Balanced Graph Edge Partition

Florian Bourse*
ENS
Paris, France
florian.bourse@ens.fr

Marc Lelarge
INRIA-ENS
Paris, France
marc.lelarge@ens.fr

Milan Vojnović
Microsoft Research
Cambridge, UK
milanv@microsoft.com

ABSTRACT

Balanced edge partition has emerged as a new approach to partition an input graph data for the purpose of scaling out parallel computations, which is of interest for several modern data analytics computation platforms, including platforms for iterative computations, machine learning problems, and graph databases. This new approach stands in a stark contrast to the traditional approach of balanced vertex partition, where for given number of partitions, the problem is to minimize the number of edges cut subject to balancing the vertex cardinality of partitions.

In this paper, we first characterize the expected costs of vertex and edge partitions with and without aggregation of messages, for the commonly deployed policy of placing a vertex or an edge uniformly at random to one of the partitions. We then obtain the first approximation algorithms for the balanced edge-partition problem which for the case of no aggregation matches the best known approximation ratio for the balanced vertex-partition problem, and show that this remains to hold for the case with aggregation up to factor that is equal to the maximum in-degree of a vertex. We report results of an extensive empirical evaluation on a set of real-world graphs, which quantifies the benefits of edge- vs. vertex-partition, and demonstrates efficiency of natural greedy online assignments for the balanced edge-partition problem with and with no aggregation.

Categories and Subject Descriptors

E.1 Data [**Data Structures**]: Graphs and networks; G.2.2 [**Mathematics of Computing**]: Discrete Mathematics–Graph Theory, Graph Algorithms; H.2 [**Information Systems**]: Database Management

General Terms

Algorithms, Experiments, Performance

*Work performed as part of a MSR-INRIA joint research centre project.

KDD'14, August 24–27, 2014, New York, NY, USA.

http://dx.doi.org/10.1145/2623330.2623660.

Keywords

Graph Edge Partition, Distributed Massive Computations, Approximation Algorithms, Streaming Heuristics

1. INTRODUCTION

A common approach to scale-out computations on large-scale input graph data is to partition the graph data and parallelize the computation on some given number of machines in a distributed cluster of machines. For example, this is of interest for iterative computations on input graph data such as computing the PageRank using the power-iteration method in systems such as Pregel [23], Giraph [8] and Spark [39], or for iterative machine learning computations such as approximate Bayesian inference in systems such as Graphlab [22, 21], or for efficiently resolving queries in large-scale databases, e.g. [3], and specifically, graph databases, e.g., Neo4j [24] and Trinity [30]. An important problem is to ensure a high-quality of a graph partition with respect to two criteria: balancing the processing load across machines and minimizing the communication cost between different machines.

A traditional approach to placing a graph on a cluster of k machines is to construct a balanced *vertex partition* that equi-partitions the set of vertices to k partitions such that the number of edges spanning the partitions is minimized. A new approach was proposed recently in [9] that advocates to perform an *edge partition* by equi-partitioning the set of edges to k partitions such that the amount of communication required to synchronize the state of vertex-copies is minimized. This new approach is motivated by the observation that standard tools, e.g., [14, 26] for constructing a balanced vertex partition perform poorly on power-law graphs [1, 19, 20]. Each cut edge contributes to storage and network overhead since both machines maintain a copy of the adjacency information, and in some cases, a ghost vertex per each cut edge is maintained [10]. Many systems resort to using hash (random) vertex placement for its simplicity of implementation. Since the graph partitioning problem is considered to be intrinsically difficult, this random partition may perform rather poorly. The design rationale of performing an edge partition is to allow for more flexibility for processing load balancing by allowing a single vertex program to span multiple machines and reduction of communication and storage overhead by evenly assigning edges to machines. Since each edge is stored exactly once, changes to edge data do not need to be communicated.

Little is known about the balanced edge-partition problem both on the grounds of theoretical computation complexity

and approximation guarantees, and practical methods with good average-case performance. This is unlike to the traditional balanced vertex-partition problem which was studied rather extensively, with the best known approximation guarantee of $O(\sqrt{\log k \log n})$ for the edge-cut cost of a partition in k clusters of a size n graph [18], and with software tools being available off-the-shelf, e.g. METIS [12, 13]. An important requirement for graph partitioning at scale is to being able to produce a good-quality graph partition under restriction to make a single pass through the graph data, which is often referred to as *streaming graph partition*. Here, again, we find a number of approaches that have been studied for the vertex-partition problem, e.g. [32, 34, 25]. On the contrary, the study of streaming heuristics for the edge-partition problem is limited to the study of the PowerGraph heuristic introduced and studied in the original proposal [9] that advocates the use of edge partitions.

In this paper we study the following two fundamental questions:

Q1 *What are the quantitative performance benefits of using the edge-partition approach as opposed to using the traditional vertex-partition approach?*

and

Q2 *What are the approximation guarantees for the edge-partition problem and is it possible to achieve a good average-case performance by using some natural heuristics for the streaming version of the problem?*

The first question requires some care to ensure a fair comparison. It is important here to compare the two approaches by using a conforming definition of the cut cost, and we distinguish two different definitions of a cut cost. The first definition of a cut cost assumes no aggregation of messages. In this case for the traditional vertex partition problem, the cost of a cut corresponds to the number of edges spanning the partition boundaries, and for the edge partition problem it corresponds to the number of copies of vertices across clusters. The second definition of a cut cost assumes aggregation of messages. For the vertex-partition problem, the cost of fetching messages by a vertex from its neighbor vertices is equal to the number of distinct remote machines to which the neighbor vertices are assigned. Similarly, for the edge-partition problem, the cost of synchronizing the state of a master vertex is equal to the number of distinct remote machines that contain at least one edge with the given vertex as one of its end-points. The second question is studied by deriving theoretical approximation guaranteed and empirical evaluation of a large-set of real-world graphs.

Our main contributions can be summarized in the following points (n is the number of vertices in the graph, m the number of edges and k the number of clusters in the partition):

• We formulate the combinatorial optimization problems for the balanced vertex-partition and edge-partition problems with and without aggregation.

• We characterize the expected cost for each of these four cases under uniform random assignment policy.

• Under uniform random assignment, the expected cost of edge-partition with aggregation is always less than or equal to that of the vertex-partition with aggregation.

• For the edge-partition problem with aggregation, we show a polynomial time $O(d_{\max}\sqrt{\log k \log n})$-approximation algorithm, where $d_{\max}$ is the maximum degree of a vertex, for any graph with $m = \Omega(k^2)$ edges. From a practical point of view, we show that from any 'good' vertex partitioning algorithm, we can derive a 'good' edge partitioning algorithm for any sparse graph.

• We show that the edge-partition problem with no aggregation corresponds to a balanced vertex-partition problem with in-degree-weighted cardinality constraints. Hence, the edge-partition problem has the same approximation guarantees as that of a vertex-partition problem. Again, this implies that any algorithm for the vertex-partition can be easily turned into an edge-partition algorithm with similar performance.

• We report results of an empirical study of a large set of real-world graphs to evaluate the benefits of using the edge-partition approach vs. the vertex-partition approach, evaluate the quality of edge partitions using the approximation algorithms derived in this paper. We found that the edge partitions for the greedy assignment streaming heuristics derived from our offline problem formulations outperform previously best-known heuristics.

