $j = 4$, $i = 6$. *Result is:* $Temp_6 = \left(\frac{1}{2}, 0\right)$

Line 13 $j = 5$. $type_5 = 1$. *We go to Line 14.*

Line 14 $j = 5$, $i = 6$. *Result is:* $Temp_6 = \left(\frac{1}{2}, 0\right)$

Line 13 $j = 6$. $type_6 = 2$. *We go to Line 16.*

Lines 16-20 $j = 6$, $i = 4$. *Result is:* $Temp_4 = \left(\frac{114}{525}, \frac{166}{525}\right)$

Line 13 $j = 6$. $type_6 = 2$. *We go to Line 16.*

Lines 16-20 $j = 6$, $i = 5$. *Result is:* $Temp_5 = \left(\frac{107}{1575}, \frac{208}{1575}\right)$

Line 13 $j = 6$. $type_6 = 2$. *We go to Line 16.*

Lines 16-20 $j = 6$, $i = 7$. *Result is:* $Temp_7 = \left(\frac{1901}{4725}, \frac{2194}{4725}\right)$

Line 13 $j = 6$. $type_6 = 2$. *We go to Line 16.*

Lines 16-20 $j = 6$, $i = 8$. *Result is:* $Temp_8 = \left(\frac{107}{1575}, \frac{208}{1575}\right)$

Line 13 $j = 6$. $type_6 = 2$. *We go to Line 16.*

Lines 16-20 $j = 6$, $i = 10$. *Result is:* $Temp_10 = \left(\frac{107}{1575}, \frac{208}{1575}\right)$

Line 13 $j = 7$. $type_7 = 1$. *We go to Line 14.*

Line 14 $j = 7$, $i = 1$. *Result is:* $Temp_1 = \left(0, \frac{3}{4}\right)$

Line 13 $j = 7$. $type_7 = 1$. *We go to Line 14.*

Line 14 $j = 7$, $i = 3$. *Result is:* $Temp_3 = \left(\frac{1}{2}, \frac{3}{4}\right)$

Line 13 $j = 7$. $type_7 = 1$. *We go to Line 14.*

Line 14 $j = 7$, $i = 6$. *Result is:* $Temp_6 = \left(\frac{1}{2}, \frac{1}{4}\right)$

Line 13 $j = 7$. $type_7 = 1$. *We go to Line 14.*

Line 14 $j = 7$, $i = 9$. *Result is:* $Temp_9 = \left(0, \frac{1}{4}\right)$

Line 13 $j = 8$. $type_8 = 1$. *We go to Line 14.*

Line 14 $j = 8$, $i = 6$. *Result is:* $Temp_6 = \left(\frac{1}{2}, \frac{1}{4}\right)$

Line 13 $j = 9$. $type_9 = 2$. *We go to Line 16.*

Lines 16-20 $j = 9$, $i = 7$. *Result is:* $Temp_7 = \left(\frac{1859}{3150}, \frac{2446}{3150}\right)$

Line 13 $j = 9$. $type_9 = 2$. *We go to Line 16.*

Lines 16-20 $j = 9$, $i = 10$. *Result is:* $Temp_{10} = \left(\frac{71}{378}, \frac{118}{378}\right)$

Line 13 $j = 10$. $type_{10} = 1$. *We go to Line 14.*

Line 14 $j = 10$, $i = 6$. *Result is:* $Temp_6 = \left(1, \frac{1}{4}\right)$

Line 13 $j = 10$. $type_{10} = 1$. *We go to Line 14.*

Line 14 $j = 10$, $i = 9$. *Result is:* $Temp_9 = \left(\frac{1}{2}, \frac{1}{4}\right)$

If we take a look at the values of Y in the "Apply" phase in Example 3.2.2 after normalization we discover that they are exactly the same as in Y in (3.5). The results of the "Scatter" phase collected in $Temp$ vectors are equivalent to the values in the matrix Y in (3.6). Taking into account these results we can conclude that our implementation computes the same thing as in Example 3.2.1.

3.3 User-item Adaptive-LPA with Item-user LPA

3.3.1 Algorithm Description

The main difference between this approach and the algorithm which is described in Section 3.2.1 is the propagation strategy from different types of nodes. While Algorithm 3 uses LPA for propagation from the vertices of type 1 and Adaptive-LPA from the vertices of type 2, the method described in this subsection uses Adaptive-LPA for scattering the information from the nodes of type 1 and LPA from type 2 vertices. The algorithm description itself

stays almost the same, except Step 3, where we should change the equation which is used for collecting information from neighbors. Taking into account the direction of propagation, we modify Equation (3.2) and get the new updated equation as follows:

$$Y_{ik} = \begin{cases} \sum_{j \in N(i)} \frac{1}{|N(j)|} Y_{jk}, & \text{if } type_i = 1 \\ \sum_{j \in N(i)} \sum_{c_{k'} \in C} \frac{P_{kk'}}{|N(j)|} Y_{jk'}, & \text{if } type_i = 2 \end{cases} \tag{3.8}$$

Step 3 from Algorithm 3 will take the new form: Propagate from every node with appropriate propagation strategy using Equation (3.8). The whole description and more details can be found in Algorithm 6.

Algorithm 6 User-item Adaptive-LPA with Item-user LPA with Seed-Clamping

```
for i = 1, 2, ..., l + u do
    for k = 1, 2, ..., n do
        if y_i = c_k then
            Y_ik ← 1
        else
            Y_ik ← 0
        end if
    end for
end for
Initialize matrix P using Equation (3.1)
Collect information from every node with appropriate strategy using Equation (3.8)
Row-normalize Y
Reset the rows of the seed nodes in Y
Repeat from step 3 until convergence
y_j ← arg max_{c_k ∈ C} Y_jk, j = l + 1, l + 2, ..., l + u
```

To understand the difference in details between the two algorithms, we apply Algorithm 6 to the same example which is provided in section 3.2.1.

Example 3.3.1 *Let us assume that we are given the same graph (see Fig. 3.1) with the same seed nodes, their corresponding labels and the same predefined set of labels.*

We go through 2 iterations as we already did in Section 3.2.1. When we apply Algorithm

6 to the graph, we will get the following changes:

Steps 1-9 *The Y matrix has 10 rows, where initially all elements are only 0 or 1. We use the same techniques as before to fill it, so we do not cover the details here anymore.*

$$Y = \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} \tag{3.9}$$

Step 10 *Initializing matrix P is based on the connections that seed nodes have and is calculated exactly in the same way, as it is done in Section 3.2.1.*

$$P = \begin{bmatrix} \frac{2}{7} & \frac{5}{9} \\ \frac{5}{7} & \frac{4}{9} \end{bmatrix} \tag{3.10}$$

Iteration 1, Step 11 *Collecting information using the appropriate propagation strategy. Let us discuss different cases. We can take vertex 0 and vertex 1 to see how changes occur. As vertex 0 is a type 1 node, we use LPA to collect information from its neighbors of type 2, but none of them has a label. Their rows in matrix Y in (3.9) just have 0's. Therefore row 0 in matrix Y in (3.11) is filled with just 0's. For vertex 1, we collect information from nodes 0, 2 and 7. Taking into account that*

vertex 0 has 2 neighbors, vertex 2 has just one and vertex 7 has 4 of them, we get: $Y_{10} = \frac{2}{7} \cdot 0/2 + \frac{5}{9} \cdot 1/2 + \frac{2}{7} \cdot 0/1 + \frac{5}{9} \cdot 0/1 + \frac{2}{7} \cdot 0/4 + \frac{5}{9} \cdot 1/4$. *Thus* $Y_{10} = \frac{5}{12}$. *Using the same procedure for* Y_{11}, *we get* $Y_{11} = \frac{4}{12}$.

$$Y = \begin{bmatrix} 0 & 0 \\ \frac{5}{12} & \frac{4}{12} \\ 0 & 0 \\ \frac{47}{84} & \frac{58}{84} \\ 0 & 0 \\ 0 & 0 \\ \frac{107}{252} & \frac{208}{252} \\ 0 & 0 \\ 0 & 0 \\ \frac{71}{252} & \frac{118}{252} \\ 0 & 0 \end{bmatrix} \tag{3.11}$$

