Sampling-based Label Propagation for Balanced Graph Partitioning

Adnan El Moussawi
Université Paris-Saclay, CNRS, Laboratoire Interdisciplinaire des Sciences du Numérique
Orsay, France
adnan.el-moussawi@lisn.fr

Ricardo Rojas Ruiz
Université Paris-Saclay, CNRS, Laboratoire Interdisciplinaire des Sciences du Numérique
Orsay, France
ricardo.rojas@student-cs.fr

Nacéra Bennacer Seghouani
Université Paris-Saclay, CNRS, Laboratoire Interdisciplinaire des Sciences du Numérique
Orsay, France
nacera.seghouani@lisn.fr

ABSTRACT

In this experience paper, we present new sampling-based algorithms for balanced graph partitioning based on the Label Propagation (LP) approach. The purpose is to define new heuristics to extend the multi-objective and scalable Balanced GRAph Partitioning algorithm B-GRAP proposed in [9]. The main challenge is related to how to build a graph sample that ensures stability and improves the convergence and the partitioning quality which depend strongly on the structure of the graph. We defined two sampling-based heuristics named RD-B-GRAP and HD-B-GRAP in order to study the behavior of the propagation according to different quality measures related to the vertex and the edge balance, to the edge cut, and also to the propagation time. The results obtained on different graphs showed that the sampling-based algorithms improve the propagation time without affecting the balance between partitions. Moreover, The edge cut is slightly improved on some graphs.

CCS CONCEPTS

• **Computing methodologies** → **Parallel algorithms**;

KEYWORDS

Graph Partitioning, vertex balance, edge balance, graph sampling, parallel graph Computing, label propagation, Giraph.

ACM Reference Format:
Adnan El Moussawi, Ricardo Rojas Ruiz, and Nacéra Bennacer Seghouani. 2022. Sampling-based Label Propagation for Balanced Graph Partitioning. In *Proceedings of the 2022 ACM/SPEC International Conference on Performance Engineering (ICPE '22), April 9–13, 2022, Bejing, China.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3489525.3511698

1 INTRODUCTION

The demand and the need of powerful graph databases have increased in the last years due to their great capabilities for complex analysis. Therefore, distributed systems propose opportunities as well as huge challenges in order to improve the performance of big data analytics. By parallelizing both storage and execution, distributed systems achieve great improvements especially when clusters or machines work independently. However, inter-node communication in distributed graphs demands great execution time when the distribution of nodes and edges is not balanced [5]. In this context, the aim of graph partitioning is to identify an optimal partition approach which can balance the workload of the graph and minimize the edge-cuts thereby reducing the communication between machines. Different kind of algorithms for graph partitioning are proposed in the literature such as spectral, multilevel, stream, machine learning-based and label propagation partitioning approaches [1, 4, 6, 20, 22].

In this paper we focused on the label propagation approach as an extension of B-GRAP a multi-objective and scalable Balanced GRAph Partitioning (B-GRAP) algorithm, which distributes vertices B-GRAP$_{VB}$ or edges B-GRAP$_{EB}$ in a balanced way [9]. The main challenge is related to the seed nodes selection and to neighboring nodes that propagate labels that the stability, the convergence and the partitioning quality depend strongly on the structure of the graph. Sampling graph seems to be a promising way to select nodes based on their connections and degree to improve the performance of the partitioning and to reduce the computation time. We defined new sampling-based heuristics named RD-B-GRAP (Random Degree) and HD-B-GRAP (High Degree) in order to study the behavior of the propagation according to different quality measures related to the vertex and the edge balance, to the edge cut and also to the propagation time.

We used Giraph[1] programming model for Hadoop and ran the algorithms on different kind of large graphs of different structures with sizes going up to 42M vertices and 1.5B edges, by varying the number of partitions, the sampling parameters, and using several measures related to the partitioning balance quality and to the computation time. The results showed that the sampling-based algorithms reduce the label propagation time on all the graphs, with a gain going from 6% to 40% comparing to B-GRAP. Furthermore, the vertex and the edge balance of partitioning remained unchanged and stable on almost of graph data sets, while scaling the number of partitions. Moreover, The edge cut was slightly improved on some of the graphs and unchanged on the others.

In the following we give a review on the partitioning and the sampling approaches. In Section 3, we give some useful notations and preliminaries. Then, Section 4 details our sampling based label propagation algorithms. Section 5, presents the implementation environment and the experiment settings, followed by Section 6

[1] https://giraph.apache.org/

ICPE '22, April 9–13, 2022, Bejing, China

ACM ISBN 978-1-4503-9143-6/22/04…$15.00
https://doi.org/10.1145/3489525.3511698

which details our experiments and results. Finally, we conclude in Section 7.

2 RELATED WORK

During the last decade, research communities working on graph datasets have given a lot of interest to the definition of new strategies for large graph parallel computing and analytics in a distributed environment. This context opened up new challenges to define efficient graph partitioning algorithms [6, 14, 20]. One of the main challenges consists in defining methods that allow to balance the workload among the nodes of a distributed computing environment and to reduce, at the same time, the communication load over the network. In this section we discuss works about graph partitioning approaches, then we present briefly graph sampling purposes and applications.