The paper is structured as follows. Section 2 formulates the problems of vertex- and edge-partition without and with aggregation. Section 3 provides characterizations of the expected cost for uniform random assignment of vertices of edges, and provide the comparison results. In Section 4, we provide our main results on the approximation guarantees for the edge-partition problem with and without aggregation. Streaming heuristics are discussed in Section 5. Empirical evaluation results are provided in Section 6. Related work is further discussed in Section 7. Finally, we conclude in Section 8.

2. PROBLEM FORMULATION

In this section, we describe the four different partition problems.

Given a directed graph $G = (V, E)$ with the set of vertices V and the set of directed edges $E \subset V \times V$. Let n and m denote the number of vertices and the number of directed edges, respectively. Throughout the paper, we shall use the following notation. Let $N_{\text{in}}(u) = \{v \in V,\ (v,u) \in E\}$ and $N_{\text{out}}(u) = \{v \in V,\ (u,v) \in E\}$ denote the set of vertices that have edges respectively incoming to and outgoing from $u \in V$. Let us define $d_{\text{in}}(u)$ and $d_{\text{out}}(u)$ the in-degree and out-degree of vertex $u \in V$, i.e. $d_{\text{in}}(u) = |N_{\text{in}}(u)|$. Note that $\sum_{u\in V} d_{\text{in}}(u) = \sum_{u\in V} d_{\text{out}}(u) = m$.

A partition of vertices into k clusters is defined as follows: $y_{v,i} = 1$ if vertex $v \in V$ is assigned to partition $i \in [k]$, and $y_{v,i} = 0$ otherwise. Each vertex being assigned to exactly one cluster, we have $\sum_{i=1}^{k} y_{v,i} = 1$ for all $v \in V$.

Similarly, a partition of edges into k clusters is defined as follows: $x_{e,i} = 1$ if edge $e \in E$ is assigned to cluster $i \in [k]$, and $x_{e,i} = 0$, otherwise. Each edge is assigned to exactly one cluster, thus $\sum_{i=1}^{k} x_{e,i} = 1$, for all $e \in E$.

Each partition problem is characterized by a communication cost (denoted by C) which has to be minimized under a condition of balanced loads of the clusters. If we denote by $L(i)$ the load of cluster $i \in [k]$, our partition problems can be written as:

$$\begin{array}{rl} \text{minimize} & C(x) \\ \text{subject to} & \max_{i\in[k]} L_i(x) \le (1+\nu)\frac{\sum_{i\in[k]} L_i(x)}{k} \\ & x \text{ is a valid partition} \end{array}$$

In the sequel we call the parameter $\nu \geq 0$ the load constraint. The precise definitions of the communication cost and the load of a cluster will depend on which problem we consider: vertex or edge partition with or without aggregation. Typically, the communication cost between two clusters is assumed to be linear in the number of messages exchanged in order to synchronize the states of the replicas and the load of a cluster is assumed to be linear in the number of basic operations done by all the elements (vertices or edges) stored on a cluster. We precisely define the four different problems in the sequel.

We can also deal with undirected graphs. In this case, we create the following directed graph from the original graph: we associate to each undirected edge $\{uv\}$ two directed edges (u, v) and (v, u). The problem formulation is then the same except for the following modification: we require that for each undirected edge $\{u, v\}$, we have $x_{(u,v),i} = x_{(v,u),i}$. Note that in this case, $d_{\text{in}}(u) = d_{\text{out}}(u)$ is the degree of vertex u in the undirected graph and m is twice the number of edges in the undirected graph.

2.1 Vertex Partition with No Aggregation

In a vertex partition, the cost function to minimize is typically the number of edges with endpoints in different clusters. This case corresponds to the case with no aggregation. In the case of k clusters, the number of such edges is given by:

$$C^{\text{VP}}(y) = \sum_{i=1}^{k} \sum_{(u,v)\in E} y_{u,i}(1 - y_{v,i}). \quad (1)$$

Standard balanced graph partitioning problem asks for an optimal partitioning of vertices under a balance condition on the cardinality of clusters. Here, we consider a more general version where for some given positive weights on vertices $(a_u,\ u \in V)$, the load of a cluster is given by:

$$L_i^{\text{VP}}(y) = \sum_{u\in V} a_u y_{u,i}. \quad (2)$$

For $\nu \geq 0$, the vertex partition problem with no aggregation is formulated as the following integer programming problem denoted $(k, a) - VP$:

(k, a)-VERTEX PARTITION WITH NO AGGREGATION:

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{k} \sum_{(u,v)\in E} y_{u,i}(1 - y_{v,i}) \\ \text{subject to} & \sum_{u\in V} a_u y_{u,i} \leq (1+\nu)\frac{\sum_{v\in V} a_v}{k},\ i \in [k] \\ & \sum_{i=1}^{k} y_{u,i} = 1,\ u \in V \\ & y \in \{0,1\}^{n\times k} \end{array}$$

The vertex cardinality constraints are accommodated as a special case with unit weights: $a_u = 1$. We shall use the notation (k, d_{in}) to refer to a balanced vertex partition with no aggregation problem where the weights correspond to in-degrees of vertices: $a_u = d_{\text{in}}(u)$, for $u \in V$.

2.2 Vertex Partition with Aggregation

When aggregation is possible, the communication cost changes as edges between two clusters can be lumped together and then incur only a cost of one. In this case, the communication cost is simply:

$$C^{\text{VPA}}(y) = \sum_{u\in V} \sum_{i=1}^{k} y_{u,i} \sum_{j\neq i} \left(1 - \prod_{v\in N_{\text{in}}(u)} (1 - y_{v,j})\right). \quad (3)$$

The balance condition remains the same as above (2) so that the vertex partition with aggregation can be formulated as the following integer programming problem denoted $(k, a) - VPA$:

(k, a)-VERTEX PARTITION WITH AGGREGATION:

$$\begin{array}{ll} \text{minimize} & \sum_{u\in V} \sum_{i=1}^{k} y_{u,i} \sum_{j\neq i} \left(1 - \prod_{v\in N_{\text{in}}(u)} (1 - y_{v,j})\right) \\ \text{subject to} & \sum_{u\in V} a_u y_{u,i} \leq (1+\nu)\frac{\sum_{v\in V} a_v}{k},\ i \in [k] \\ & \sum_{i=1}^{k} y_{u,i} = 1,\ u \in V \\ & y \in \{0,1\}^{n\times k} \end{array}$$

We shall use the notation (k, d_{out}) to refer to a balanced vertex partition with no aggregation problem where the weights correspond to in-degrees of vertices: $a_u = d_{\text{out}}(u)$, for $u \in V$.

2.3 Edge Partition with No Aggregation

We consider now a situation where each edge is assigned to exactly one cluster. As a result, a vertex might have copies in different clusters. For example in Figure 1, node u is replicated 3 times, once in each of the clusters 1, 3 and 4. For each vertex v, we assume that there is a designated *master vertex* assigned to exactly one cluster whose task is to collect messages from all neighbors vertices of v, perform a computation using the input messages, and then communicate the result to all copies of vertex v. The location of a master vertex is indicated by $z_{v,i} = 1$ if the master of vertex v is located in cluster i, and $z_{v,i} = 0$, otherwise. Note that the master of a vertex v is not necessarily located in a cluster that contains at least one copy of vertex v.