Iteration 1, Step 12 *Row-normalization:*

$$Y = \begin{bmatrix} 0 & 0 \\ \frac{5}{9} & \frac{4}{9} \\ 0 & 0 \\ \frac{47}{105} & \frac{58}{105} \\ 0 & 0 \\ 0 & 0 \\ \frac{107}{315} & \frac{208}{315} \\ 0 & 0 \\ 0 & 0 \\ \frac{71}{189} & \frac{118}{189} \\ 0 & 0 \end{bmatrix}$$

Iteration 1, Step 13 *Reset the rows of the seed noes:*

$$Y = \begin{bmatrix} 0 & 1 \\ \frac{5}{9} & \frac{4}{9} \\ 0 & 0 \\ \frac{47}{105} & \frac{58}{105} \\ 1 & 0 \\ 0 & 0 \\ \frac{107}{315} & \frac{208}{315} \\ 0 & 1 \\ 0 & 0 \\ \frac{71}{189} & \frac{118}{189} \\ 1 & 0 \end{bmatrix}$$

Iteration 2, Step 11 *Collecting information using appropriate strategy:*

$$Y = \begin{bmatrix} \frac{316}{945} & \frac{314}{945} \\ \frac{5}{12} & \frac{4}{12} \\ \frac{5}{9} & \frac{4}{9} \\ \frac{47}{84} & \frac{58}{84} \\ \frac{342}{1575} & \frac{498}{1575} \\ \frac{107}{1575} & \frac{208}{1575} \\ \frac{107}{252} & \frac{208}{252} \\ \frac{1859}{3150} & \frac{2446}{3150} \\ \frac{107}{1575} & \frac{208}{1575} \\ \frac{71}{252} & \frac{118}{252} \\ \frac{2417}{9450} & \frac{4198}{9450} \end{bmatrix}$$

Iteration 2, Step 12 *Row-normalization:*

$$Y = \begin{bmatrix} \frac{158}{315} & \frac{157}{315} \\ \frac{5}{9} & \frac{4}{9} \\ \frac{5}{9} & \frac{4}{9} \\ \frac{47}{105} & \frac{58}{105} \\ \frac{171}{420} & \frac{249}{420} \\ \frac{107}{315} & \frac{208}{315} \\ \frac{107}{315} & \frac{208}{315} \\ \frac{1859}{4305} & \frac{2446}{4305} \\ \frac{107}{315} & \frac{208}{315} \\ \frac{71}{189} & \frac{118}{189} \\ \frac{2417}{6615} & \frac{4198}{6615} \end{bmatrix}$$

Iteration 2, Step 13 *Reset the rows of the seed nodes:*

$$Y = \begin{bmatrix} 0 & 1 \\ \frac{5}{9} & \frac{4}{9} \\ \frac{5}{9} & \frac{4}{9} \\ \frac{47}{105} & \frac{58}{105} \\ 1 & 0 \\ \frac{107}{315} & \frac{208}{315} \\ \frac{107}{315} & \frac{208}{315} \\ 0 & 1 \\ \frac{107}{315} & \frac{208}{315} \\ \frac{71}{189} & \frac{118}{189} \\ 1 & 0 \end{bmatrix}$$

Taking into account the stopping criterion which we use, the iteration process stops after 5 iterations and Y matrix takes the following form:

$$Y = \begin{bmatrix} 0 & 1 \\ 0.483101 & 0.516899 \\ 0.469892 & 0.530108 \\ 0.447619 & 0.552381 \\ 1 & 0 \\ 0.416121 & 0.583879 \\ 0.403428 & 0.596572 \\ 0 & 1 \\ 0.416121 & 0.583879 \\ 0.375661 & 0.624339 \\ 1 & 0 \end{bmatrix} \tag{3.12}$$

We go to Step 15 now.

Step 15 *Taking into account the information which is represented in the matrix Y in (3.12), we get the following labels for previously unlabeled nodes: y_1 = "female", y_2 = "female", y_3 = "female", y_5 = "female", y_6 = "female", y_8 = "female" and y_9 = "female".*

As we can see latter results are different from the labels which we got in Example 3.2.1. In the latter example all vertices were predicted as "female", while some nodes got "male" labels in Example 3.2.1. Hence based on the structure of the graph Algorithm 3 and Algorithm 6 may lead us to different results.

3.3.2 Stopping Criteria

Since Algorithm 6 does not differ a lot from Algorithm 3 described in section 3.2.1, we use the exactly same stopping criterion which is provided in Section 3.2.2.

3.3.3 Scalable Implementation

The implementation details of the method proposed in Section 3.3.1 are not much different from the approach which is presented in section 3.2.3. As the propagation directions are not the same for these two algorithms, the only difference in the implementation occurs during the "Scatter" phase. More details can be found in Algorithm 7.

Algorithm 7 Scatter phase for User-item Adaptive-LPA with Item-user LPA

Scatter:
for $i \in Neighbors(j)$ **do**
  if $type_j = 1$ **then**
    for $s = 1, 2, \ldots, n$ **do**
      for $k = 1, 2, \ldots, n$ **do**
        $Temp_{is} \leftarrow Temp_{is} + P_{sk} Y_{jk}/n_j$
      end for
    end for
  else
    $Temp_i \leftarrow Temp_i + Y_j/n_j$
  end if
end for

Chapter 4

DATASETS

For the experimental results we use a subset of the MyPersonality project dataset[1]. MyPersonality was a popular Facebook application introduced in 2007 in which users took a standard Big Five Factor Model psychometric questionnaire and gave consent to record their responses and Facebook profile. The dataset consists of information about users' demographics, personality traits, and activities. Not all of this information is available for all users, therefore we selected users who have age, gender, the Big5 personality traits [2], namely openness to experience (Opn), conscientiousness (Con), extraversion (Ext), agreeableness (Agr), and neuroticism (Neu), and page likes. The Big5 personality traits are rated on a scale from 1 to 5. More details about them are summarized in Table 4.2

Our sample dataset includes 28,525,505 page likes, 126,468 users and 3,874,202 pages. As we want to label the users as "having the specific personality trait or not", we treat it as a vinary classification problem. Thus we made two groups for each personality trait based on the median score. The median values for all traits are presented in Table 4.1. To classify users' age, we made six age groups as follows $[0, 17)$,$[17, 24)$,$[24, 31)$,$[31, 46)$,$[46, 73)$, $[73, 112]$.

Table 4.1: Median score of the Big5 personality traits in the Facebook dataset.

traits	Opn	Con	Ext	Agr	Neu
Median	4	3.5	3.67	3.65	2.75

To create different graphs, we follow a snowball sampling approach where we start from a random user and select all pages that a user likes and then collect all users liking those pages.

[1] http://mypersonality.org/

Table 4.2: Description of the Big5 personality traits.

Trait	Description
Openness	Appreciation for art, emotion, adventure, unusual ideas, curiosity, and variety of experience. Openness reflects the degree of intellectual curiosity, creativity and a preference for novelty and variety a person has. People who are high in this trait tend to be more adventurous and creative.
Conscientiousness	It includes high levels of thoughtfulness, with good impulse control and goal-directed behaviors, preference of planned rather than spontaneous behavior. people who are high on conscientiousness tend to be organized and mindful of details.
Extraversion	Energy, positive emotions, surgency, assertiveness, sociability and the tendency to seek stimulation in the company of others, and talkativeness. People who are high in extroversion are outgoing and tend to gain energy in social situations. People who are low in extroversion (or introverted) tend to be more reserved.
Agreeableness	A tendency to be compassionate and cooperative rather than suspicious and antagonistic towards others. It is also a measure of one's trusting and helpful nature, and whether a person is generally well-tempered or not. People who are high in agreeableness tend to be more cooperative while those low in this trait tend to be more competitive and even manipulative.
Neuroticism	It is characterized by sadness, moodiness, and emotional instability. Neuroticism also refers to the degree of emotional stability and impulse control and is sometimes referred to by its low pole, "emotional stability". high need for stability manifests as a stable and calm personality, but can be seen as uninspiring and unconcerned. A low need for stability causes a reactive and excitable personality.