Graph Partitioning. A common strategy in large graph partitioning is to use *multilevel approaches* [6]. Their main idea is to generate a first partition on the basis of a reduced view of the graph in which a vertex represents many vertices of the original graph. For example a triangle of three vertices can be reduced to one. The algorithms then expands the graph taking into account the whole initial graph, such as METIS [15] algorithm. Another known algorithm is Scotch [8] which deals with the graph changes, in contrary to METIS. *Streaming* approaches [24] process the graph iteratively. These methods are faster than multilevel algorithms but they build partitioning with lower quality, in term of cuts and it's generally difficult to parallelize streaming algorithms.
Other works have used the label propagation approach (LP) [21] to partition large graphs. LP was mainly used for community detection in social networks [7, 13]. Making use of LP for the graph partitioning problem was motivated by the lightweight mechanism that uses the network structure to guide its progress. LP partitioning methods generate less intermediary results than multilevel approaches, which need to store many intermediate results such as the coarser graph, and run with a lower complexity. Furthermore, LP method is semantic-aware, given the existence of local closely connected substructures, a label tends to propagate within such structures. In [19] the authors defined a distributed partitioning algorithm called Spinner that considers only edge balance. Spinner is based on LP approach and runs on the top of Giraph API.Compared to the previous work, Spinner supports parallelism and can adapt an existing partitioning to consider graph updates by adding or removing vertices and edges and changing the number of partitions. The algorithm divides N vertices across K partitions, while trying to keep similar the number of edges in each partition. In the same context other approaches have been proposed to take advantage from distributed computation and Map-Reduce programming paradigms. In [2] the authors embedded the nodes onto a line, and then processed them in a distributed manner guided by a linear embedding order. Their focus was on balanced-partitioning and on minimizing the total cut size. Recently, [9] proposed a multi-objective LP based partitioning algorithm B-GRAP. Comparing to previous methods, B-GRAP takes into account either edge-balance or vertex balance constraint. The authors showed experimentally that B-GRAP outperforms existing LP based partitioning approaches.

Graph sampling. A graph sample is a subset of vertices and/ or edges from original graph. The biggest advantage of sampling methods is their execution efficiency so that the graph transformation procedure won't take longer time than straightforward computation on original graph. It has a wide spectrum of applications, e.g. exploring, visualizing, scaling up analysis, etc. Commonly used techniques are vertex or edge sampling, traversal based sampling, substructure sampling to find patterns such as triangles and triads and streaming sampling [10–12]. Sampling graph has evolved during the last decade to more advanced graph exploration approaches such as Forest Fire and Frontier sampling and Random Walk algorithm variants.

In our work, we aim to investigate the sampling approaches for graph partitioning problem. Combining the sampling with the graph partitioning could help to improve the performance of the partitioning and to reduce the computation time. More particularly, in the case of LP based approaches, a sample of the graph can be used to initiate the label propagation process, instead of using the whole graph [19] or based on an heuristic [9]. In fact, a subgraph sample which takes into account the graph structure and selects relevant nodes allows to propagate efficiently and rapidly the labels through the whole graph. Furthermore, comparing to multilevel partitioning approaches that use a coarsened graph as an an entry for the partitioning, the sampling requires less computation time and resources, as it uses few intermediate results.

3 BACKGROUND AND PRELIMINARIES

3.1 Problem Formulation and Notations

The label propagation algorithm was defined in the context of community detection in social networks [7, 13]. This approach re-used in graph partitioning research context thanks to its lightweight and intuitive mechanism. Given a number of partitions of the graph, the naive LP algorithm simply works as follows: (i) At first, each vertex is assigned to a partition randomly; (ii) Then, the label of each vertex is propagated and updated iteratively to its neighborhood, where each vertex takes the most frequent label among its neighborhood as its own label. The process ends when labels no longer change. In the following we describe formally the LP algorithm.

Given a number of partitions K, a directed graph $G = \langle V, E, \omega \rangle$, where V is a set of vertices and E a set of weighted edges with $\omega : E \rightarrow \mathbb{R}^+$. Let $L = \{l\}_{l=1}^{K}$ be a set of partition labels defined by a labeling function $\phi : V \rightarrow L$ such that $\phi(v) = l$ means that v belongs to the partition with label l. The naïve LP algorithm proceeds as follows. Initially, a unique label l_v is assigned to each vertex v. Then, the label of each $v \in V$ is propagated and updated iteratively to its neighborhood $N(v) = \{u \in V | (v,u) \vee (u,v) \in E\}$ and is updated until a given convergence criteria is reached. The label updating is done by taking into account the most frequent label among $N(v)$ labels. More formally, let $\mathcal{F}_{\text{LP}}(v,l)$ be the frequency of a label l in the neighborhood of v, defined by:

$$\mathcal{F}_{\text{LP}}(v,l) = \sum_{u \in N(v)} \omega(v,u)\delta\big(\phi(u), l\big) \tag{1}$$

where $\phi(u)$ gives the current label of u and δ is the Kronecker delta function, which is equal 1 if $\phi(u) = l$, and 0 otherwise. The label

of vertex v is replaced by the label that maximizes the frequency function: $l_v = \text{argmax}_l \ \mathcal{F}_{\text{LP}}(v, l)$.

If many maximal labels exist and do not include the current label of v, one of them is randomly chosen. LP algorithm stops if $\sum_{v \in V} \sum_{l \in L} \mathcal{F}_{\text{LP}}(v, l)$ converges according to a given threshold ϵ.

We note that naïve LP algorithm does not take into account the directions of edges. To consider directed graphs, virtual edges are added [9], such that: $\forall (v, u) \in E, \omega(v, u) = 2$ and if $(u, v) \notin E$, (u, v) is added with $\omega(u, v) = 1$.

3.2 B-GRAP

B-GRAP [9] aims to define a K-balanced and LP-based partitioning algorithm that decreases the total cuts while considering the vertex balance or the edge balance constraints in directed graphs.

B-GRAP objective functions. The basic LP approach updates the labels without caring about balance, consequently the trivial optimal solution of Eq. (1) is assigning all vertices to a single label. To deal with this, the authors of B-GRAP have considired the balance constraint in the update function by adding a penalty term, which whenever the assignment of a vertex to a partition violates the balance constraint. The new update function is defined as follow.

$$\mathcal{F} = \mathcal{F}_{\text{LP}} + \lambda \mathcal{P} \tag{2}$$

$\mathcal{P}$ represents penalty terms and λ is a weight parameter. Two update functions $\mathcal{P}_{\text{VB}}$ and $\mathcal{P}_{\text{EB}}$ were defined to respectively deal with vertex and edge balance constraints:

$$\mathcal{P}_{\text{VB}}(l) = \frac{1}{K} - \frac{size(V, l)}{|V|} \tag{3}$$

This function measures the divergence between the perfect balance ratio and the ratio of vertices with label l.

$$\mathcal{P}_{\text{EB}}(l) = \frac{1}{K} - \frac{size(E, l)}{|E|} \tag{4}$$

$\mathcal{P}_{\text{EB}}$ discourages a vertex to move to a partition with l label, when the ratio of edges in the partition l is closer or larger than the balance factor. Note that comparing to vertex balance, edge balance minimizing the edge cuts implicitly by maximizing the edge locality in each partition.