In case of no aggregation, we assume that a unit cost is incurred per each edge incident to a given vertex that resides in a cluster different from that of the master vertex of the given vertex. For example in Figure 1, suppose that the master for vertex u is located in cluster 4, then a communication cost of 4 is payed for vertex u as there are four incoming edges that reside in clusters other than 4. In a distributed iterative computation, each such copy of a vertex, will require a message to be sent to the master vertex across different partitions. More formally, the communication cost can be written as:

$$C^{\text{EP}}(x, z) = \sum_{u\in V} \left(d_{\text{in}}(u) - \sum_{i=1}^{k} z_{u,i} \left(\sum_{v\in N_{\text{in}}(u)} x_{(v,u),i} \right) \right). \quad (4)$$

Each master vertex is assumed to incur a processing cost that is equal to the number of edges incident to the corresponding vertex. In a distributed iterative computation, the processing cost of the master is assumed to be linear in the number of messages received per iteration. More formally, the load of cluster i is given by (note it is independent of x):

$$L_i^{\text{EP}}(z) = \sum_{u\in V} d_{\text{in}}(u) z_{u,i}.$$

Finally, for any $\nu \geq 0$, the edge partition problem with no aggregation denoted $k - EP$ is given by:

k-EDGE PARTITION WITH NO AGGREGATION:

$$\begin{array}{ll} \text{minimize} & \sum_{u\in V} \left(d_{\text{in}}(u) - \sum_{i=1}^{k} z_{u,i} \left(\sum_{v\in N_{\text{in}}(u)} x_{(v,u),i} \right) \right) \\ \text{subject to} & \sum_{u\in V} d_{\text{in}}(u) z_{u,i} \leq (1+\nu)\frac{m}{k},\ i \in [k] \\ & \sum_{i=1}^{k} x_{e,i} = 1,\ e \in E \\ & \sum_{i=1}^{k} z_{u,i} = 1,\ u \in V \\ & x \in \{0,1\}^{m\times k} \\ & z \in \{0,1\}^{n\times k} \end{array}$$

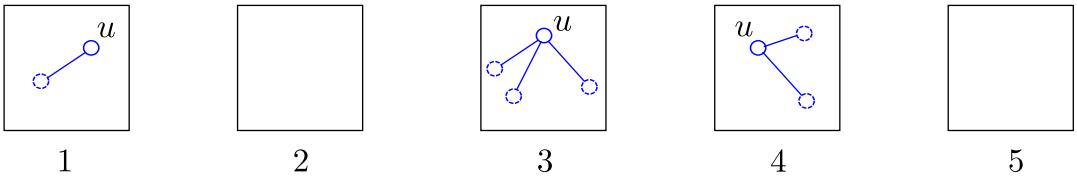


Figure 1: An edge partition (only incoming edges to u are represented).

Note that for directed graphs, the optimal communication cost is zero. Indeed for any admissible $z_{u,i}$, we can take $x_{(v,u),i} = z_{u,i}$ so that the cost becomes: $C^{\mathrm{EP}}(x,z) = 0$. However for undirected graphs, the communication cost will generally not be zero because of the constraint $x_{(u,v),i} = x_{(v,u),i}$ for each undirected edge $\{u,v\}$.

2.4 Edge Partition with Aggregation

If aggregation is possible then both the communication cost and the load are modified as computations are done locally, the results of these computations are lumped and then sent to each master. As a result, each vertex $u \in V$ incurs a communication cost which is simply the number of clusters containing at least an edge $(v,u) \in E$ minus one (corresponding to the cluster containing the master). Summing the contribution of each vertex, we get:

$$C^{\mathrm{EPA}}(x) = \sum_{u\in V}\sum_{i=1}^{k}(1 - \prod_{v\in N_{\mathrm{in}}(u)}(1 - x_{(v,u),i})) - n. \quad (5)$$

Now, since computations are made locally, the load of a cluster is simply proportional to the number of edges contained in this cluster:

$$L_i^{\mathrm{EPA}}(x) = \sum_{e\in E} x_{e,i}.$$

Finally, for any $\nu \geq 0$, the edge partition problem with no aggregation denoted $k-EPA$ is given by (note that we can remove the additive constant n in the cost function without modifying the optimal solution; we keep it to allow to make comparison with the other problems):

k-EDGE PARTITION WITH AGGREGATION:

$$\begin{array}{ll} \text{minimize} & \sum_{u\in V}\sum_{i=1}^{k}(1 - \prod_{v\in N_{\mathrm{in}}(u)}(1 - x_{(v,u),i})) - n \\ \text{subject to} & \sum_{e\in E} x_{e,i} \leq (1+\nu)\frac{m}{k},\ i \in K \\ & \sum_{i=1}^{k} x_{e,i} = 1,\ e \in E \\ & x \in \{0,1\}^{m\times k} \end{array}$$

3. RANDOM ASSIGNMENT

In this section we consider the expected cost of random assignment policy that for the case of edge partition assigns each edge independently and uniformly at random to one of k partitions, and likewise for the case of vertex partition assigns each vertex independently and uniformly at random to one of k partitions. We shall provide exact characterizations of these expected costs for the case without and with aggregation. We then show that aggregation lowers the expected communication cost (in most cases). Moreover, also expected costs are the same for edge and vertex partition with no aggregation, we find that when aggregation is possible, edge partition has a lower expected cost than vertex partition. There are two statistics of the in-degree sequence of a given graph that play an important role here for the case without and with aggregation, respectively, the *number of (directed) edges*: $m = \sum_{v\in V} d_{\mathrm{in}}(v)$ and

$$\psi(d_{\mathrm{in}},k) = \frac{1}{n}\sum_{v\in V}\left(1 - \frac{1}{k}\right)^{d_{\mathrm{in}}(v)}.$$

The latter statistic corresponds to a *moment generating function of the degree sequence evaluated at the value* $\log(1 - 1/k)$.

3.1 Edge Partition

PROPOSITION 1. *For any given graph with sequence of in-degrees $d_{\mathrm{in}} = (d_{\mathrm{in}}(v))_{v\in V}$, we assign edges independently and uniformly at random across k clusters. In the case of edge partition with no aggregation, we assign the master to a random cluster. Then we have the following expected communication costs:*

$$\mathbb{E}[C^{\mathrm{EP}}] = \left(1 - \frac{1}{k}\right)m \quad (6)$$

$$\mathbb{E}[C^{\mathrm{EPA}}] = kn(1 - \psi(d_{\mathrm{in}},k)) - n. \quad (7)$$

COROLLARY 1. *As soon as $nk \geq m$, aggregation lowers the communication cost under a random edge partition, i.e. $\mathbb{E}[C^{\mathrm{EP}}] > \mathbb{E}[C^{\mathrm{EPA}}]$. Moreover, for any given graph, the gap between the two expected costs is at least $(n - m/k)$.*

Proofs are provided in Appendix [7].

3.2 Vertex Partition

PROPOSITION 2. *For any given graph with sequence of degrees $d_{\mathrm{in}} = (d_{\mathrm{in}}(v))_{v\in V}$, we assign vertices independently and uniformly at random across k clusters. Then we have the following expected communication costs:*

$$\mathbb{E}[C^{\mathrm{VP}}] = \left(1 - \frac{1}{k}\right)m \quad (8)$$

$$\mathbb{E}[C^{\mathrm{VPA}}] = n(k-1)(1 - \psi(d_{\mathrm{in}},k)). \quad (9)$$

COROLLARY 2. *Aggregation lowers the communication cost under a random vertex partition, i.e. $\mathbb{E}[C^{\mathrm{VP}}] > \mathbb{E}[C^{\mathrm{VPA}}]$.*

Proofs are provided in Appendix [7].