Name	# of users	# of pages	# of nodes	# of edges
Facebook 10%	12,646	1,222,000	1,234,646	5,237,573
Facebook 20%	25,293	1,871,628	1,896,921	11,704,616
Facebook 30%	37,940	2,280,430	2,320,370	15,375,255
Facebook 40%	50,587	2,640,969	2,691,556	18,699,856
Facebook 50%	63,234	2,933,590	2,996,824	21,694,284
Facebook 60%	75,880	3,208,188	3,284,068	24,491,403
Facebook 70%	88,527	3,413,568	3,502,095	26,540,556
Facebook 80%	101,174	3,560,184	3,661,358	27,422,054
Facebook 90%	113,821	3,745,237	3,859,058	28,164,121
Facebook 100%	126,468	3,874,202	4,000,670	28,525,505

Table 4.3: Statistics of 10 sample graphs based on the Facebook dataset.

We continue collecting users and pages to reach $x\%$ of the users where $x = \{10, 20, \ldots, 100\}$. Then we select all the page likes of these users from the dataset to make a sample. With this approach we created 10 samples with various graph sizes, the statistics of which are presented in Table 4.3.

To conduct our experiments the users are divided into n different parts, where $n \in \{2, 3, 5, 10\}$. The users for different folds were selected randomly and none of the folds intersect.

Chapter 5

RESULTS

Our goal in this chapter is to evaluate the label propagation strategies presented in the previous chapters in terms of accuracy and scalability. To this end we apply them to predict age, gender and personality of Facebook users given their page likes. We perform an extensive analysis based on various sizes of graphs, different sizes of sets of seed nodes and stopping criteria. Besides the comparison in terms of accuracy for different strategies, our interests also lie into the running time of Algorithm 2 and its ability to scale up to large datasets. We present the running time results for different clamping approaches in terms of number of iterations. Accuracy results of different algorithms with different stopping criteria and portions of seed nodes are presented in Section 5.1, while running time results to demonstrate scalability are discussed in Section 5.2.

5.1 Accuracy

5.1.1 Effect of Clamping Approaches and Stopping Criteria

As our interest lies into the behavior of LPA, we applied it to the Facebook graphs. As we need reliable experimental results and the choice of specific set of seed nodes might affect the results, to address this problem we used 10 different sets of randomly chosen seed nodes (each has 10% of users) for every experiment, except the one where we change the portion of seed nodes. As our chosen sets of seed nodes do not intersect, when we increase the size of each set, their number decreases (e.g., if the set of seed nodes include 20% of users, than we have 5 different sets). After that we took an average of obtained results. All the results were compared to majority baseline. The majority baseline predicts for each node the frequent label across the seed nodes.

To get insight into the influence of LPA over prediction during iterations, we applied it to the Facebook100% graphs, varying the number of iterations. This means that we did not clamp anything except the seed nodes and that we force the propagation process to stop after a fixed number of iterations (sometimes not every node has a label). From the results, which are presented in Figure 5.1, it can be concluded that for almost all personality traits increasing the number of iterations will lead us to better accuracy. The results for age and gender showed, that more iterations made prediction accuracy worse. Hence we are concentrated on personality trait prediction tasks using LPA and did not include the results for other ones.

To stop the iterative process in a more natural way than fixing number of iterations, next we used a clamping approach, which means that a label for a node is decided if it has not been changed during several consecutive iterations. Using this strategy with Algorithm 2 leads us to similar results as fixing number of iterations. According to the graphs, which are presented in Figure 5.2, increasing the number of consecutive iterations for deciding a label for a node leads us to better accuracy results for all personality trait prediction tasks, although the difference is insignificant.

Except "number of iterations" based stopping approaches, we also applied Algorithm 2 to all datasets which are presented in Table 4.3 using the L_2 norm and a threshold for it as a stopping criterion. Experimental results which are presented in Figure 5.3 offer us insight into the influence of changing the threshold for the L_2 norm. It can be deduced that when the threshold is decreased, the prediction accuracy of all personality traits comes nearer to majority baseline, while increasing it gives us better results, although the differences are insignificant for most of the traits. We can also see the same behavior w.r.t prediction of age and gender. In this case the difference between using thresholds of 0.1 and 0.01 is clearly visible. As lower threshold means that LPA needs more iterations to stop, taking into account the results which are presented in Figure 5.1, we expected that decreasing the threshold would lead us to better accuracy. As it turned out the results presented in

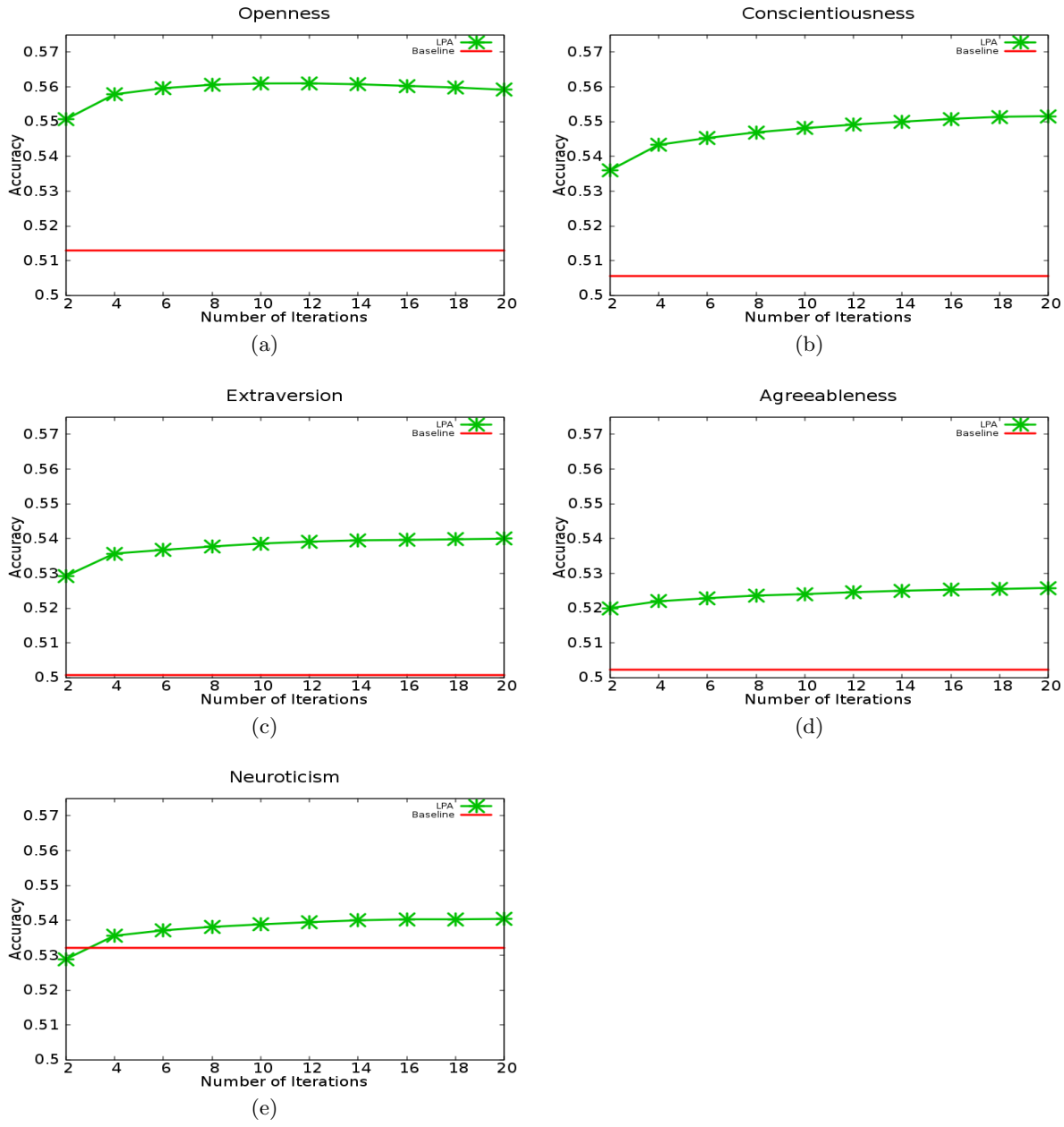


Figure 5.1: Accuracy results of using LPA with different number of iterations on the Facebook100% dataset for (a) Openness, (b) Conscientiousness, (c) Extroversion, (d) Agreeableness, and (e) Neuroticism. Results are represented as 2D plots where the x axis is the number of iterations and the y axis is the accuracy of the prediction.

