Fractional greedy and partial restreaming partitioning: New methods for massive graph partitioning.

Ghizlane Echbarthi, Hamamache Kheddouci

Fractional Greedy and Partial Restreaming Partitioning : New Methods For Massive Graph Partitioning

Ghizlane ECHBARTHI
University Lyon1
Lyon, France
Email: ghizlane.echbarthi@etu.univ-lyon1.fr

Hamamache KHEDDOUCI
University Lyon1
Lyon, France
Email: hamamache.kheddouci@univ-lyon1.fr

***Abstract*—Graph partitioning is an important challenging problem when performing computation tasks over large distributed graphs; the reason is that a good partitioning leads to faster computations. In this work, we first introduce a new heuristic for streaming partitioning and show that it outperforms the state-of-the-art heuristics for streaming partitioning, leading to exact balance and lower cut. Secondly, we introduce the partial restreaming partitioning which is a hybrid streaming model allowing only several portions of the graph to be restreamed while the rest is to be partitioned on a single pass of the data stream. We show that our method yields partitions of similar quality than those provided by methods restreaming the whole graph *(e.g ReLDG, ReFENNEL)*, while incurring lower cost in running time and memory since only several portions of the graph will be restreamed.**

I. Introduction

Actual graph datasets are massive. The World Wide Web consists of trillions of unique links [1], Facebook contains over one billion active users [2], Twitter consists of millions of users, biological networks are also of similar sizes, *e.g* protein interaction networks, to name but a few.
However, the growing size of graph datasets makes it challenging for performing usual computations over graphs, *e.g* community detection, counting triangles, identifying protein associations as well as many other computation tasks. A popular solution to this problem is to divide the graph over several clusters and then run parallel algorithms to perform computations.

Graph partitioning is a NP-hard problem aiming to divide a graph dataset into distinct sets under the constraints of equilibrating the clusters' size and minimizing the number of edges crossing the clusters, this partitioning strategy is well known as the balanced graph partitioning problem. Many platforms for graph processing are proposed such as PEGASUS [3], Pregel [4] and Graphlab [5] that use Hash partition as a default partitioner where vertices are assigned to clusters uniformly at random and where the only goal is to equilibrate the clusters' size. Hash Partition is careless about the graph structure, and its partitioning results may be sub-optimal in terms of edges cut. As seen in previous works [6], [7], a good partitioning leads to fast computations, the reason is that the balanced sizes of clusters make sure that each processor is assigned the same amount of data and in most cases, the minimized crossing edges minimizes the network overhead. In fact, more sophisticated heuristics for graph partitioning are required in order to optimize computations.

A. Streaming graph partitioning:

In the setting of dynamic graphs, graph partitioning problem is called streaming graph partitioning where the graph is serially processed, with nodes arriving in a data stream with a certain order : Random, Breadth first search or Depth first search. Once a vertex is loaded,a program has to decide about the cluster in which the vertex will be placed. This program is called the partitioner.

B. Application:

Graph Partitioning is widely applied in distributed graph computation systems. Many systems are proposed: Pregel [4], GraphLab [5], Microsoft's Trinity [8] and Horton [9] to name a few. But these systems use a Hash method to distribute vertices over clusters, which means that the partitioning is done absolutely at random and does not care about the graph structure. This is equivalent to using a random cut for partitioning which leads to balanced partitions. However, random cut will lead to slower computations due to the communication cost. To deal with this, another heuristics for partitioning should be used in order to optimize compuations. Fortunately, the existing systems support custom partitioning heuristics instead of Hash method, so it is feasible to substitute the Hash method with another more sophisticated heuristic.

C. Our contribution:

In this paper, we are presenting a new heuristic for balanced graph partitioning called Fractional Greedy. We show that it yields high quality partitions with exact balanced parts' size and lower edges cut. At the same time, we introduce the partial restreaming partitioning which is a hybrid model of data streaming. In this model a portion of graph dataset is restreamed several times, and the rest of the graph dataset is streamed once, as in the streaming model used in [6], [7]. The

main strength of this proposed method is avoiding restreaming the whole graph to minimize the runtime and the memory cost. In the proposed partial restreaming model, at most half of the graph is restreamed, this leads to faster processing and require only half of graph nodes to be kept in memory for subsequent iterations of the restream. We show that by restreaming only a part of the graph dataset we obtain a good partition as in the setting of fully restreaming the graph dataset [10]. We use LDG [6] and Fennel [7] and our proposed heuristic FG for partitioning. The three heuristics were adapted for our streaming model.