3.3 Edge vs. Vertex Partition

Note that the expected costs are the same for the edge and vertex partitions when there is no aggregation. The following easy corollary shows that the situation is different with aggregation:

COROLLARY 3. *For every given graph, the expected cost of vertex partition with aggregation is greater than the expected cost of edge partition with aggregation. Moreover, the gap between the two expected costs is at least* $n\psi(d_{\text{in}}, k)$.

PROOF. Follows directly from (7) and (9). □

4. APPROXIMATION ALGORITHMS

In this section, we restrict ourselves to the undirected case so that $N_{\text{in}}(u) = N_{\text{out}}(u) = N(u)$, $d_{\text{in}}(u) = d_{\text{out}}(u) = d(u)$ and $\sum_{u\in V} d(u) = m$ is twice the number of (undirected) edges.

4.1 Hardness Results

It is well-known that the $(2,1)$-vertex partition problem with no aggregation, i.e. the minimum bisection problem is NP-complete but efficient approximation algorithms have been obtained in [6]. The following theorem shows that it is not possible to derive a finite approximation factor for each of the four problems when k is not a constant.

THEOREM 1. *Each of the four problems with* $\nu = 0$*: (k,d)-vertex partition with/with no aggregation and k-edge partition with/with no aggregation, has no polynomial time approximation algorithm with finite approximation factor unless* $P = NP$.

PROOF. The case of $(k,1)$-vertex partition with no aggregation is Theorem 2.1 in [17] and the other cases follow by the same reduction from the 3-Partition problem (details omitted). □

4.2 Edge Partition with Aggregation

For edge partition with aggregation we show that there exists a polynomial-time algorithm with approximation ratio that matches that of the vertex partition problem up to a $d_{\max}$ factor, where $d_{\max}$ is the maximum degree of a vertex. Since the best known approximation ratio for the vertex partition problem is poly-logarithmic in the number of vertices of the graph, this yields a poly-logarithmic approximation algorithm for any sparse graph, i.e. such that $d_{\max} = O(\log^c n)$, for some positive constant c. The algorithm uses as a subroutine an approximation algorithm for the vertex partition problem with no aggregation and the load of a partition defined to be the sum of degree weighted vertices assigned to this partition. The algorithm can be summarized in the following two steps:

- Partition the set of vertices according to an approximation algorithm for the (k,d)-VP problem with load parameter ν, and let $S_1, S_2, \ldots, S_k$ denote the output of this algorithm.
- For each pair of vertex partitions S_i and S_j, assign the set of cut edges with end-vertices in S_i and S_j to partitions i and j such that if the number of such edges is even, exactly half of edges is assigned to each partition, and otherwise, either of them gets at least half of edges (to ensure a balanced partitioning).

THEOREM 2. *There exists a polynomial-time algorithm that given a feasible solution y with load constraint ν to (k,d)-*VP *of cost $C^{\text{VP}}(y)$ outputs a feasible solution x with load constraint $\nu + \frac{k^2}{m}$ to k-*EPA *of cost $C^{\text{EPA}}(x)$ such that*

$$\frac{1}{d_{\max}} C^{\text{VP}}(y) \le C^{\text{EPA}}(x) \le C^{\text{VP}}(y). \tag{10}$$

PROOF. The proof consists of two steps. First, we show that given a feasible vertex partition y for the (k,d)-VP problem with the load parameter ν, the given construction of an edge partition x is a feasible solution for the k-EPA problem with the load parameter $\nu + k^2/m$. Second, we show that the objective functions $C^{\text{EPA}}(x)$ and $C^{\text{VP}}(y)$ satisfy the relations in (10).

Suppose y is a feasible solution to the (k,d)-VP problem and let $C^{\text{VP}}(y)$ denote the value of the cut. Let S_i be the set of vertices in partition i, i.e. $S_i = \{v \in V : y_{v,i} = 1\}$. The total cost of the cut can be expressed as $\frac{1}{2}\sum_{i\in[k]} |E(S_i, V \setminus S_i)|$ where for $A, B \subset V$, $E(A,B)$ denotes the set of edges with exactly one end in A and the other end in B. Now, note that

$$|E(S_i, S_i)| = \frac{1}{2}\left(\sum_{u\in V} y_{u,i} d(u) - |E(S_i, V \setminus S_i)|\right).$$

Since y is assumed to be a feasible assignment,

$$|E(S_i, S_i)| \le (1+\nu)\frac{m}{k} - \frac{1}{2}|E(S_i, V \setminus S_i)|. \tag{11}$$

Let us define the edge-partition $P_1, P_2, \ldots, P_k$ as follows: for each pair $i, j \in [k]$ such that $i \neq j$, split $E(S_i, S_j)$ into two sets $P_{i,j}$ and $P_{j,i}$ of balanced cardinalities, i.e. $P_{i,j}, P_{j,i} \subset E(S_i, S_j)$ such that $P_{i,j} \cup P_{j,i} = E(S_i, S_j)$, $P_{i,j} \cap P_{j,i} = \emptyset$, and $-1 \le |P_{i,j}| - |P_{j,i}| \le 1$. Define $P_i = E(S_i, S_i) \cup \cup_{j\neq i} P_{i,j}$, for $i \in [k]$.

It is easy to observe that for all $i \in [k]$,

$$-k+1 \le |\cup_{j\neq i} P_{i,j}| - \frac{1}{2}|E(S_i, V \setminus S_i)| \le k-1. \tag{12}$$

Hence

$$\begin{aligned} |P_i| &= |E(S_i, S_i) \cup (\cup_{j\neq i} P_{i,j})| \\ &= |E(S_i, S_i)| + |\cup_{j\neq i} P_{i,j}| \\ &\le (1+\nu)\frac{m}{k} + k - 1 \\ &\le (1 + \nu + \frac{k^2}{m})\frac{m}{k} \end{aligned}$$

where the first inequality follows by (11) and (12) and the second inequality is by the fact $k - 1 \le (1+\nu)(k^2/m)m/k$. This implies that the given edge partitioning satisfies the constraints of the k-EPA problem.

It remains to show that the objective functions $C^{\text{EPA}}(x)$ and $C^{\text{VP}}(y)$ satisfy the relations in (10). First note that only edges cut in the vertex partition y contribute to the communication cost $C^{\text{EPA}}(x)$, indeed removing all the edges which are not cut in y will not change $C^{\text{VP}}(y)$ nor $C^{\text{EPA}}(x)$. Moreover each edge cut in y contribute to at most one in $C^{\text{EPA}}(x)$ so that $C^{\text{EPA}}(x) \le C^{\text{VP}}(y)$.

Inversely, each vertex $u \in V$ incurs a cost of at most $d_{\max}$ in the communication cost $C^{\text{VP}}(y)$ (corresponding to the case where each incident edge to u is cut in y). A vertex $u \in V$ incurs a cost of zero in $C^{\text{VP}}(y)$ if none of the incident edge to u is cut. In this last case, the vertex will not contribute to the communication cost $C^{\text{EPA}}(x)$ by previous remark. Hence we have $C^{\text{VP}}(y) \le d_{\max} C^{\text{EPA}}(x)$. □

COROLLARY 4. *Assume that $m = \Omega(k^2)$, then there exists a polynomial-time algorithm with approximation ratio $O(d_{\max}\sqrt{\log k \log n})$ for k-*EPA *problem with $\nu = 1$.*

PROOF. This is a corollary of Theorem 2 and an easy extension of Theorem 1.1 in [18] to the (k,d)-VP problem showing that there exists a polynomial-time approximation algorithm with the approximation ratio $O(\sqrt{\log k \log n})$. □

4.3 Edge Partition with No Aggregation

We shall show that the edge partition is in a one-to-one correspondence to a vertex partition problem. This will show that the computational complexity of the balanced edge partition with no aggregation corresponds to that of a balanced vertex partition problem (with no aggregation). This reduction will be established by the following simple two-step construction:

1. Suppose y is an optimal solution to the (k,d)-vertex partition problem.
2. Then, partition edges by assigning each edge $(u,v) \in E$ to the cluster of either of its end vertices u and v, as specified by y, arbitrarily.