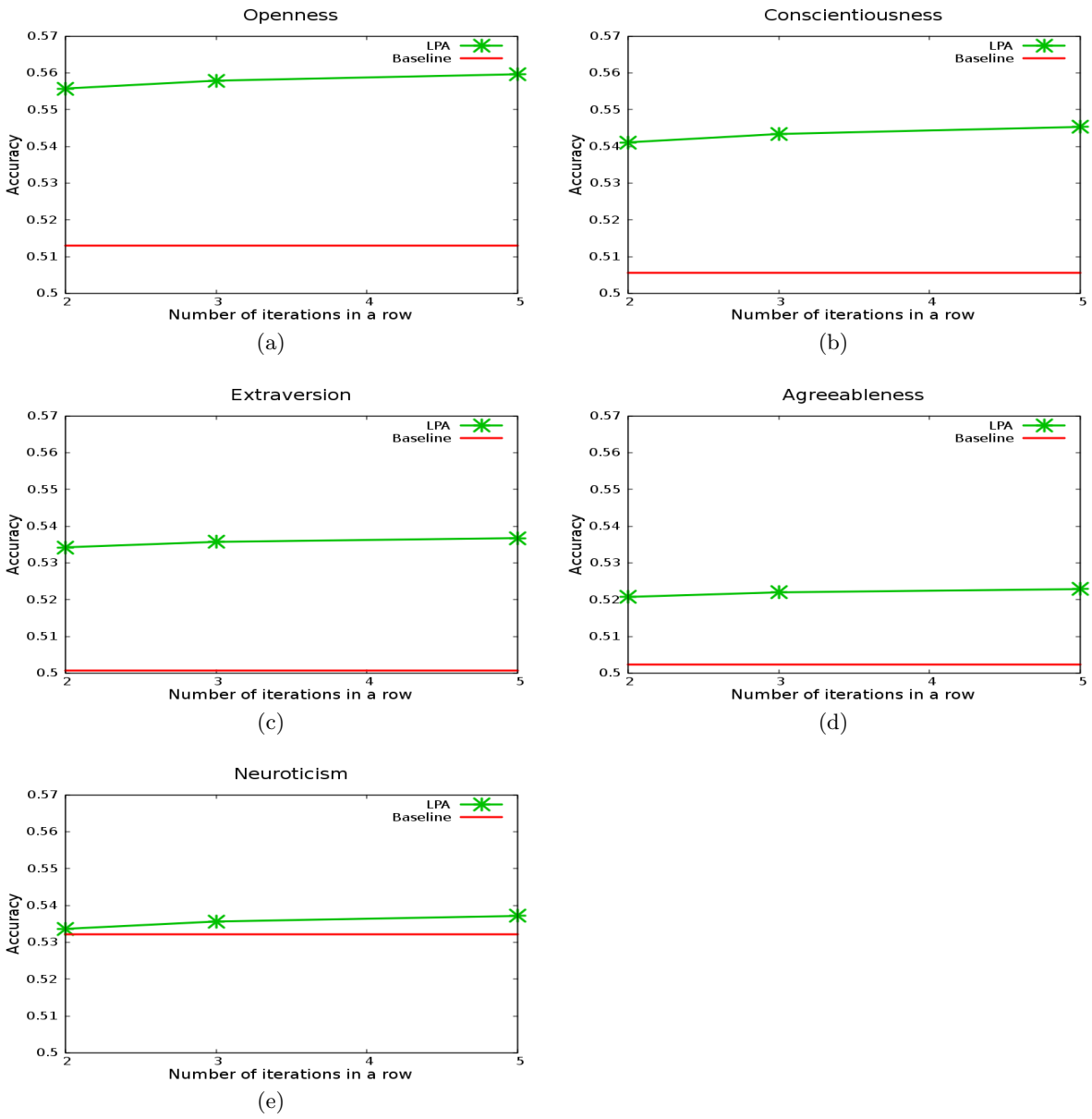


Figure 5.2: Accuracy results of using Algorithm 2 with Iteration-Clamping Strategy with different number of consecutive iterations on the Facebook 100% dataset for (a) Openness, (b) Conscientiousness, (c) Extroversion, (d) Agreeableness, and (e) Neuroticism. Results are represented as 2D plots where the x axis is the number of iterations in a row after which clamping happens and the y axis is the accuracy of the prediction.

Approach	Gender	EXT	OPN	CON	NEU	AGR
Baseline	0.613	0.516	**0.653**	0.508	0.545	0.510
Vertex-Centric based LPA (2)	0.689	0.536	0.647	**0.543**	**0.547**	**0.529**
User-item LPA with item-user Adaptive-LPA (1)	0.673	0.529	0.631	0.534	0.523	0.503
User-item LPA with item-user Adaptive-LPA (2)	**0.691**	0.536	0.647	**0.543**	0.527	0.506
User-item LPA with item-user Adaptive-LPA (3)	0.687	0.530	0.639	0.537	0.524	0.504
User-item Adaptive-LPA with item-user LPA (1)	0.672	0.529	0.621	0.534	0.520	0.503
User-item Adaptive-LPA with item-user LPA (2)	0.685	**0.538**	0.615	**0.543**	0.522	0.504
User-item Adaptive-LPA with item-user LPA (3)	0.685	0.530	0.622	0.537	0.520	0.505

Table 5.1: Accuracy results based on the smallest Facebook sample (Facebook10%) using different algorithms

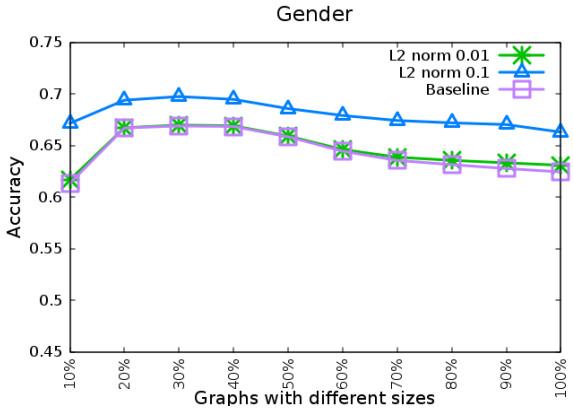
Figure 5.3 are not consistent with the ones which we have showed in Figure 5.1.

From the results which have already been discussed, we can conclude that Iteration-Clamping strategy works better than other ones. Until this time every experiment was held for Algorithm 2 from Chapter 2 to see its behavior w.r.t. different stopping criteria. According to the results, we decided to use the clamping strategies for our proposed algorithms which were presented in Chapter 3 for further experiments.