We also introduce the partial selective restreaming partitioning, which follows the same idea behind the partial restreaming partitioning but in this case, we select the portions that we wish to restream depending on their average degree and density. This selective method perform well independently of the order in which the vertices arrive.

The rest of the paper is organized as follows. In Section 2, we discuss the related work. In Section 3 we present our proposed heuristic and our partial restreaming approach. In Section 4 we present the evaluation set up. Section 5 presents and discusses the main results. In Section 6 we conclude.

II. Related work

A. Balanced Graph Partitioning problem

K-way graph partitioning is known to be a NP-hard problem [11]. It asks to divide a given graph into k balanced shards while minimizing the edges between these shards. In other words, given a graph $G = (V, E)$ with $|V| = n$, the balanced k-partitioning aims to partition V into k subsets, V_1, V_2, V_k such that $V_i \bigcap V_j = \emptyset$ for i and $\bigcup V_i = V$ and the number of edges of E whose incident vertices belong to different subsets is minimized. When k = 2, it is the problem of minimum bisection which is also a NP-hard problem [12], with an approximation algorithm presented in [13] with a polylogarithmic approximation guarantee.

The k-way graph partitioning is very important in distributed computation systems and it directly affects their performance: the k-partitioning aims to reduce the overall runtime of the application, since it assigns each processor or machine the same amount of data while minimizing the parallel overhead by minimizing the number of edges cut. Due to this importance, many heuristics achieving good performance have been proposed. Among the offline algorithms, we focus on METIS which is a multilevel method for graph partitioning [14]. The offline methods are not suitable anymore for processing huge graphs, this leads to develop new methods which can handle graphs of billions of nodes: online algorithms are another type of methods for graph partitioning which use a streaming model to partition a graph [6], [7].

Recent work by Nishimura *et al.* [10] introduces the restreaming graph partitioning, which consists in several passes of the data stream while capitalizing on results of previous streams. Ugander *et al.* [15] presented the problem of balanced label propagation partitioning, aiming to partition a graph into k balanced parts using the label propagation method after a partitioning initialization to ensure high quality partitions in massive graphs. Duong *et al.* [16] introduced the network sharding problem which consists in using a strategy of celebrities node replication over shards in order to minimize the number of shards queried while performing computations. Among online algorithms, we focus on Linear Deterministic Greedy [6], FENNEL [7] and restreaming algorithms [10].

B. Offline methods

Offline methods require complete information about the graph to partition. There are many methods for graph partitioning such as spectral methods, known to produce excellent partitions for a wide class of problems despite their high expense. Geometric methods use geometric information about the graph to partition it. Geometric algorithms are known to be fast but usually produce partitions of worse quality than those of spectral methods. There is also the local spectral partitioning methods like EvoCut [17], but it still require information about large portions of the graph and perform large computation after loading the graph data. Among the offline methods, we focus on METIS [14] which is a fast multilevel method and it is used as a reference for comparing offline and online partitioning methods.

C. Online methods

Unlike offline methods, the online methods for partitioning do not require full information about the graph. In fact, this property makes the online methods the most suitable for processing massive graphs. To the best of our knowledge, online partitioning methods in streaming context were introduced by Stanton and Kliot [6]. Tsourakakis *et al.* [7] presented FENNEL as an online streaming partitioning method. These methods consist in single shot stream of dataset. Afterward, Nishimura *et al.* [10] introduced the restreaming partitioning problem, consisting in several passes of stream dataset.

1) The streaming model: Recent work in streaming partitioning [6], [7], [10] adopts almost the same streaming graph model in which vertices arrive, each with its adjacency list. The vertices arrive in a certain order: Random, BFS or DFS [18], [7], and once the vertex is assigned to a shard, it is never replaced afterwards. The streaming graph model consists in a cluster of k machines each one of capacity C such that the total capacity of the k machines can hold the whole graph. When the vertex arrives in a stream, a partitioner must decide in which one of the k machines the vertex must be placed.