THEOREM 3. *k-edge partition problem is in a one-to-one correspondence with the (k,d)-vertex partition problem.*

Proof of this theorem is provided in Appendix [7]. Again thanks to an easy extension of Theorem 1.1 in [18], we obtain the following corollary of Theorem 3:

COROLLARY 5. *There exists a polynomial-time algorithm with approximation ratio $O(\sqrt{\log k \log n})$ for the k-edge partition problem.*

5. STREAMING HEURISTICS

In this section, we discuss streaming heuristics for the balanced edge partition problem without and with aggregation. We shall first introduce a state-of-the-art heuristic for the balanced edge partition problem and point out to some of its deficiencies. We shall then move to introducing a class of novel streaming heuristics that are inspired by the analysis of the offline version of the problem in Section 4.

5.1 PowerGraph Heuristic

Suppose edges arrive in an arbitrary order. In the description of the algorithm below, $S(v)$ will be a set evolving as edges arrive and it will contain the clusters with at least one already observed edge incident to vertex v. Initially, $S(v) = \emptyset$, for all $v \in V$. The algorithm defined below requires to know for each arriving edge, the number of unassigned edges for each end-vertex of this edge. For this to be realized, it suffices to know the degree of each vertex, and then the number of the unassigned edges of a vertex can be kept as part of the algorithm state (this we omit in the description presented below).

PowerGraph heuristic Per each arrival of an edge (u,v), do the following:
1. If the intersection of $S(u)$ and $S(v)$ is non-empty, then assign edge (u,v) to a cluster I in this intersection.
2. Otherwise, assign edge (u,v) to a cluster I in $S(u) \cup S(v)$ (containing either u or v) with the most unassigned edges.
3. Otherwise, if both $S(u)$ and $S(v)$ are empty, then assign edge (u,v) to a partition I with the least number of assigned edges.
4. $S(u) \leftarrow S(u) \cup \{I\}$ and $S(v) \leftarrow S(v) \cup \{I\}$.

This heuristic can be showed to correspond to a greedy assignment that for each arriving edge minimizes the expected cut cost conditional on that all subsequent edges are assigned to partitions uniformly at random. This can be seen as a derandomization of the uniform random assignment, and is thus guaranteed to yield a cut cost that is guaranteed to be less or equal to that of the uniform random assignment.

An important observation from the definition of the Powergraph heuristic is that edges are assigned using a greedy assignment that prioritizes assignment of an edge to a partition that already contains one of the end-vertices, and the load balancing of edges is performed only if neither of the end-vertices have been already assigned. This may result in a gross imbalance of the partition sizes. For example, a worst-case is a breadth-first search traversal of a connected tree, where all edges end up being assigned to one partition, which results in an extreme imbalance.

5.2 Greedy Online Assignments

5.2.1 Edge partition

For the basic formulation, we derive a greedy online assignment for edge partition with aggregation by an irrevocable assignment of edges to clusters as they are observed in the input stream to a cluster that minimizes the marginal cost. We denote by P_i the clusters of an edge-partition. Then for any $S \subset E$, $V(S)$ is the set of vertices spanned by S, i.e. all vertices adjacent to an edge in S. Note from (5) that the marginal cost of the cut size by adding an edge (u,v) to cluster i is equal to

$$|V(P_i \cup (u,v))| - |V(P_i)| = |V(P_i \cup (u,v)) \setminus V(P_i)|.$$

It is noteworthy that in any case the marginal cost of the cut size is either 0, 1 or 2: first, it is zero if the edge is placed in a cluster in which both of its end-vertices have been already assigned, second, it is equal to 1 if exactly one of its end-vertices has been previously assigned, and third, it is equal to 2 if neither of its vertices have been previously assigned to this cluster.

Since $|V(P_i \cup (u,v)) \setminus V(P_i)| = 2 - |V(P_i) \cap \{u,v\}|$, a greedy online heuristic for minimizing the cut size is to assign each input edge (u,v) to a cluster $i \in I$ where

$$I = \text{argmax}_{j \in [k]} \{|V(P_j) \cap \{u,v\}|\} \tag{13}$$

which means that the edge is assigned to a cluster in the following priority order: (i) in a cluster that already contains both of its end-vertices, if any exists, (ii) in a cluster that already contains exactly one of its end-vertices, if any exists, and else (iii) to an arbitrary cluster. Such an assignment is greedy with respect to the cost of the cut criteria and is entirely ignorant of the load balancing criteria.

A way to define a greedy assignment that accounts for both criteria is to redefine the optimization problem as follows. Let $c : R_+ \to R_+$ be an increasing convex function such that $c(0) = 0$. Then, redefine k-EPA by moving the hard constraints into the objective function as a soft penalty function, as given in here

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{k} |V(P_i)| + c(|P_i|) \\ & (P_1, P_2, \ldots, P_k) \in \mathcal{P}(G). \end{array}$$

The greedy online assignment with respect to this problem amounts to assigning an input edge (u,v) to a cluster $i \in I$

where

$$I = \text{argmax}_{j\in[k]}\{|V(P_j)\cap\{u,v\}|-[c(|P_j\cup(u,v)|)-c(|P_j|)]\}.$$

We refer to this class of greedy assignment policies as the least incremental cost (IC) assignment.

An alternative class of heuristics is derived by following an approach pursued for balanced vertex partitioning that amounts to discounting the marginal benefit in (13) with the cardinality of the partition. Specifically, for given decreasing function $d : R_+ \to R$, assign an input edge (u, v) to a cluster I where

$$I = \text{argmax}_{j\in[k]}\{|V(P_j) \cap \{u, v\}|d(|P_j|)\}.$$

Notice that unlike to the PowerGraph heuristic, the greedy online assignments as defined in this section do not require any extra information about the input data, such as knowing the degree of each vertex.

5.2.2 *Vertex partition*

We use as a subroutine a greedy online algorithm for approximating vertex partitioning defined in [34], which is very similar to the one described in the previous section. It supposes vertex arrival. The idea of the algorithm is to minimize the following objective function: $\sum_{i\in[k]} |E(S_i, V\backslash S_i)|+ c(|S_i|)$. This can be done with this greedy online assignment: Per vertex u arrival, we assign it to a partition $i \in I$ where

$$I = \text{argmax}_{j\in[k]}\{|N_{\text{in}}(u) \cap P_j| + c(|P_j \cup u|) - c(|P_j|)\}.$$

We refer to this class of greedy vertex assignment policies as the vertex partition least incremental cost (VP-IC) assignment.