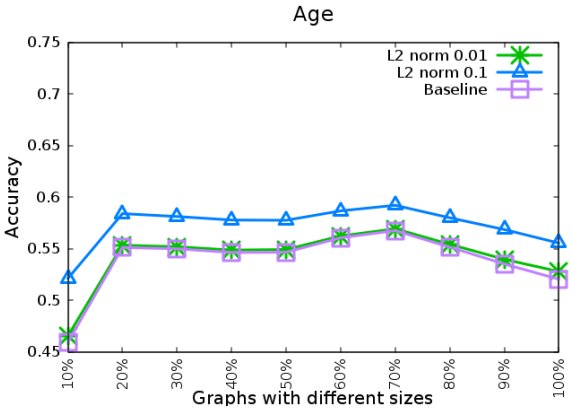
We used the following stopping strategies in combination with the LPA algorithms which were presented in Chapters 2 and 3:

- Strategy (1) : Seed-Clamping
- Strategy (2) : Iteration Clamping with number of iterations $i = 1$
- Strategy (3) : Iteration Clamping with number of iterations $i = 2$

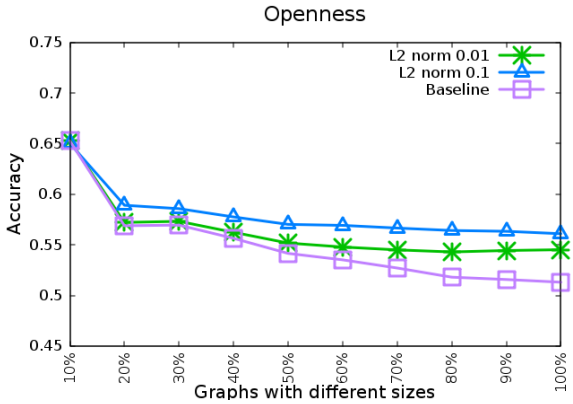
Table 5.1 includes experimental results of the different algorithms using various strategies for clamping. All experiments were done on the Facebook 10% sample. In most cases


Gender
L2 norm 0.01
L2 norm 0.1
Baseline
Accuracy
0.75
0.7
0.65
0.6
0.55
0.5
0.45
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%
Graphs with different sizes

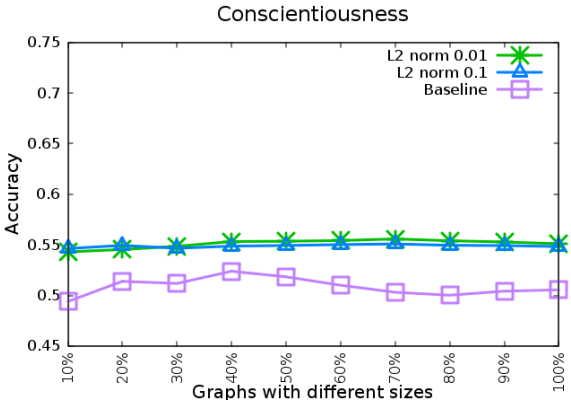
(a)


Age
L2 norm 0.01
L2 norm 0.1
Baseline
Accuracy
0.75
0.7
0.65
0.6
0.55
0.5
0.45
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%
Graphs with different sizes

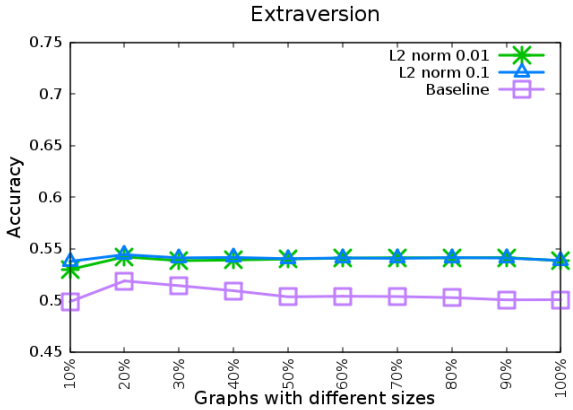
(b)


Openness
L2 norm 0.01
L2 norm 0.1
Baseline
Accuracy
0.75
0.7
0.65
0.6
0.55
0.5
0.45
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%
Graphs with different sizes

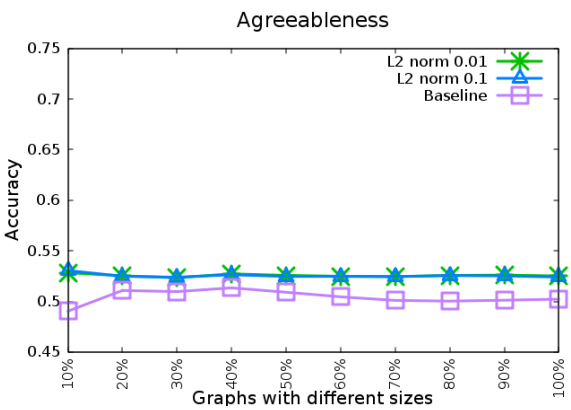
(c)


Conscientiousness
L2 norm 0.01
L2 norm 0.1
Baseline
Accuracy
0.75
0.7
0.65
0.6
0.55
0.5
0.45
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%
Graphs with different sizes

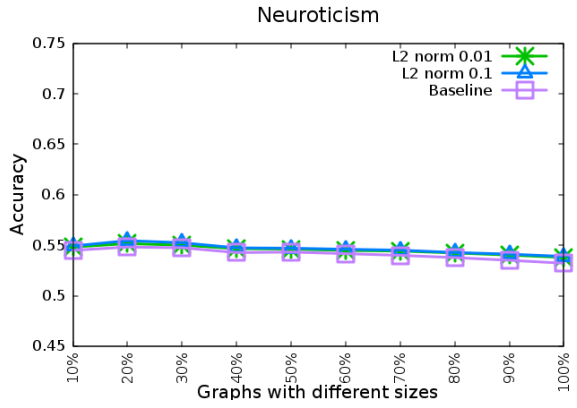
(d)


Extraversion
L2 norm 0.01
L2 norm 0.1
Baseline
Accuracy
0.75
0.7
0.65
0.6
0.55
0.5
0.45
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%
Graphs with different sizes

(e)


Agreeableness
L2 norm 0.01
L2 norm 0.1
Baseline
Accuracy
0.75
0.7
0.65
0.6
0.55
0.5
0.45
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%
Graphs with different sizes

(f)



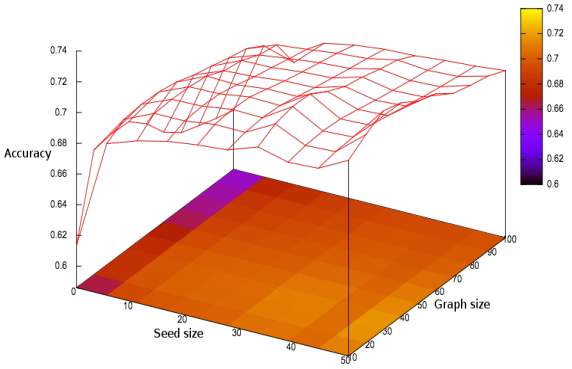
(g)

Figure 5.3: Accuracy results of using Algorithm 2 with stopping criterion of L_2 norm on Facebook graph (a) Gender, (b) Age, (c) Openness, (d) Conscientiousness, (e) Extroversion, (f) Agreeableness, and (g) Neuroticism. Results are represented as 2D plots where the x axis is the graph size and y axis is the prediction accuracy. The purple line corresponds to results of the majority baseline, while the green one denotes the threshold of 0.01 and the blue one corresponds the threshold of 0.1.

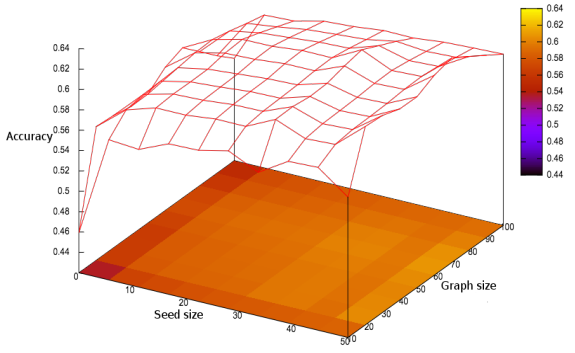
Algorithm 2 leads us to the best accuracy, but sometimes none of them can beat the majority baseline. Since the experimental results done on the Facebook100% sample are not better than baseline and in the most cases they are lower, we do not include them here.