2) One Pass Streaming Heuristics:

1) *Linear Deterministic Greedy.* In [6], Stanton and Kliot proposed 10 different online heuristics for partitioning graphs. The most effective heuristic which performs well over all datasets they have used is Linear Deterministic Greedy LDG. The goal of LDG is to assign a vertex in a shard which contains the most of its neighbors while avoiding overloading shards. LDG yields the best results in terms of edges cut in FEM datasets, this in due to the structure of FEMs, their edges are highly local which make it possible to have very good partitions.

2) *FENNEL.* Recent work by Tsourakakis *et al.* [7], introduce another greedy heuristic approach for graph partitioning FENNEL. This heuristic is very performant, it yields very good quality partitions of low fraction of edge cut and balanced parts.

3) Multi-passes streaming Heuristics: In [10], Nishimura *et al.* introduced the problem of restreaming partitioning which consists in several passes of dataset stream. In the restreaming framework considered, subsequent streams of LDG and FENNEL have access to the result of previous streams [10]. In particular, Joel Nishimura et al. chose to make a restreaming version of the two single shot streaming partitioning presented earlier LDG and FENNEL denoted by reLDG and reFENNEL. Partition quality yielded by restreaming methods competes with the partitions found by METIS, the offline method presented earlier. However, if restreaming partitioning leads to exact balance - guaranteed for FENNEL - it can be too expensive for partitioning massive graphs, since it must restream the whole graph several times. Thus, the tradeoff is to be done between computational cost and quality of partitions.

III. Proposed Methods

This section presents our main contributions. In section 3.1 we present our streaming partitioning heuristic called Fractional greedy. In section 3.2 we introduce the partial restreaming partitioning: first we present the partial restreaming model used, then we present the instance of the partial restreaming partitioning model using Linear Deterministic Greedy LDG [6], Fennel [7] and Fractional Greedy as heuristics for partitioning. Finally, we give another variant of partial restreaming, called Selective Partial Restreaming partitioning where portions to be restreamed are selected depending on their degree and density.

Notations. We will be using the following notation throughout the paper. We consider a simple undirected graph $G = (V, E)$, let$|V| = n$ be the number of vertices in G and $|E| = m$ be the number of edges of G. Let the current vertex loaded be v, and $N(v)$ represents his neighbors. k is the number of clusters or parts we wish to divide the graph to. s is the number of the streaming iterations. Let $P^t = (S_1^t, , S_k^t)$ be a partition of the graph. $S_i, \cdots, S_k$ are called clusters such that $S_i \subseteq V$ and $S_i \cap S_j = \emptyset$ for every $i \neq j$. $P^{t-1} = (S_1^{t-1}, \cdots S_k^{t-1})$ represents the partition obtained from the precedent stream. Let $e(S, S - V)$ be the set of edges with ends belonging to different clusters. We define $\lambda = |e(S, S - V)|/m$ as the fraction of edge cut and it should be minimized during partitioning. We define ρ as the maximum load normalized, it expresses the balance between clusters' size and we have $\rho = \frac{maximumload}{n/k}$, $maximumload$ representing the size of the biggest cluster. Each cluster S_i is of size C. In our work we set $C = \lceil n/k \rceil$. C represents also the size of the portion to be restreamed. We can choose that several portions of the graph should be restreamed, then β is the number of portions to be restreamed (each portion is of size C).

A. Fractional Greedy: a streaming partitioning heuristic

Our proposed heuristic is designed for the constrained graph partitioning problem, which aims first to balance the size of the parts and then minimize the crossing edges between parts [19]. We formally define the Fractional Greedy heuristic by maximizing the following objective function:

$$f(v, P_i^t) = (P_i^t \cap N(v)) - g(P_i^t) \tag{1}$$

In other words, foreach vertex loaded, we compute the index ind of the part (or the cluster) as follows:

$$ind = \underset{i}{argmax}\ (P_i^t \cap N(v)) - g(P_i^t) \tag{2}$$

f has two components: one term that computes the intra-parts edges and the second g, that computes the penalization cost depending on the size of the part. The main idea is to assign the loaded vertex to a part that contains the most of its neighbors and doesnt reach the maximal size. g is defined as:

$$g(P_i^t) = \frac{1}{1 - \frac{|P_i^t|}{C}} \tag{3}$$

The cost function g penalizes the parts of large size. In the case of a vertex v, we compute the objective function for each one of the k parts, if the parts that have the most neighbors of v had reached the maximal size C, then this part should not be selected or the corresponding value of the objective function should not be maximal. Thats why we choose a cost function that penalizes rapidly parts of large sizes even though it contains the most neighbors of v.