5.3 Streaming Edge Partition Based on Balanced Vertex Partition

Suppose edges arrival in an arbitrary order. Suppose we have some partitioning of the vertices S_i, for all $i \in \{1..k\}$, which are balanced with regard to degree weights. We can derive easily an edge partitioning from this vertex partitioning as follows. Per each arrival of an edge (u, v), flip a fair coin, and do either of the following: (1) assign (u, v) to partition i such that $u \in S_i$, or (2) assign (u, v) to partition i such that $v \in S_i$. This will give an edge partitioning with a communication cost lesser than the vertex partitioning's one, and should keep a maxload ratio close to the previous one thanks to random assignment.

6. EVALUATION

In this section we report results of empirical evaluation of several hypotheses on a dataset containing samples of five real-world graphs whose properties are summarized in Table 1. This dataset contains samples of real-graphs with quite some diversity along several dimensions, including the scale ranging from hundred of thousands to millions of vertices, sparsity ranging from a few to hundreds of edges per vertex, and type including social graphs, co-purchasing of products and co-authorship of scientific papers. In the first part of this section, we shall evaluate expected communication cost for edge and vertex partition problems with and without aggregation under random assignment of edges of vertices, under our benchmark random edge or vertex assignment, which is commonly deployed in practice. This part of our analysis provides empirical support to the following claims:

(H1) For balanced edge partition, aggregation of messages typically provides significant reduction of the communication cost.

(H2) Balanced edge partition with aggregation can have significantly smaller expected communication cost than balanced vertex partition with aggregation.

In the second part, we shall focus on evaluating our approximation algorithms for balanced edge partition problem, for the offline and the online version of the problem. For the online version, we shall compare with previously-proposed method for balanced edge partition problem, namely PowerGraph heuristics. Our empirical validation results provide support to the following claims:

(H3) PowerGraph heuristic provides significant reduction of the communication cost with respect to random edge partition. However, this is at the expense of a maximum load of a partition that may assume rather large values.

(H4) Least incremental cost heuristic (described in Section 5.2.1) typically has communication cost that is smaller or equal to that of PowerGraph heuristic and nearly perfectly balanced loads of partitions.

(H5) Balanced edge partition derived from a vertex partition with degree-weighted vertices using an offline solver tends to have significantly smaller communication cost than online assignment strategies.

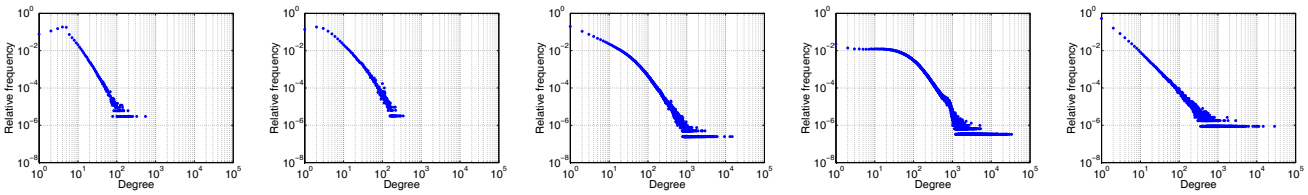
(H6) Balanced edge partition derived from a vertex partition with degree-weighted vertices using the least marginal cost online heuristic tends to have much smaller communication cost than random edge assignment, but worse than least marginal cost strategy for balanced edge partition problem.

6.1 Communication Cost under Random Edge or Vertex Assignments

We evaluate expected communication costs under random edge or vertex assignments. In Section 3 we provided a complete characterization of the expected communication costs under random or vertex assignment, and observed that they all depend in one way or another on the degree sequence of the given input graph. The degree distributions of the graphs in our dataset are showed in Figure 2. As one would expect they all exhibit some degree of a power-law dependence. For all the examples, the 0.99-quantile of the vertex degree is in the order of 10 vertices. A notable distinction of Orkut graph data is that an approximately same portions of vertices have degrees in the range from one to order ten. In Figure 3 we show expected communication cost for edge partition with and without aggregation, and vertex partition with aggregation versus the number of partitions ranging from 2 to 32 partitions. In this figure, the circles represent empirical means computed over 10 repeated partitions. As one would expect, the empirical means are highly concentrated around the expected values computed using the degree sequence in the characterization results presented in Section 3. For all graphs in our dataset, aggregation of messages for balanced edge partition problem yields significant reduction of communication cost. For a small number of partitions, the communication cost of edge partition with aggregation can be as small as 1/10 of that with no aggregation. From Figure 3, we also observe that balanced edge partition with aggregation typically has smaller communication cost than vertex partition with aggregation, and there

	Nodes	Edges	Mean degree	Median degree	90% quantile	Max degree	Description
Amazon	334 863	925 872	5.53	3	9	549	Co-purchasing
dblp	317 080	1 049 866	6.62	3	13	343	Co-authorship
Livejournal	3 997 962	34 681 189	17.35	5	41	14 815	Social
Orkut	3 072 441	117 185 083	76.28	44	161	33 313	Social
Youtube	1 134 890	2 987 624	5.27	1	7	28 754	Social

Table 1: Datasets used in our experiments.



(a) Amazon (b) dblp (c) Live Journal (d) Orkut (e) Youtube

Figure 2: Degree distributions for the graphs in our dataset.

exist cases where the difference is significant. Overall, these results provide support to hypothesis (H1) and (H2) asserted at the beginning of this section.

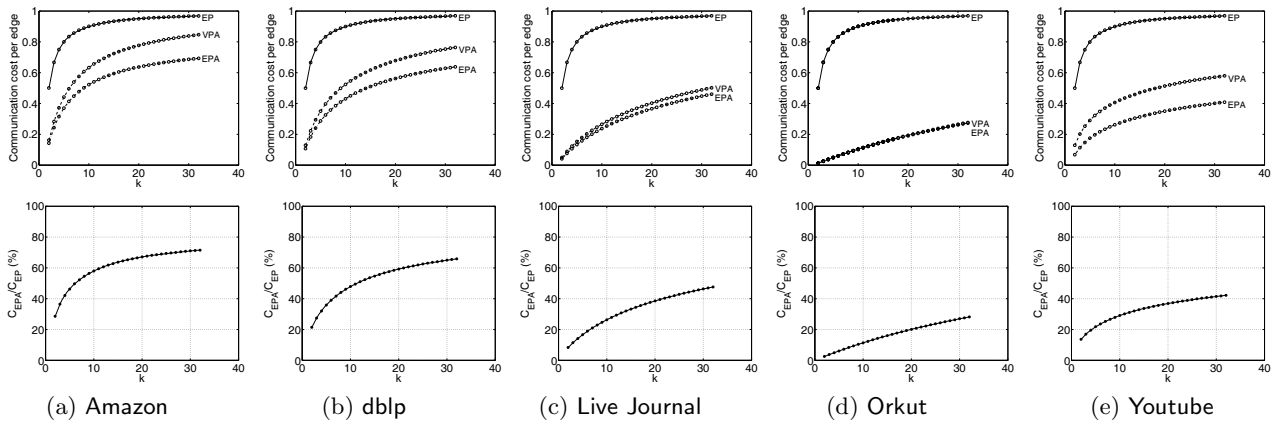
6.2 Approximation Algorithms

We evaluated various approximation algorithms for balanced edge partition problem with respect to the two criteria of the communication cost and the maximum load of a partition. In Figure 4, we present results for different approximation algorithms for each input graph from our dataset. RND refers to random edge assignment. VP-IC refers to edge partition derived as explained in Section 5.3 from a vertex partitioning given by a greedy online assignment as defined in Section 5.2.2 with $c(x) = \alpha x^\gamma$, $\alpha = n(\frac{k}{m})^\gamma$ and $\gamma = 1.5$. PG refers to Powergraph heuristic defined in Section 5.1. IC refers to greedy online assignment as defined in Section 5.2.1 with $c(x) = \alpha(\sum_{u \in x} d_u)^\gamma$, $\alpha = m\frac{k^{\gamma-1}}{n^\gamma}$ and $\gamma = 1.5$. And, finally, VP-METIS refers to edge partitioning derived as explained in Section 5.3 from a vertex partitioning given by the offline tool METIS, which is a state-of-the-art software tool for approximating balanced vertex partition problem.