5.1.2 Seed/Graph size results

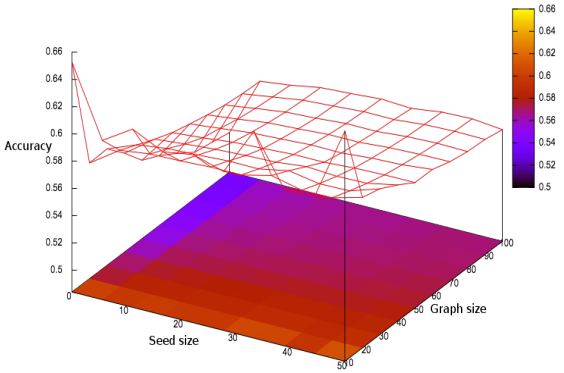
The accuracy of prediction by Algorithm 2 based on different sizes of graphs, which are presented in Table 4.3, and various sizes of seed nodes (10%, 20%, 33% and 50%) are presented in Figure 5.4. We varied size of the graph and portion of seed nodes to observe their influence over the accuracy of prediction of LPA. For all traits, increasing the size of the seed nodes from 10% to 50% improves the prediction results. However increasing the size of the graph does not necessary increase the accuracy of the predictions. This can also be observed by the results which are presented in Figure 5.5, where increasing the size of the graph does not give us better accuracy.


Accuracy
Seed size
Graph size

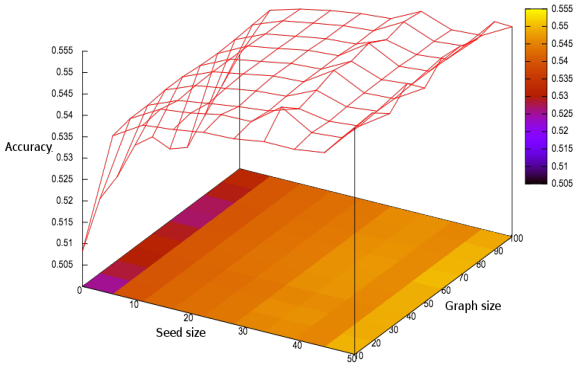
(a)


Accuracy
Seed size
Graph size

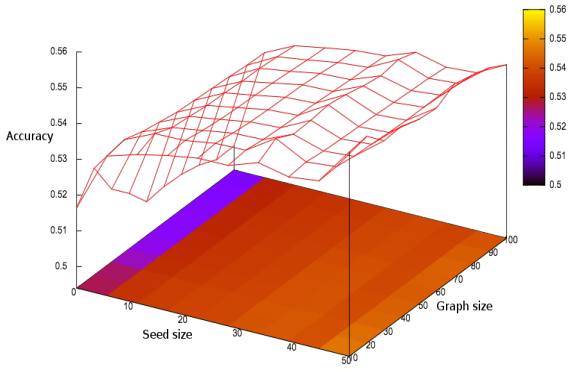
(b)


Accuracy
Seed size
Graph size

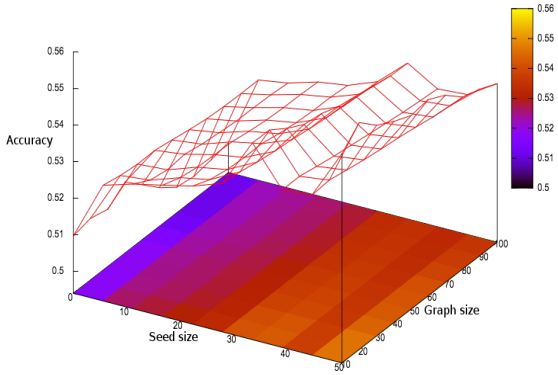
(c)


Accuracy
Seed size
Graph size

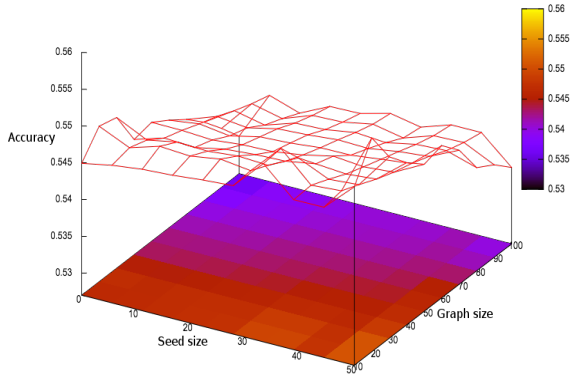
(d)


Accuracy
Seed size
Graph size

(e)


Accuracy
Seed size
Graph size

(f)



(g)

Figure 5.4: Accuracy results of using Algorithm 2 on the Facebook datasets for (a) Gender, (b) Age, (c) Openness, (d) Conscientiousness, (e) Extroversion, (f) Agreeableness, and (g) Neuroticism. Results are represented as 3D plots where the x axis is the seed size, the y axis is the graph size and the z axis is the accuracy of the prediction. The colored contour map or density distribution is mapped on the bottom plain, where lighter colors are higher accuracy.

5.2 Scalability Results

Besides the accuracy results of the provided approaches, we are also interested in their running time to get insight into their scalability. We ran our Grappa implementation on the Sampa group cluster at the University of Washington. It is a cluster of dual 6-core, 2.66 GHz Intel Xeon X5650 processors with 24 GB of memory per node, connected with a 40Gb Mellanox ConnectX-2 InfiniBand network.

We applied Algorithm 2 with Iteration-Clamping strategies to the graphs of different sizes. To get reliable results, we uses 10 different sets of randomly chosen seed nodes every time, then we took an average of obtained results. Our desire is to find out how changing the number of consecutive iterations affects the running time. We ran the algorithm for all datasets provided in Table 4.3 on a varying number of computational nodes using different numbers of consecutive iterations. From the results, which are presented in Figure 5.6, it can be concluded that after some point there is no use in increasing the number of compu-