B. Partial Restreaming Partitioning

Partial Restreaming Model:

We consider a simple streaming model as described in Section 2, where vertices arrive in a random order with their adjacency lists, the heuristic used for partitioning must make a decision about the cluster to place the current vertex. In the partial restreaming model, two major phases exist: the restreaming phase and the one pass streaming phase. Namely, a first loaded portion of the graph dataset of size $\beta * C$ is going to be restreamed and the rest is going to be processed in the simple streaming phase. In other words, in this model, multi-passes of the stream is allowed for only a part of the dataset. Let P^t be the partition obtained at time t, P^{t-1} represents the partition obtained at the precedent iteration of the restream. When we attain the number of restreaming iterations allowed for the portion concerned, we continue streaming the rest of the graph dataset normally, such that we dont use information about the last partitioning to build a new one. In the section below we describe the partitioning heuristics used in our model.

Partial Restreaming Partitioning :

As a partitioning strategy, we use the state-of-the-art heuristics [6], [7] in order to compare them with our proposed

heuristic FG. In this section we describe how we adapt these heuristics (LDG, FENNEL, FG) to our partial restreaming model.

1) Linear Deterministic Greedy
$LinearDeterministicGreedyLDG$ is the best performing heuristic in [6]. It greedily assigns the vertices to clusters while adding a penalization for big clusters to emphasize balance. LDG assigns vertices to clusters that maximize:

$$LDG = \underset{i}{argmax}\ (P_i^t \cap N(v)) * (1 - \frac{|P_i^t|}{C}) \quad (4)$$

In our streaming model, we consider 2 phases: the restreaming phase and the simple streaming phase, which consists in one pass. In the first phase, the LDG function will use information about the last partitioning to decide about the placement of the current vertex such that:

$$PartLDG = \underset{i}{argmax}\ (P_i^{t-1} \cap N(v)) * (1 - \frac{|P_i^t|}{C}) \quad (5)$$

Where $PartLDG$ makes a reference to partial LDG for partially restreaming LDG. In the second phase (the one pass streaming phase) the vertices are assigned following the LDG function as follows:

$$PartLDG = \underset{i}{argmax}\ (P_i^t \cap N(v)) * (1 - \frac{|P_i^t|}{C}) \quad (6)$$

2) Fennel
$Fennel$ yields partitions of good quality compared to LDG, with lower fraction of edge cut and also emphasizes balance [7]. We adapt Fennel function to the two phases of our streaming model. In the first restreaming phase, $PartFennel$ is as follows:

$$PartFennel = \underset{i}{argmax}\ (P_i^{t-1} \cap N(v)) - \alpha\gamma|P_i^t|^{\gamma-1} \quad (7)$$

Where $PartLDG$ makes a reference to partial $Fennel$ for partially restreaming $Fennel$. While in the second phase, $PartFennel$ is as follows:

$$PartFennel = \underset{i}{argmax}\ (P_i^t \cap N(v)) - \alpha\gamma|P_i^t|^{\gamma-1} \quad (8)$$

The parameter setting of γ will be presented in Section 4.

3) Fractional Greedy [1]
The adaptation of FG to the partial restreaming model is as follows:
In the restreaming phase, PartFG corresponds to:

$$PartFG = \underset{i}{argmax}\ (P_i^{t-1} \cap N(v)) - \frac{1}{1 - \frac{|P_i^t|}{C}} \quad (9)$$

And in the second phase, PartFG is defined as:

$$PartFG = \underset{i}{argmax}\ (P_i^t \cap N(v)) - \frac{1}{1 - \frac{|P_i^t|}{C}} \quad (10)$$

[1]Refer to Section 3.1 for FG description.