In this part of analysis we evaluate performance of three one-pass streaming algorithms for balanced edge partition, including random, least marginal cost, and PowerGraph, and the approximation algorithm for the offline version of the problem in Section 4.2. In particular, in the latter case, we use METIS solver for finding an approximate solution to the subroutine balanced vertex partition with degree-weighted vertices. For the online version of the problem, we observe that for all the graphs in our dataset, least marginal cost algorithm yields a smaller or equal communication cost than PowerGraph. We observe that PowerGraph exhibits varying quality of partition with respect to the maximum load of a partition, which in some cases attains significantly larger values than any other method. Especially, under PowerGraph assignment, the maximum load tends to increase with the number of partitions, and can achieve maximum load values of more than 10% of extra load to the mean number of edges per partition. On the other hand, the least marginal cost assignment achieves nearly perfect load balancing, and still performs better or nearly equal than PowerGraph with respect to the communication cost. Overall, these observations provide empirical support to the hypotheses (H3) and (H4). For the offline approximation algorithm described in Section 4.2, we observe that it can perform significantly superior performance with respect to the communication cost than one-pass streaming algorithms. This provides empirical support to hypothesis (H5). Finally, we also observe that using a one-pass streaming algorithm for the balanced vertex partition problem with degree-weighted vertices and then deriving an edge partition from the resulting vertex partition can provide significant reduction of the communication cost with respect to random edge assignment, but worse than the least marginal cost strategy for balanced edge partition problem. This provides empirical support to hypothesis (H6). An implication of this hypothesis is that one needs to exercise some care in designing a scalable partition strategy based on a few passes through graph data based on the approach in Section 4.2 that uses a balanced vertex partition problem as a subroutine.

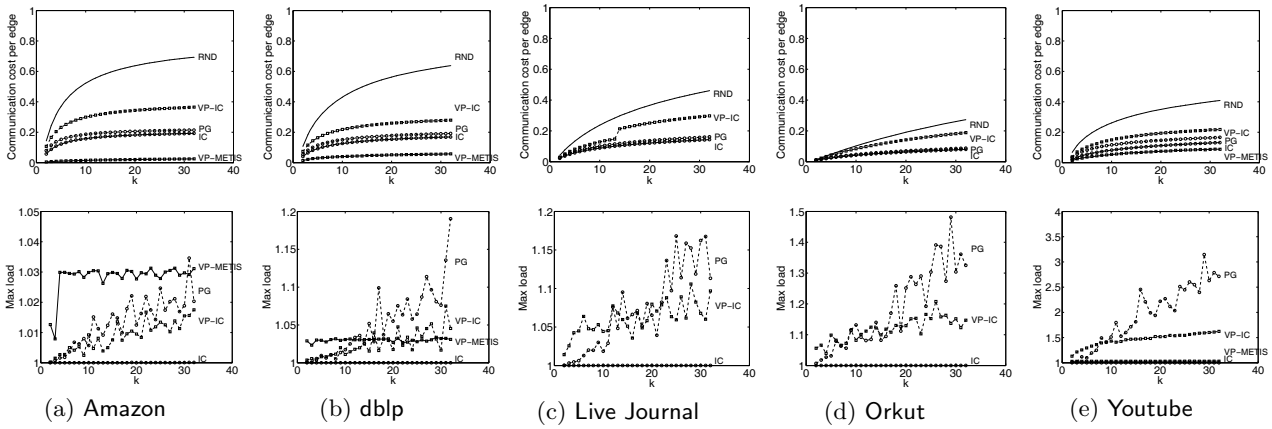
7. RELATED WORK

Traditional balanced graph partitioning problem, referred to in this paper as the vertex partition problem with no aggregation, has been studied extensively by theoretical computer science and more applied communities. The best known approximation ratio for this problem is $O(\sqrt{\log k \log n})$ [17] and [18], using a semi-definite programming relaxation and a randomized rounding. The solvers used in practice rely on a combination of various heuristics, e.g. one commonly used software is METIS [12, 14, 13] that combines a number of heuristics including the well-known Kernighan and Li heuristic [15]. Graph partitioning has attracted a lot of attention recently by the systems community to scale out large-scale computations, e.g. for parallel distributed computing in distributed clusters of machines [9, 22, 21, 11, 38, 40, 16], graph databases and graph analytics platforms [30, 37, 36, 4], search and social networks [35, 5, 28], and multicore and share memory systems [27, 2, 31, 29]. In recent



(a) Amazon (b) dblp (c) Live Journal (d) Orkut (e) Youtube

Figure 3: Expected communication costs for edge partition, edge partition with aggregation, and vertex partition with aggregation vs. the number of partitions k.



(a) Amazon (b) dblp (c) Live Journal (d) Orkut (e) Youtube

Figure 4: Approximation algorithms: (top) communication cost and (bottom) maximum load.

work, scalable strategies for graph strategies have been investigated that are based on one-pass streaming algorithms. An early work in this direction is [32] that focused on the traditional graph partitioning problem, improved strategies in [34], and a multi-pass version in [25]. [35] studied local improvements algorithms that aim at minimizing the cut subject to a budget constraint on the number of vertices to reallocate. In this recent work, novel variants of graph partitioning problems have emerged that are different from the traditional balanced graph partitioning problem with respect to the definition of the cut cost function and the load balancing constraints. One such new variant is the balanced edge partition problem, originally proposed in [9], which we have studied in this paper.

From a theoretical standpoint, the balanced edge partition problem is an instance of a submodular function minimization subject to cardinality constraints. For general class of submodular load balancing problems, the approximation guarantee of $O(\sqrt{n/\log n})$ was showed to hold in [33]. Our approximation ratio of $O(d_{\max}\sqrt{\log k \log n})$ provides a better approximation guarantee for the subclass of balanced edge partition problems for any graphs with $d_{\max} = o(\sqrt{n})$.

8. CONCLUSION

In this paper we analyzed balanced graph edge partition problem that was recently proposed as an alternative to traditional balanced graph vertex partition for scaling out distributed computations in data analytics platforms. We provided explicit characterization of the expected communication cost for different variants of the graph partition problem under commonly deployed scheme of randomly assigning edges of vertices to a given number of partitions. We provided approximation algorithms for balanced edge partition problem with and without message aggregation. Interestingly, both problems use a balanced vertex partition problem as a subroutine, thus allowing us to reutilize known techniques and off-the-shelf software for this traditional graph partition problem. For scalable partitioning based on a single pass through graph data, we showed that the least marginal cost greedy heuristic provides better performance than the best previously known heuristic with respect to both the communication cost and load balancing of partitions. The results in this paper present a first step towards a more thorough analysis of balanced edge partition problem both on the side of theoretical approximation

guarantees and the design of practical algorithms with good average-case performance.

9. REFERENCES

[1] A. Abou-Rjeili and G. Karypis. Multilevel algorithms for partitioning power-law graphs. In *Proc. of the 20th Int'l Conf. on Parallel and Distributed Processing*, IEEE IPDPS'06, pages 124–124, Washington, DC, USA, 2006.