The algorithm 1 describes the main procedures of B-GRAP: initialisation and label propagation. To initialize B-GRAP, the authors considered only *hub* vertices having a high outgoing degree $d^+(.)$. Their intuition is that the higher $d^+(v)$, the more $\phi(v)$ will be propagated and considered as frequent label. They defined their algorithm 1 as following: Given a directed graph $G = \langle V, E, \omega \rangle$, d_{inf} denotes the minimum out degree threshold to consider that a vertex v as a *hub* vertex. The algorithm proceeds as follows. First, the set of labels L is initialized according to the input number of partitions K value (Line 1). Then, each $v \in V$, such $d^+(v) > d_{inf}$ is randomly assigned a label $\in L$ and those labels are propagated to neighbors (Line 2). Then, the label of these neighbors are updated and propagated iteratively using an update function (Lines 3-7). The vertices are then checked and those not reached by the update/propagation step are initialized randomly, to ensure that all vertices are assigned a label (Lines 8-9). The algorithm repeats the update/propagate step (Line 10-12) until convergence (Line 13).

Algorithm 1 B-GRAP

Input: $G = \langle V, E, w \rangle, K, \epsilon, d_{inf}$
Output: a partitioned graph $G = \langle V, E, w, \phi \rangle$
1: $L = \{l\}_{l=1}^{K}$
2: **for** $\{v \in \{v \in V, d^+(v) > d_{inf}\}\}$ **do**
3: $\phi(v) = random(L)$ and propagate to $N(v)$
4: **end for**
5: **while** $\Delta(\mathcal{F}_{\text{LP}}(G, L)) \leq \epsilon$ **do**
6: **for** $(v \in V, l \in L)$ **do**
7: get the set of frequent labels w.r.t an update function
8: **end for**
9: Update and propagate $\phi(v)$ to $N(v)$
10: **for** $(v \in V,\ \phi(v) = null)$ **do**
11: initialize $\phi(v)$ randomly from L and propagate to $N(v)$
12: **end for**
13: **end while**
14: **return** $G = \langle V, E, w, \{\phi(v)\}_{v \in V} \rangle$

Algorithm 2 Sampling-based heuristics

Input: $G = \langle V, E, w \rangle, K, \epsilon, \tau, \beta, \sigma$
Output: a partitioned graph $G = \langle V, E, \omega, \phi \rangle$
1: Compute σ Seeds
2: Compute Sample and propagate (see Algorithm 3)
3: Update and propagate as in Algorithm (1) Lines 5-15
4: **return** $G = \langle V, E, w, \{\phi(v)\}_{v \in V} \rangle$

4 SAMPLING-BASED LABEL PROPAGATION ALGORITHMS

4.1 Main procedures

One of the biggest problems in label propagation based partitioning algorithms is to deal with graphs of different structures. The initialization step affects both the partitioning quality and the execution time. Our purpose is to study how we can introduce a graph sampling in initialization step in order to deal with these issues. [25] proposed a graph Rank Degree sampling method using a edge selection rule based on a node ranking. In order to define our initialization heuristics, we identify two main steps: (i) seed vertices selection to retrieve the initial σ vertices; (ii) sampling to take β vertices from the neighbors of selected seeds in order to initiate the LP process. In the second step, only τ vertices are selected from the neighbors of a seed at each iteration, this allows to reduce the impact of seeds with a high degree (hub vertices) on the sampling. In fact, without τ parameter, most of the sampled vertices will be selected from the neighborhood of the hub seeds. The Figure 1 shows an overview on the sampling-based HD-B-GRAP and RD-B-GRAP heuristics and summarizes the two steps comparing to B-GRAP.

The algorithm 2 describes the main procedures of sampling based heuristics for seed selection (Line 1), neighbor sampling (Line 2, $initializeSample()$ function) and label updates and propagation according to B-GRAP (Line 3). The function $initializeSample()$ is detailed in Algorithm 3. It is an iterative function that will continue to execute itself until the stopping condition is met. In our case, the stopping condition is when the sample size has been reached and all partition sets have been initialized or in formal terms $|V_S| \geq \beta$ and $|P_l| \geq 1, \forall l \in L$, where V_S is the sampled vertices from the graph and P_l is the l-th partition.

In the case that a partition set is empty, the algorithm will attempt to initialize it and to balance all partition sets at the same time. Thus, it will move a vertex from one partition with excess nodes (i.e. $|P_k| > |G_S|/K$) to the uninitialized partition (Lines 11-14). This procedure will iterate until both stopping conditions are true.

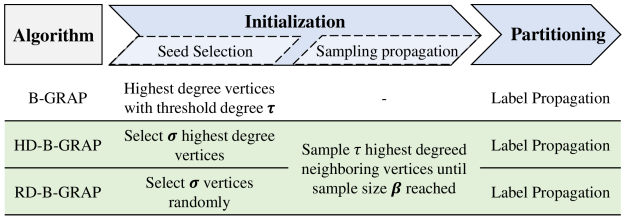

Figure 1: Sampling-based B-GRAP algorithms

4.2 RD-B-GRAP algorithm

RD-B-GRAP algorithm obtains a neighbor according to its rank in descending order (omitting previously sampled vertices) and selects τ neighbors to be seeds in the next iteration. A complete execution of the RD-B-GRAP is as follows (Algorithms 2 and 3).

In the seed selection sub-step, a seed will be selected as a random uniform sample of the nodes. These seeds will later propagate in the sampling propagation sub-step as follows: i) each vertex in the seed will request the degree to all of its neighbors; ii) the same vertex, in the following superstep, will receive the answering messages from un-sampled neighbors and will only store those from the τ highest degree neighbors; finally, iii) the vertex will notify the τ highest degree neighbors that they have been sampled and will be seed for the next iteration. The sampling propagation sub-step will execute until the sample size is reached. Finally, the algorithm moves to the label propagation step and returns the partitioning when it converges.