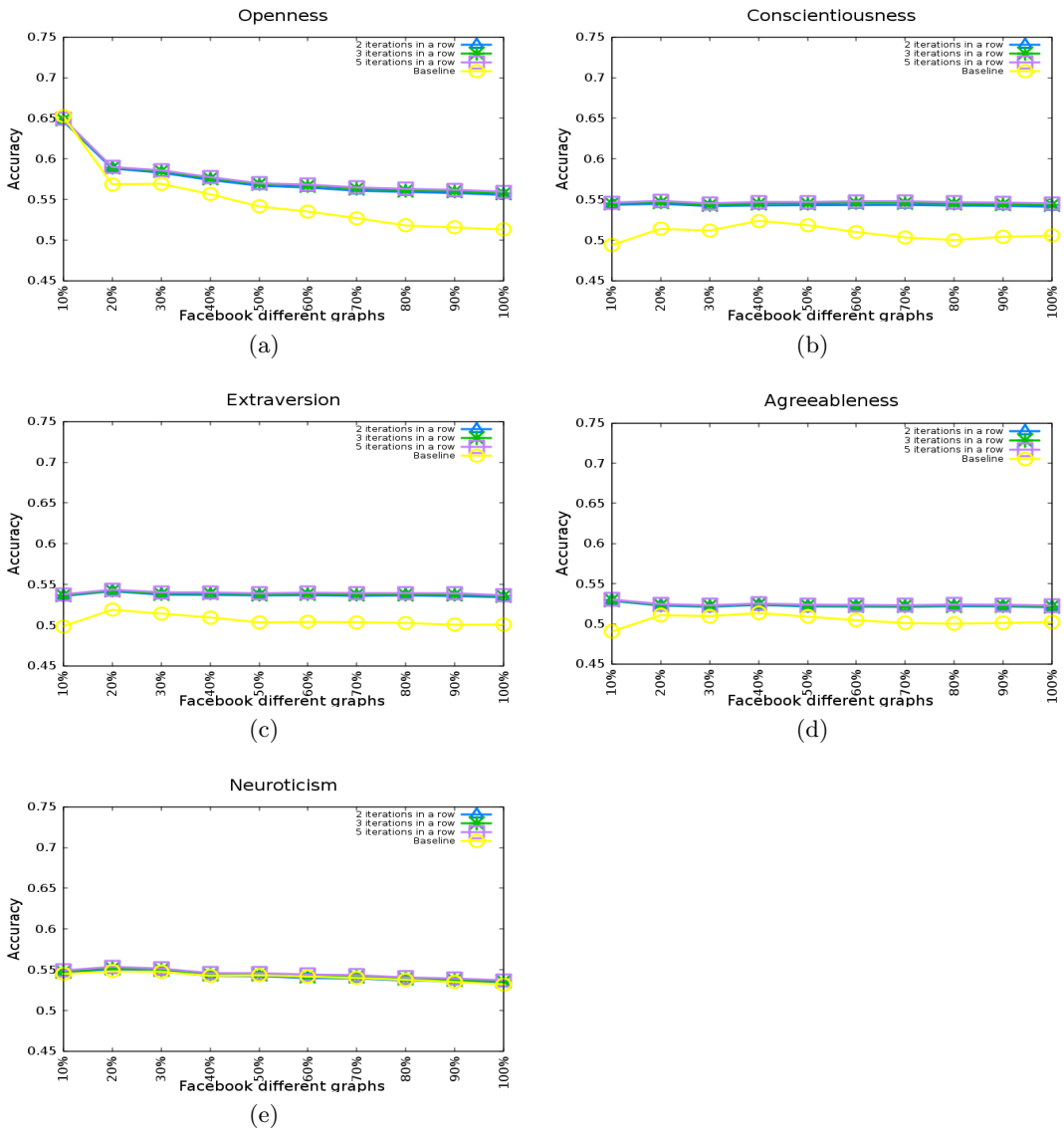
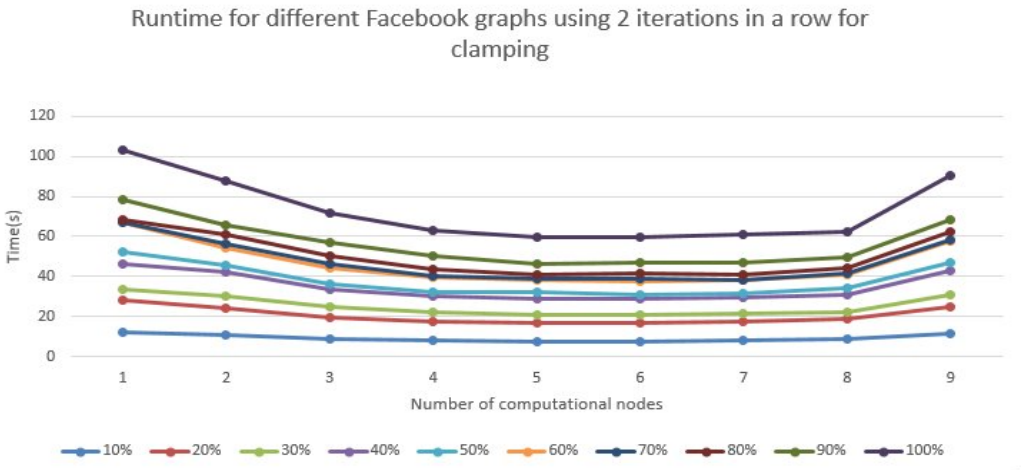
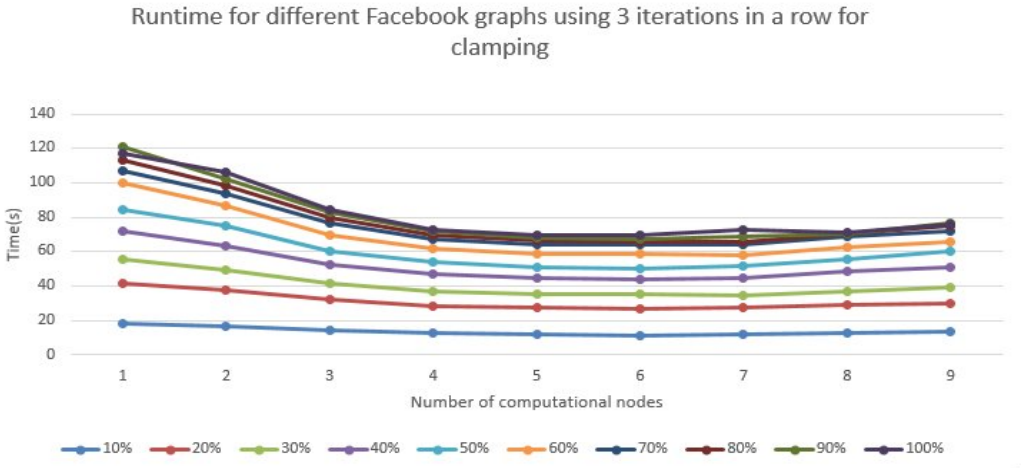


Figure 5.5: Accuracy results of using Algorithm 2 with Iteration-Clamping Strategy with different number of consecutive iterations on the Facebook dataset for (a) Openness, (b) Conscientiousness, (c) Extroversion, (d) Agreeableness, and (e) Neuroticism. Results are represented as 2D plots where the x axis is the graph size and the y axis is the accuracy of the prediction. The purple line corresponds to clamping after 5 iterations in a row, the green one denotes 3 iterations and blue corresponds to 2 iterations. The baseline is represented with a yellow line.

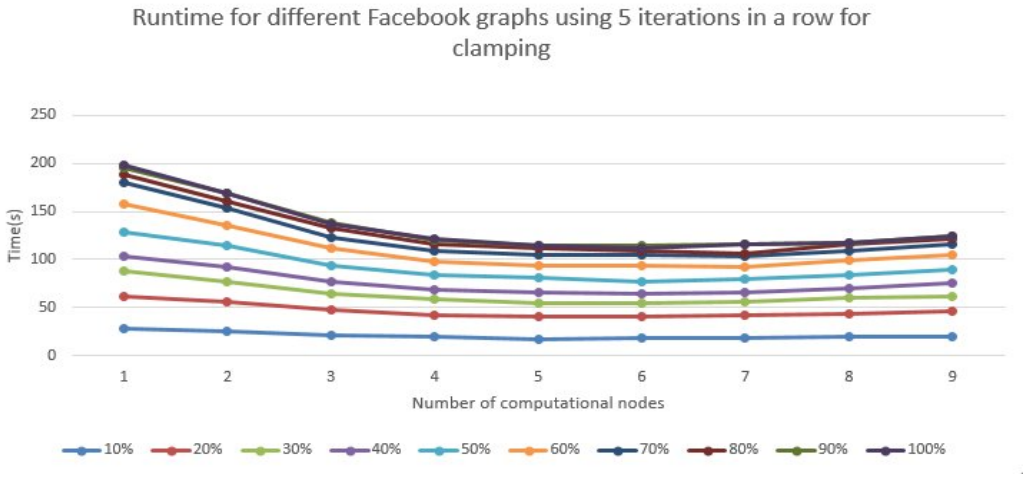
tational nodes. The optimal number can be assumed to be 6 or 7 for all different number of iterations. Using more computational nodes induces a computational time increase. In spite of decreasing the running time when the number of nodes is doubled until it reaches 8, we can not reduce the running time to half and achieve linear scaling. For more details please refer to Figure 5.6.



(a)



(b)



(c)

Figure 5.6: Scalability results of using LPA with Iteration-Clamping strategy on Facebook graph (a) 2 iterations in a row, (b) 3 iterations in a row, (c) 5 iterations in a row. Results are represented as 2D plots where the x axis is the number of computational nodes and the y axis is computational time.

5.3 Conclusion

The experimental results, which are provided in Section 5.1, give us more insight into the behavior of Algorithm 2 using different stopping criteria. As it turned out, increasing the number of iterations leads us to better accuracy for all personality traits, while decreasing the threshold of the L_2 norm also decreases the prediction accuracy. As decreasing the threshold means more number of iterations, it may be concluded that after some point increasing number of iterations causes the drop of prediction accuracy.