[2] V. Agarwal, F. Petrini, D. Pasetto, and D. A. Bader. Scalable graph exploration on multicore processors. In *Proc. of the 2010 ACM/IEEE Int'l Conf. for High Performance Computing, Networking, Storage and Analysis*, SC '10, pages 1–11, Washington, DC, USA, 2010.

[3] C. Curino, E. Jones, Y. Zhang, and S. Madden. Schism: a workload-driven approach to database replication and partitioning. In *VLDB '10*, 2010.

[4] M. Curtiss, I. Becker, T. Bosman, S. Doroshenko, L. Grijncu, T. Jackson, S. Kunnatur, S. Lassen, P. Pronin, S. Sankar, G. Shen, G. Woss, C. Wang, and N. Zhang. Unicorn: A system for searching the social graph. In *VLDB '13*, 2013.

[5] Q. Duong, S. Goel, J. Hofman, and S. Vassilvitskii. Sharding social networks. In *ACM WSDM '13*, pages 223–232, New York, NY, USA, 2013.

[6] U. Feige and R. Krauthgamer. A polylogarithmic approximation of the minimum bisection. *SIAM J. Comput.*, 31(4):1090–1118, Apr. 2002.

[7] M. V. Florian Bourse, Marc Lelarge. Balanced edge partition. Technical Report MSR-TR-2014-20, Microsoft Research, 2014.

[8] T. A. S. F. Giraph. http://giraph.apache.org, 2014.

[9] J. E. Gonzalez, Y. Low, H. Gu, D. Bickson, and C. Guestrin. Powergraph: distributed graph-parallel computation on natural graphs. In *OSDI'12*, pages 17–30. USENIX Association, 2012.

[10] D. Gregor and A. Lumsdaine. The parallel bgl: A generic library for distributed graph computations. In *Proceedings of POOSC*, 2005.

[11] U. Kang, C. E. T., and C. Faloutsos. Pegasus: A peta-scale graph mining system. In *ICDM*, pages 229–238, 2009.

[12] G. Karypis and V. Kumar. Metis-unstructured graph partitioning and sparse matrix ordering system, version 2.0. 1995.

[13] G. Karypis and V. Kumar. Parallel multilevel graph partitioning. In *Proc. of the 10th Int'l Parallel Processing Symposium*, IEEE IPPS '96, pages 314–319, Washington, DC, USA, 1996.

[14] G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing*, 48(1):96 – 129, 1998.

[15] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell Syst. Tech. J.*, 49(2):291–307, Feb. 1970.

[16] Z. Khayyat, K. Awara, A. Alonazi, H. Jamjoom, D. Williams, and P. Kalnis. Mizan: A system for dynamic load balancing in large-scale graph processing. In *Proc. of the 8th ACM European Conference on Computer Systems*, ACM EuroSys '13, pages 169–182, New York, NY, USA, 2013.

[17] A. Konstantin and H. Räcke. Balanced graph partitioning. In *SPAA '04*, pages 120–124, 2004.

[18] R. Krauthgamer, J. S. Naor, and R. Schwartz. Partitioning graphs into balanced components. In *SODA '09*, pages 942–949, 2009.

[19] K. Lang. Finding good nearly balanced cuts in power law graphs. Technical Report YRL-2004-036, Yahoo! Research Labs, 2004.

[20] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, 2008.

[21] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J. M. Hellerstein. Distributed graphlab: a framework for machine learning and data mining in the cloud. *Proc. VLDB Endow.*, 5(8):716–727, Apr. 2012.

[22] Y. Low, J. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin, and J. M. Hellerstein. Graphlab: A new framework for parallel machine learning. In *UAI*, pages 340–349, 2010.

[23] G. Malewicz, M. H. Austern, A. Bik, J. Dehnert, I. Horn, N. Leiser, and G. Czajkowski. Pregel: a system for large-scale graph processing. In *ACM SIGMOD '10*, pages 135–146, 2010.

[24] Neo4j. http://www.neo4j.org, 2014.

[25] J. Nishimura and J. Ugander. Restreaming graph partitioning: simple versatile algorithms for advanced balancing. In *ACM KDD '13*, pages 1106–1114, 2013.

[26] F. Pellegrini and J. Roman. Scotch: A software package for static mapping by dual recursive bipartitioning of process and architecture graphs. In *High-Performance Computing and Networking*, volume 1067 of *Lecture Notes in Computer Science*, pages 493–498. Springer Berlin Heidelberg, 1996.

[27] V. Prabhakaran, M. Wu, X. Weng, F. McSherry, L. Zhou, and M. Haridasan. Managing large graphs on multi-cores with graph awareness. In *USENIX ATC'12*, pages 4–4, 2012.

[28] J. M. Pujol, V. Erramilli, G. Siganos, X. Yang, N. Laoutaris, P. Chhabra, and P. Rodriguez. The little engine(s) that could: scaling online social networks. In *ACM SIGCOMM '10*, pages 375–386, New York, NY, USA, 2010.

[29] A. Roy, I. Mihailovic, and W. Zwaenepoel. X-stream: Edge-centric graph processing using streaming partitions. In *ACM SOSP'13*, 2013.

[30] B. Shao, H. Wang, and Y. Li. Trinity: A distributed graph engine on a memory cloud. In *Proceedings of the VLDB Endowment*, VLDB '13, 2013.

[31] J. Shun and G. E. Blelloch. Ligra: a lightweight graph processing framework for shared memory. In *ACM PPoPP '13*, pages 135–146, New York, NY, USA, 2013.

[32] I. Stanton and G. Kliot. Streaming graph partitioning for large distributed graphs. In *ACM KDD '12*, pages 1222–1230, 2012.

[33] Z. Svitkina and L. Fleischer. Submodular approximation: Sampling-based algorithms and lower bounds. *SIAM J. Comput.*, 40(6):1715–1737, Dec. 2011.

[34] C. Tsourakakis, C. Gkantsidis, B. Radunovic, and M. Vojnovic. Fennel: Streaming graph partitioning for massive scale graphs. In *ACM WSDM '14*, 2014.

[35] J. Ugander and L. Backstrom. Balanced label propagation for partitioning massive graphs. In *ACM WSDM '13*, pages 507–516, 2013.

[36] V. Venkataramani, Z. Amsden, N. Bronson, G. Cabrera III, P. Chakka, P. Dimov, H. Ding, J. Ferris, A. Giardullo, J. Hoon, S. Kulkarni, N. Lawrence, M. Marchukov, D. Petrov, and L. Puzar. TAO: how facebook serves the social graph. In *ACM SIGMOD '12*, pages 791–792, 2012.

[37] R. S. Xin, J. E. Gonzalez, M. J. Franklin, and I. Stoica. Graphx: a resilient distributed graph system on spark. In *First International Workshop on Graph Data Management Experiences and Systems*, ACM GRADES '13, pages 2:1–2:6, 2013.

[38] S. Yang, X. Yan, B. Zong, and A. Khan. Towards effective partition management for large graphs. In *ACM SIGMOD '12*, pages 517–528, 2012.

[39] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *NSDI'12*, pages 2–2, Berkeley, CA, USA, 2012. USENIX Association.

[40] J. Zhou, N. Bruno, and W. Lin. Advanced partitioning techniques for massively distributed computation. In *ACM SIGMOD '12*, pages 13–24, 2012.