Algorithm 3 RD-B-GRAP and HD-B-GRAP initializeSample()

Input: $G, Seeds, K, \tau, \beta$
Output: Sampled vertex sets with labels V_S
1: $L \leftarrow \{l\}_{l=1}^{K}$
2: $V_S \leftarrow \emptyset$
3: $\{P_l \leftarrow \emptyset\}_{l=1}^{K}$
4: **while** $|V_S| \geq \beta$ and $|P_l| \geq 1, \forall l \in L$ **do**
5: $C \leftarrow \emptyset$, candidate set, $S \leftarrow \emptyset$, sampled in superstep set
6: **for** $s \in Seeds$ **do**
7: find unvisited neighbors $N(s)$ degrees, rank them in descending order
8: $C \leftarrow C \cup$ first τ vertices from ranked $N(s)$
9: **for** $c \in C$ **do**
10: generate random number r
11: **if** $r \leq \frac{\beta - |V_S|}{|C|}$ **then**
12: initialize $\phi(c)$ randomly from L
13: $P_{\phi(c)} \leftarrow P_{\phi(c)} \cup \{c\}$
14: $S \leftarrow S \cup \{c\}$
15: **end if**
16: **end for**
17: **end for**
18: $V_S \leftarrow V_S \cup S$
19: $Seeds \leftarrow S$
20: **end while**
21: **return** $\langle V_S, \{\phi(v)\}_{v \in V_S} \rangle$

In the real execution, each of the sampled vertices would be assigned a label and aggregated into the counter of the label. Before the algorithm finishes, it would have to verify that all partition sets have a minimum of 1 vertex inside. If it is not the case, it will relabel vertices from overpopulated sets. Now we will go on to see how the HD-B-GRAP differs from this algorithm.

4.3 HD-B-GRAP algorithm

RD-B-GRAP and HD-B-GRAP are actually quite similar, the difference lies in the seed initialization sub-step, as we could see in Figure 1. We note that this sampling technique will result in a sample with a higher bias to high degree vertices, which could lead to a reduced Label Propagation execution time.

A complete execution of the HD-B-GRAP is as follows (Algorithms 2 and 3). First we select the highest degree σ vertices. In the seed selection sub-step, we first generate the degree distribution of the entire graph and then select the vertices with the highest degree. These seeds will later propagate in the sampling propagation sub-step as follows: i) each vertex in the seed will request the degree to all of its neighbors; ii) the same vertex, in the following superstep, will receive the answering messages from un-sampled neighbors and will only store those from the τ highest degree neighbors; finally, iii) the vertex will notify the τ highest degree neighbors that they have been sampled and will be seed for the next iteration. The sampling propagation sub-step will execute until the sample size is reached. Finally, the algorithm moves to the label propagation step and returns the partitioning when it converges.

5 EXPERIMENT SETTINGS

We achieved different experiments on different graph data sets in order to study the impact of the sampling on the LP partitioning. We specifically evaluated the edge-cut quality, the balance quality and the LP computation time of HD-B-GRAP and RD-B-GRAP comparing to B-GRAP. All the experiments are done on a single-node Hadoop cluster. The machine has 80 compute cores and 1.5Tera RAM, but 64 cores and 786GB RAM are used by Hadoop.

Data sets description In our experiments, we use ten graph data sets of different degree distributions and different sizes. LastFM (LF), Wikitalk (W), Pockec (P), Flixster (F), LiveJournal (LJ), Orkut (O) and Twitter (T) are social online networks graphs. BerkeleyStanf (B) is the berkely.edu and stanford.edu web graph. DelaunaySC (D) and Graph500 (G) are synthetic graphs. Notice that only (G) is an undirected graph. The table Table 1 summarises for each graph data set the edge and vertex numbers, the percentages of vertices with at most one degree value and at least 100 degree value and also the number of vertices with the maximum degree value.

We evaluate our algorithm over all the graphs presented in Table 1, by varying the number of partitions $K \in \{2^i\}_{i=1}^{6}$. We execute 10 runs of each algorithm for each graph and each value of K to ensure the significance of the results. For all experiments, we compute the average variation of the following measures with respect to the number of partitions K and over the runs:
The maximum normalized unbalance of vertices ($\mathbf{MNU}_{\mathrm{VB}}$) and of edges ($\mathbf{MNU}_{\mathrm{EB}}$): this metric is used to measure the unbalance and represents the percentage-wise difference of only the largest partition from a perfectly balanced partition.

$$\mathbf{MNU}_{\mathrm{VB}} = \frac{\max(|V_l|)}{|V|/K}, \quad \mathbf{MNU}_{\mathrm{EB}} = \frac{\max(|E_l|)}{|E|/K}, \quad \text{with } l \in L.$$

The edge-cuts ratio (**EC**): the ratio of edges connecting each two vertices in two different partitions w.r.t the total number of edges.
The computation time (**Time**) required to complete the label propagation processing.

We note that to compute **EC** and **MNU** only the original input edges of the graph are considered. For all experiments, we set $\epsilon = 10^{-3}$ as a threshold stop value and we set τ and the out degree

Table 1: Data sets description

Graph	LastFM	WikiTalk	BerkeleyStanf	Flixster	DelaunaySC	LiveJournal	Orkut	Graph500	Twitter
Directed	yes	yes	yes	yes	yes	yes	yes	no	yes
$\|V\|$	1.2	2.4M	0.7M	2.5M	8.4M	4.8M	2.7M	4.6M	41.7M
$\|E\|$	4.5	5M	7.6M	7.9M	25.2M	69M	117.2M	258.5M	1.5B
$\|\{v\}_{d^+(v)\leq 1}\%\|$	49.6%	97.2%	11.0%	60.2%	25.8%	30.0%	16.7%	23.6%	10.5%
$\|\{v\}_{d^+(v)\geq 100}\%\|$	0.1%	0.3%	0.9%	0.1%	0.0%	2.0%	8.6%	8.5%	3.2%
Max degree	1.1k	3k	84k	1.5k	28	22.9k	33.3K	544k	4.9M
Source	[23]	[17]	[18]	[23, 26]	[4, 23]	[3]	[23]	[23]	[16]

average $\bar{d}^+ = \frac{|E|}{|V|}$[2]. The penalty term weight parameter λ in the update function $\mathcal{F}$ is set to 1. This gives an equal importance to the penalty term $\mathcal{P}$ and to $\mathcal{F}_{\text{LP}}$ according to the update functions defined in Section 3.2. For HD-B-GRAP and RD-B-GRAP, we set $\beta = 0.2, \sigma = 0.1 \times \beta$ and $\tau = 2$. We note that the choice of these values is studied using several graphs and configurations, for more details see Section 6.3.