The results showed that our proposed algorithms do not provide the satisfactory results in terms of the prediction accuracy. Sometimes it is lower than baseline. The reason can also lie in the structure of the graph. Majority of labels of seed nodes might have a big influence over the results if most of them are connected to the other type of nodes.

Our observations over scalability results of Algorithm 2 showed that the running time decreases when we increase the number of computational nodes to some point. This can be

caused by the size of the graph too. As long as the algorithm needs a lot of information exchange between computational nodes during each iteration, when a graph is not large enough, its distribution over multiple nodes might cause the huge loss of the time.

Chapter 6

CONCLUSION

6.1 Summary

In Chapter 2 we have first reviewed an existing label propagation algorithm (LPA). To complement the traditional, matrix multiplication based view of LPA, we have presented a vertex-centric interpretation as well. We have illustrated through examples that both interpretations are the same w.r.t. the work they do. In order to make LPA applicable for large graphs, we have proposed an implementation on top of the distributed shared memory system Grappa, using a vertex-centric GraphLab-like API in particular.

In Chapter 3, we have proposed two novel label propagation algorithms specifically for bipartite graphs: (1) User-item LPA with Item-user Adaptive-LPA and (2) User-item Adaptive-LPA with Item-user LPA, along with different clamping strategies. We have implementated both approaches using Grappa's vertex-centric API, and have presented pseudo-code for the corresponding algorithms in Chapter 3.

Next, in Chapter 5, we have first compared the accuracy of different stopping criteria and clamping strategies for LPA on different Facebook graphs introduced in Chapter 4. We ran our experiments on a Facebook dataset, for the task of inferring 7 different kinds of labels for Facebook users, namely age, gender and their personality. The results showed that LPA works better for inferring personality traits of users, while the results of inferring age and gender come closer to baseline when iterative process lasts for longer time. From the experimental results we can conclude that Iteration-Clamping strategy works better than other ones. Using this clamping strategy also means that we stop the process when none of the nodes changes its most probable label during the iteration.

Then we have compared the accuracy of the two algorithms proposed in Chapter 3 to

the LPA algorithm from Chapter 2. Our experimental results showed that our proposed algorithms from Chapter 3 do not outperform the LPA algorithm from Chapter 2 w.r.t. accuracy. From our observations we have found out that the majority of labels of seed nodes have an influence over results. If most seed nodes' connections to intermediate vertices are from the ones, which have the same labels as majority of the seed nodes, accuracy result of the algorithms proposed in Chapter 3 become closer to baseline.

Finally, we have investigated the scalability of our Grappa implementation of LPA experimentally. On our largest test graphs, i.e. 4,000,670 vertices and 28,525,505 edges, LPA took no more than 200 sec using one computational node. Increasing the number of computational nodes, i.e. allowing for parallelization, decreased the runtime to 112 sec when using 6 computational nodes. A further increase in the number of computational nodes did not lead to a further decrease in runtime anymore. A reasonable explanation for this is the time loss during the interchanging of the information between computational nodes.

6.2 Future Work

In this work we assumed that labels are given only for one type of nodes in the bipartite graph, i.e. all the seed nodes belong to only one category of the nodes. An interesting direction for future work would be to consider the scenario where we are given labeled seed nodes from both categories. In this scenario, a label propagation approach would be able to leverage this additional information to determine the relation between different classes from different types of nodes. This scenario can be divided into two cases: (1) we are given all labeled nodes from the one category, while we know only a portion of labels for another type of nodes and (2) we are given different portions of seed nodes from different categories.

During first case, the algorithm will infer labels only for the nodes from one category. For the second situation the label propagation approach would also be able to infer the labels for both types of nodes. Taking this fact into account, this future research would also involve the construction of new algorithms, which will be applicable for inferring labels for both sides of bipartite graphs.

These new algorithms will also be applicable for document classification task. It can be represented as a bipartite graph, where one type of nodes determine documents, while another kind of nodes correspond to terms, and edges denote term occurrence in documents.

BIBLIOGRAPHY

[1] Michael J. Barber and John W. Clark. Detecting network communities by propagating labels under constraints. *Physical Review E*, 80(2):026129, 2009.

[2] Paul T Costa and Robert R McCrae. The revised neo personality inventory (NEO-PI-R). *The SAGE Handbook Of Personality Theory And Assessment*, 2:179–198, 2008.

[3] Chris Ding, Tao Li, and Dinging Wang. Label propagation on k-partite graphs. *International Conference on Machine Learning and Applications (ICMLA'09)*, pages 273–278, 2009.

[4] Thiago P. Faleiros and Alneu A. Lopes. Bipartite graph for topic extraction. *Proc. International Conference on Artificial Intelligence (IJCAI'15)*, pages 4361–4362, 2015.

[5] Golnoosh Farnadi, Zeinab Mahdavifar, Ivan Keller, Jacob Nelson, Ankur Teredesai, Marie-Francine Moens, and Martine De Cock. Scalable adaptive label propagation in grappa. In *IEEE International Conference on Big Data (IEEE-BigData 2015)*. IEEE, 2015.

[6] Steve Gregory. Finding overlapping communities in networks by label propagation. *New Journal of Physics*, 12:103018, 2010.

[7] William Gropp, Ewing Lusk, and Anthony SKjellum. *Using MPI: portable parallel programming with the message-passing interface*, volume 1. MIT press, 1999.

[8] Rob Hall, Stephen E Fienberg, and Yuval Nardi. Secure multiple linear regression based on homomorphic encryption. *Journal of Official Statistics*, 27(4):669–691, 2011.

[9] Yucheng Low, Joseph Gonzalez, Aapo Kyrola, Danny Bickson, Carlos Guestrin, and Joseph M. Hellerstein. Distributed GraphLab: A framework for machine learning in the cloud. In *VLDB Endowment (PVLDB)*, volume 5(8), pages 716–727, 2012.

[10] Jacob Nelson, Brandon Holt, Brandon Myers, Preston Briggs, Luis Ceze, Simon Kahan, and Mark Oskin. Latency-tolerant software distributed shared memory. *In 2015 USENIX Annual Technical Conference (USENIX ATC 15)*, 2015.

[11] Mark E.J. Newman. Fast algorithm for detecting community structure in network. *Physical Review E*, 69(6):066133, 2004.

[12] Mark E.J. Newman. Finding community structure in mega-scale social networks. *Proc. 16th international conference on World Wide Web*, pages 1275–1276, 2007.

[13] Usha N. Raghavan, Reka Albert, and Soundar Kumara. Efficient modularity optimization by multistep greedy algorithm and vertex mover refinement. *Physical Review E*, 76(4):046112, 2008.

[14] Rafael G. Rossi, Alneu A. Lopes, and Solange O. Rezende. A parameter-free label propagation algorithm using bipartite heterogeneous networks for text classification. *Proc. 29th Annual ACM Symposium on Applied Computing (SAC'14)*, page 7984, 2014.

[15] Rafael G. Rossi, Alneu A. Lopes, and Solange O. Rezende. Optimization and label propagation in bipartite heterogeneous networks to improve transductive classification of texts. *Information Processing & Management*, 52(2):217257, 2016.

[16] Philipp Schuetz and Amedeo Caflisch. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 77(3):036106, 2007.

[17] Jierui Xie and Boleslaw K. Szymanski. Community detection using a neighborhood strength driven label propagation algorithm. *Proc.IEEE Network Science Workshop, West Point*, NY June 22-24:188–195, 2011.

[18] Xiao-Ying Yan, Shao-Wu Zhang, and Song-Yao Zhang. Prediction of drugtarget interaction by label propagation with mutual interaction information derived from heterogeneous network. *Molecular BioSystems*, 12(2):520–531, 2016.

[19] Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. *Technical Report CMU-CALD-02-107*, Carnegie Mellon University, 2002.