6 RESULTS ANALYSIS

In the following, we firstly present the results of running the sampling based partitioning algorithms HD-B-GRAP and RD-B-GRAP comparing to B-GRAP using vertex and edge objective functions, and also to the state art results. Then we the study about β, δ and τ parameters robustness.

6.1 Vertex-balance evaluation

In this experiments we compare the results obtained with HD-B-GRAP, RD-B-GRAP and B-GRAP using the vertex balance constraints. We study the impact of each sampling approach on the vertex balance quality, on the edge-cut quality and on the LP computation time.

Figure 2 shows that the sampling in the most of the graphs doesn't impact the balance quality and $\mathbf{MNU}_{\text{VB}}$ is similar between all the algorithms, expect for Wikitalk and BerkeleyStanf and $K \geq 32$: the quality of RD-B-GRAP decreases slightly on Wikitalk and poorly on BerkeleyStanf, HD-B-GRAP performs poorly on both graphs. The results are slightly better than B-GRAP on LastFM and Flixster graphs with an improvement rate equals $\approx 1\%$.

The quality of the edge-cut remains stable with very good values. We can notice a high improvement of **EC** for HD-B-GRAP on WikiTalk, that counteracts the $\mathbf{MNU}_{\text{VB}}$ distortion, and a slight improvement on some other graphs. The total average improvement percent on each graph and for all values of K is summarized as follow: (i) HD-B-GRAP: 25% on WikiTalk, $4 \sim 5\%$ on BerkeleyStanf and Flixster, $0 \sim 1.5\%$ on other graphs, (ii) RD-B-GRAP: $4 \sim 5.5\%$ on WikiTalk, BerkeleyStanf and on Flixster, $0 \sim 1.3\%$ on other graphs.

Regarding the LP computation time, both sampling based initialization outperform the original B-GRAP. The improvement varies between 6.3% and 40.7% for HD-B-GRAP and between 6.9% and 38.6% for RD-B-GRAP. In the case of a large graph, such as Twitter, the total average improvement of LP computation time is 12.7% ($\approx$ 390 seconds) for HD-B-GRAP and 11.9% ($\approx$ 370 seconds). The average improvements percent for our sampling based algorithms comparing to B-GRAP for each graphs and w.r.t. to $\mathbf{MNU}_{\text{VB}}$, **EC** and LP time are given in Table 2. . In this table we emphasis the values higher than 5% and we mark in red color the worst values ($\leq -5\%$).

[2]The choice of this value was studied in [9]

Table 2: The percent of quality (EC, $\mathbf{MNU}_{\text{VB}}$) and performance (LP time) improvements of HD-B-GRAP and RD-B-GRAP w.r.t. B-GRAP and using the vertex balance.

Vertex Balance	HD-B-GRAP (%)			RD-B-GRAP (%)		
Graph	$\mathbf{MNU}_{\text{VB}}$	**EC**	**LP Time**	$\mathbf{MNU}_{\text{VB}}$	**EC**	**LP Time**
LastFM	0.8	0.0	**39.2**	0.9	0.0	**39.8**
WikiTalk	-25.1	**24.4**	**32.8**	-5.5	**5.4**	**12.6**
BerkeleyStanf	-29.5	4.6	**40.7**	-36.0	**5.3**	**38.6**
Flixster	1.5	4.3	**42.3**	1.0	4.0	**32.6**
DelaunaySC	0.0	1.5	**18.6**	0.0	1.2	**10.5**
LiveJournal	-0.1	0.0	**11.1**	-0.1	-0.1	**13.6**
Orkut	0.1	0.4	**7.2**	-0.1	0.4	**6.9**
Graph500	0.1	0.6	**6.3**	0.2	0.8	**9.5**
Twitter	0.0	1.3	**12.6**	0.1	1.3	**11.9**

6.2 Edge-balance evaluation

The results on balance quality and edge quality are similar to the vertex balance case, similarly for the LP computation time, which was improved over all graph data sets. In particular, the results show a high improvement on the two largest graphs Graph500 and Twitter with a percent of gain equals respectively to 40.3% and to 18.7%.

Comparison with the state of the art methods In the following, we compare the results obtained with our proposed solutions on Twitter graph to the results reported in [19]. Note that the authors has evaluated only the edge balance quality, as they deal only with this constraint. They used the ratio of local edges ρ to evaluate the partitioning quality, which equals to the number of edges connecting two vertices belonging to the same partition divided by the total number of edges. This metric is inversely proportional to the ratio of edge-cuts. The computation time wasn't considered in this experiment, as the algorithms was executed in different environments.

The table 3 summarizes the results obtained on Twitter graph with the following algorithms: B-GRAP, HD-B-GRAP, RD-B-GRAP, Spinner [19], Fennel [24] and Metis [15].

B-GRAP performs generally better HD-B-GRAP and HD-B-GRAP w.r.t. to the local edge quality, and has similar balance quality w.r.t. to the $\mathbf{MNU}_{\text{EB}}$. We assume that the deterioration of the cut quality is due to the presence of many interconnected communities in the Twitter graph, that are difficult to separate, which makes it hard to obtain a good sampling. In particular, the seed selection method of HD-B-GRAP, based on highest degrees, can lead to the selection of some interconnected hub vertices which share a common neighbors.

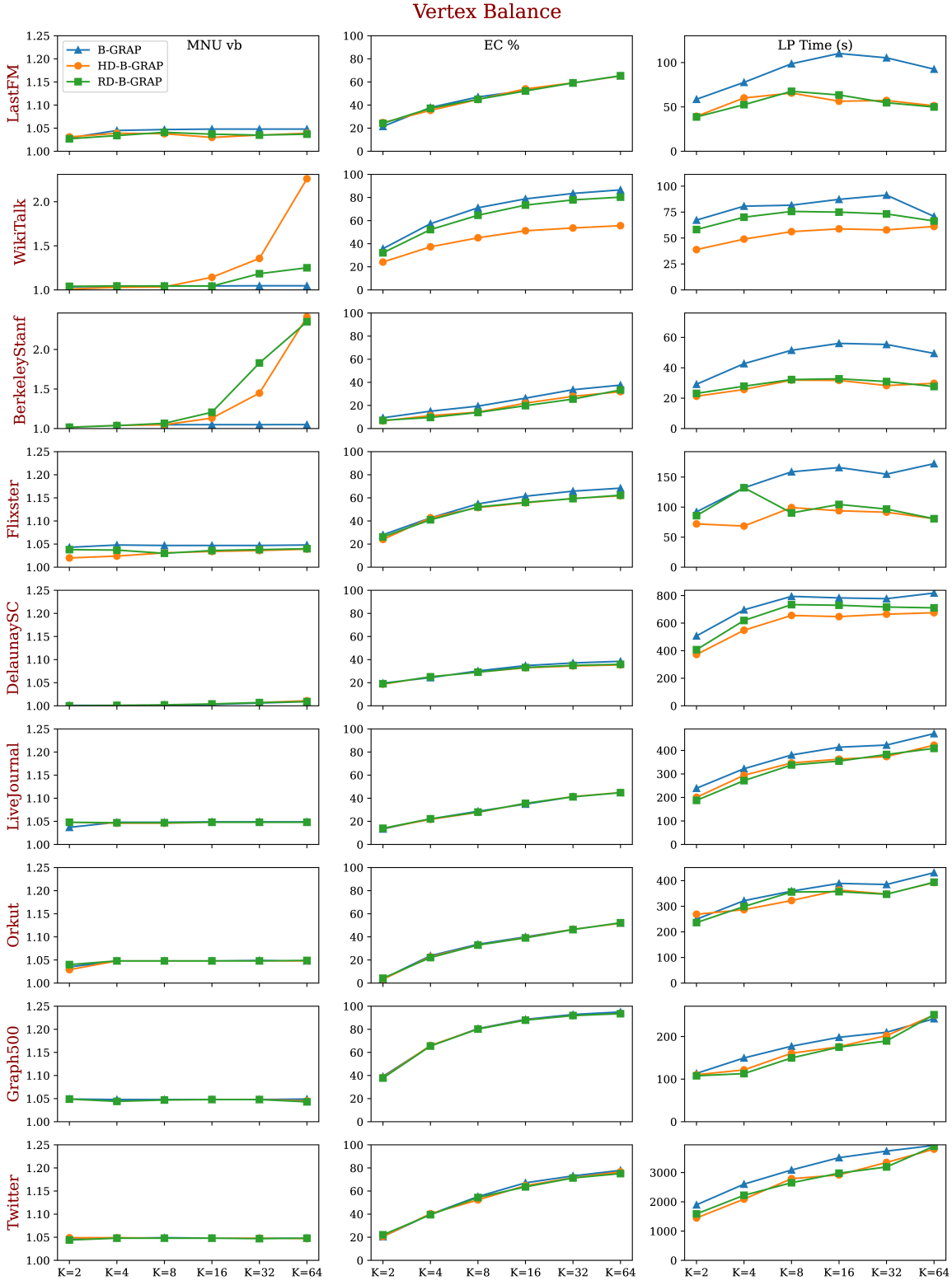

Figure 2: Variation of the average scores of MNU_{VB}, EC and Time for the partitioning obtained with B-GRAP, HD-B-GRAP and RD-B-GRAP, w.r.t. K and with the vertex balance constraint.

Comparing to other algorithms, B-GRAP has a slightly better locality quality than Spinner [19], which is based on the LP approach. Fennel performs similarly to B-GRAP with respect to ρ, except on $k = 2$ the quality is the best, it performs poorly with respect to $\mathbf{MNU}_{\mathrm{EB}}$ comparing to all other algorithms. Metis [15] remains the best approach w.r.t. to quality of the partitioning and the balance. However, we remind that Metis has a high computation cost.

6.3 Parameters robustness

In order to study the robustness of our sampling based partitioning algorithms HD-B-GRAP and RD-B-GRAP to the sampling parameters, we defined two different main tests. Firstly, we studied the impact of the sizes of the sample and of the seeds. Secondly, we studied the impact of τ, the neighboring nodes set size. Next we explain our intuition for each experiment and we analyze the results.

To achieve these experiments, we used Twitter graph and we fixed the number of partitions K to 4. For the first experiment, the sample size $\beta \in 10\%, 15\%, 20\%, 25\% \times |V|$, with $\sigma = \beta/10$ and τ fixed to 5. Finally, for each value of β, we made 10 runs of each algorithm and we compute $\mathbf{MNU}_{\mathrm{VB}}$, **EC**, the sampling computation time and the LP computation time. In the second experiment, $\tau \in \{5, 25, 50, 100\}$ and we fix the values of β and σ respectively to 15% and 1.5% of the vertices number, then we proceed similarly to compute the evaluation measures. The results that we have obtained for $\mathbf{MNU}_{\mathrm{VB}}$ were similar for the different configurations. Therefore, they are not relevant to study the robustness.

Impact of the sizes of seeds and sample: Our initial hypothesis in regard to this parameter is that as β value increases, the algorithm should take more time to execute the sampling and less time to execute the label propagation. Conversely, as σ increases its value, we expect that the time required for the sampling will be reduced, while the expected behavior on the label propagation is that it will take less time to converge. The overall effect of changing both β and σ (since we defined σ to be 10% of β), should lead to a reduced time because σ grows in terms of τ. In terms of the quality of partitioning, we needed to show how it will be affect.

From the results shown in Figure 3, we can see that our hypothesis is validated. It shows some slight variations in the edge-cut **EC**, but the standard deviations of **EC** overlap, thus we cannot conclude that there is an improvement in this regard. Conversely, we can see that there is a clear improvement in execution time, both in terms of the sampling and the label propagation steps. This execution time decreasing can be obtained since $\beta = 15\%$ and $\sigma = 1.5\%$.

Impact of the neighboring nodes set size: Regarding this parameter, our initial hypothesis in terms of time is that this might be able to speed up the sampling and that in has no inference in the label propagation. However, it doesn't require to have a high value in order to speed it up since the expected growth will be capped at a maximum of $|S| * \tau$ vertices sampled per superstep, leading to exponential growth over iterations. We must keep in mind, nevertheless, that the size of the seed varies through iterations and in some cases it might contract in comparison to a previous superstep. Note that for this test, we set $\beta = 15\%$ and $\sigma = 1.5\%$, since they were the best configuration obtained in the previous experiment.

From the results shown in Figure 4, we can confirm that the algorithms are not sensitive to the value of τ. Since all metrics overlap the standard deviation, we cannot say that the results are better nor worse. Thus, we can say that a value of 5 for τ is as good as having a value of 100.

7 CONCLUSIONS AND PERSPECTIVES

In this paper, we defined and studied HD-B-GRAP and RD-B-GRAP algorithms that combine graph sampling and label propagation for graph balanced partitioning, i.e. vertex or edge balance constraint.

To conclude, our algorithms have in general better edge-cut quality than the original B-GRAP algorithm and this gain comes at a trade off for balance. This unbalance is not really noteworthy in medium to large data sets, but on small data sets it is significant. We also see that our algorithms tend to be slower since the sampling stage has been introduced, however, once the sampling is achieved, there are significant improvements on the label propagation stage. In few cases, this gain was enough to counteract the time added for the sampling. In vertex-balance case, regarding the LP convergence, both algorithms have reduced computation time over all the graphs by 6% to 40%. The conclusions in edge-balance case, on either balance quality or edge quality are similar. We notice a high improvement on the two largest graphs Graph500 and Twitter with a percent of gain equals respectively to 40.3% and to 18.7%. The comparison with some exiting edge-balanced partitioned methods on Twitter graph have showed a slight deterioration in **EC**, especially for HD-B-GRAP. For this graph, we think that presence of many interconnected communities impacts the quality of the sampling. In terms of parameter robustness, changing the sample size β and, consequently the seeds size σ could lead to a reduced execution time in both sampling and label propagation stages. In the case of the Twitter graph, the improvement can be achieved when $\beta = 15\%$ and $\sigma = 1.5\%$. For the seed neighboring nodes set size parameter τ, it was difficult to conclude for Twitter data set.

There are many perspectives that we can explore. First of all, we would like to study if an optimal configuration for the hyperparameters exists and or if it depends on the graph topology. In the same way, we have been able to prove that graph sampling can improve the performance of label propagation algorithms, it would be interesting to extend this work to other existing approaches of graph partitioning on which the initialization is performed at random, such as multilevel algorithms. Finally, since label propagation is originally a community detection algorithm, we could explore sampling based algorithms to try to tackle this open problem.

REFERENCES

[1] Avdiukhin, D., Pupyrev, S., and Yaroslavtsev, G. Multi-dimensional balanced graph partitioning via projected gradient descent. *Proceedings of the VLDB Endowment 12* (04 2019), 906–919.

[2] Aydin, K., Bateni, M., and Mirrokni, V. Distributed balanced partitioning via linear embedding. *Algorithms 12*, 8 (2019).

[3] Backstrom, L., Huttenlocher, D., Kleinberg, J., and Lan, X. Group formation in large social networks. In *Proc. of the SIGKDD Inter. Conf. on Knowledge discovery and data mining* (2006), ACM Press.

[4] Bader, D. A., Meyerhenke, H., Sanders, P., and Wagner, D. *Graph Partitioning and Graph Clustering*, vol. 588. American Mathematical Society Providence, RI, 2013.

[5] Buluç, A., and Madduri, K. Graph partitioning for scalable distributed graph computations. In *Graph Partitioning and Graph Clustering, 10th DIMACS Implementation Challenge Workshop* (2012), vol. 588, American Mathematical Soc.,

Table 3: Comparison of the state of the art algorithms for Twitter graph while scaling the number of desired partitions and using the edge balance constraint. The table reports the ratio of local edges ρ and the MNU_{EB}.

	$k = 2$		$k = 4$		$k = 8$		$k = 16$		$k = 32$	
Approach	ρ	MNU_{EB}	ρ	MNU_{EB}	ρ	MNU_{EB}	ρ	MNU_{EB}	ρ	MNU_{EB}
HD-B-GRAP	0.84	1.02	0.66	1.03	0.52	1.05	0.30	1.10	0.29	1.07
RD-B-GRAP	0.80	1.02	0.69	1.04	0.46	1.05	0.32	1.06	0.34	1.14
B-GRAP	0.85	1.01	0.70	1.04	0.52	1.05	0.42	1.06	0.33	1.07
Spinner [19]	0.85	1.05	0.69	1.02	0.51	1.05	0.39	1.04	0.31	1.04
Fennel [24]	0.93	1.10	0.71	1.10	0.52	1.10	0.41	1.10	0.33	1.10
Metis [15]	0.88	1.02	0.76	1.03	0.64	1.03	0.46	1.03	0.37	1.03

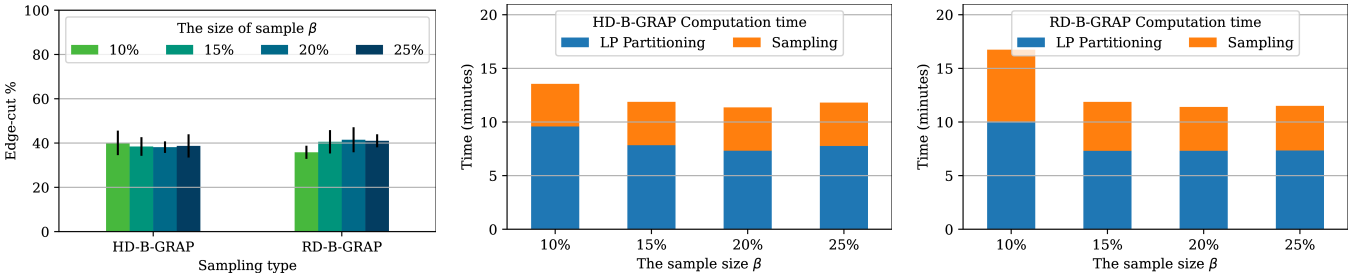

Figure 3: Robustness of HD-B-GRAP and RD-B-GRAP to the sample size: Average scores of MNU_{VB}, EC and execution time obtained on Twitter graph, with $K = 4$, and while varying parameters β and σ

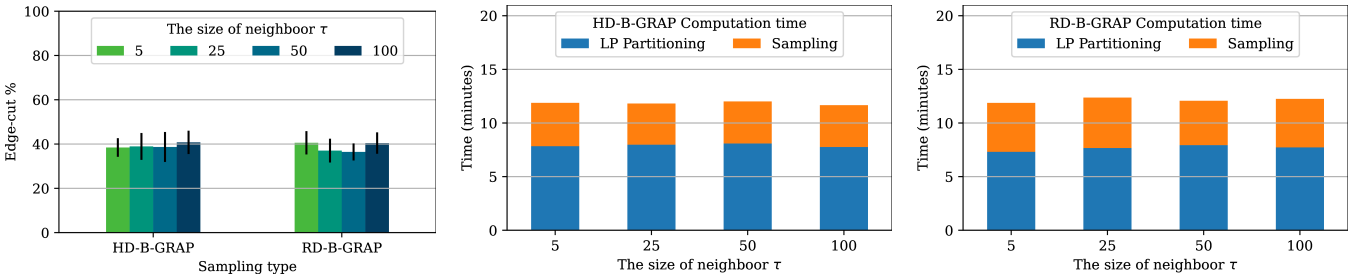

Figure 4: Robustness of HD-B-GRAP and RD-B-GRAP to the neighbor size τ: Average scores of MNU_{VB}, EC and execution time obtained on Twitter graph, with $K = 4$, and while varying parameter τ

pp. 83–102.

[6] Buluç, A., Meyerhenke, H., Safro, I., Sanders, P., and Schulz, C. *Recent Advances in Graph Partitioning.* Springer, 2016, pp. 117–158.

[7] Chakraborty, T., Dalmia, A., Mukherjee, A., and Ganguly, N. Metrics for community analysis: A survey. *ACM Comput. Surv. 50*, 4 (2016), 1–37.

[8] Chevalier, C., and Pellegrini, F. Pt-scotch: A tool for efficient parallel graph ordering. *Parallel Computing 34*, 6 (2008), 318–331.

[9] El Moussawi, A., Bennacer Seghouani, N., and Bugiotti, F. A graph partitioning algorithm for edge or vertex balance. In *Database and Expert Systems Applications (Dexa)* (2020), vol. 12391, pp. 23–37.

[10] Frank, O. Social network analysis, estimation and sampling in. In *Encyclopedia of Complexity and Systems Science*, R. A. Meyers, Ed. Springer, 2009, pp. 8213–8231.

[11] Frank, O. Network sampling. In *Inter. Encyclopedia of Statistical Science*, M. Lovric, Ed. Springer, 2011, pp. 941–942.

[12] Frank, O., and Shafie, T. Random multigraphs and aggregated triads with fixed degrees. *Netw. Sci. 6*, 2 (2018), 232–250.

[13] Gregory, S. Finding overlapping communities in networks by label propagation. *New Journal of Physics 12*, 10 (oct 2010), 103018.

[14] Heidari, S., Simmhan, Y., N. Calheiros, R., and Buyya, R. Scalable graph processing frameworks: A taxonomy and open challenges. *ACM Computing Surveys 51* (2018), 1–53.

[15] Karypis, G., and Kumar, V. Multilevel graph partitioning schemes. In *Proc. of the 24th Inter. Conf. on Parallel Processing (ICPP)* (1995), vol. 3, pp. 113–122.

[16] Kwak, H., Lee, C., Park, H., and Moon, S. What is Twitter, a social network or a news media? In *Proc. of the 19th Inter. Conf. on World wide web (WWW)* (2010), ACM, pp. 591–600.

[17] Leskovec, J., Huttenlocher, D., and Kleinberg, J. Signed networks in social media. In *Proc. of the 28th Inter. Conf. on Human factors in computing systems* (2010), p. 1361.

[18] Leskovec, J., Lang, K. J., Dasgupta, A., and Mahoney, M. W. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics 6*, 1 (2009), 29–123.

[19] Martella, C., Logothetis, D., Loukas, A., and Siganos, G. Spinner: Scalable graph partitioning in the cloud. In *Proc. - Int. Conf. Data Engineering* (2017).

[20] Meyerhenke, H., Sanders, P., and Schulz, C. Parallel graph partitioning for complex networks. *IEEE Trans. on Parallel and Distributed Systems 28*, 9 (2017), 2625–2638.

[21] Raghavan, U. N., Albert, R., and Kumara, S. Near linear time algorithm to detect community structures in large-scale networks. *Physical review. E, Statistical, nonlinear, and soft matter physics* (2007), 036106.

[22] Rais, H., Abed, S., and Watada, J. Computational comparison of major proposed methods for graph partitioning problem. *Journal of Advanced Computational Intelligence and Intelligent Informatics 23* (01 2019), 5–17.

[23] Rossi, R. A., and Ahmed, N. K. The network data repository with interactive graph analytics and visualization. In *Proc. of the 29 AAAI* (2015).

[24] Tsourakakis, C., Gkantsidis, C., Radunovic, B., and Vojnovic, M. FENNEL: Streaming graph partitioning for massive scale graphs. In *Proc. of the 7th ACM Inter. Conf. on Web search and data mining* (2014), pp. 333–342.

[25] Voudigari, E., Salamanos, N., Papageorgiou, T., and Yannakoudakis, E. J. Rank degree: An efficient algorithm for graph sampling. In *Inter. Conf. on Advances in Social Networks Analysis and Mining, ASONAM* (2016), IEEE Computer Society, pp. 120–129.

[26] Zafarani, R., and Liu, H. Users joining multiple sites: Distributions and patterns. In *8th Inter. AAAI Conf. on Weblogs and Social Media* (2014).