Selective Partial Restreaming Partitioning:

Instead of restreaming the first loaded portion of the graph dataset, we try to select portions of size C that would lead to a good quality partitioning. We set degree and density parameters to be the criteria for selecting portions to be restreamed in the first phase of the model. In other words, when a portion is loaded, we check its average degree and its average density, if it is higher than the average degree (respectively average density) of the whole graph, we select this portion for restreaming, otherwise it will be processed in the second phase of one pass streaming.
selection criteria. In order to select a portion for restreaming, two criteria are considered:

1) Average degree : The average degree of a portion is the average of vertices degrees inside the portion. Notice that the degree of a vertex is the number of its neighbors no matter they are inside or outside the portion concerned. We take the average degree as a criterion to make sure that the portion which is going to decide for the partitioning of the graph must influence the partitioning decision of a large number of vertices.
2) Average density : The average density of a portion represents the number of edges inside it. It is important to have edges inside the portion to make better partitioning decision as the objective functions used make decision depending on edges, otherwise the partitioning of the portion will be done at random and it will definitly lead to lower quality partitioning of the whole graph.

The average degree and density of the whole graph is an information which is not always available, we can substitute this by progressively adding degree and density information as the data is loaded. A portion with high density and high degree vertices should act like a kernel to yield partitions of good quality. In fact, portions with vertices having high degree would attract and influence the partitioning of large number of other vertices, and the density criterion inside the portion makes sure to take into consideration the edges to make better decision for the partitioning. In Section 5, we show that by selecting portions of high degree average and high density average we obtain partitions with better quality than those obtained by simply restreaming the first loaded portion.

IV. Evaluation Set Up

1) Evaluation datasets:

Two types of graph datasets were used: web and social. We test our methods on ten graph datasets listed in Table I, all obtained from the SNAP repository [20]. Vertices with 0 degree and self-loops were removed. All the graphs were made undirected by reciprocating the edges. Graph datasets were chosen in order to be small enough so that we can find offline solutions with METIS and still big enough to capture the behavior of the online heuristics.

TABLE I
GRAPH DATASETS USED FOR OUR TESTS.

Graph	$\|N\|$	$\|M\|$	**Avgdeg**	**type**
wikivote	7115	100762	14.16	social
enron	36692	183831	5.01	social
Astro ph	18771	198050	10.55	social
slashdot	77360	469180	6.06	social
Web nd	325729	1090108	3.34	web
stanford	281903	1992636	7.06	web
Web google	875713	4322053	4.93	web
Web berkstan	685230	6649470	9.7	web
Live journal	4846609	42851237	8.84	social
orkut	3072441	117185085	38.14	social

2) Methodology:

We first run FG, LDG and Fennel in one pass stream setting on our graph datatsets for $k = 40$ in order to compare the fraction of edge cut represented by λ and the balance represented by ρ. For $Fennel$, the parameter γ is set to 5 in order to give as balanced parts as those given by FG and LDG. Then we examine results of edge cut for different values of β. We begin by running $PartLDG$ and $PartFennel$ and $PartFG$ (partially restreaming LDG and partially restreaming $Fennel$ and partially restreaming FG respectively) on WebGoogle and LiveJournal for different values of β and see how the fraction of edge cut reacts to the change in β. After that, we run our methods on the ten graph datasets, for $k = 40$, $s = 10$ and $\beta = k/2$. Notice that the ordering of vertices is done at random. $\beta = k/2$ means that we are restreaming the half of the graph.
We compare our results to the whole graph restreaming methods [10] and to METIS, which represents the offline methods. Afterwards, we test the partial restreaming methods $(PartLDG, PartFennel)$ and the partial selective restreaming methods $(PartSLDG, PartSFennel)$ on seven graphs. Last but not least, we show the running time gain for the partial methods $(PartLDG, PartFennel, PartFG)$ over the ten graphs for $k = 40$ and $s = 10$. Running time gain is calculated as follows:

$$Gain_{PartLDG} = \frac{ReLDG - PartLDG}{ReLDG - LDG}$$

Where $ReLDG$ refers to the execution time of the version of LDG where the whole graph is restreamed, LDG is the one pass streaming version.

$$Gain_{PartFennel} = \frac{ReFENNEL - PartFENNEL}{ReFENNEL - FENNEL}$$

Same as $Gain_{PartLDG}$, $ReFENNEL$ refers to the execution time of the version of $FENNEL$, where the whole graph is restreamed and $FENNEL$ is the one pass streaming version.

$$Gain_{PartFG} = \frac{ReFG - PartFG}{ReFG - FG}$$

$Gain_{PartFG}$ is the runtime gain for Fractional Greedy, where $ReFG$ refers to restreaming FG and FG is the one pass streaming version. $PartFG$ is the partial restreaming version of FG. We notice that the runtime computed includes only the partitioning runtime.

V. RESULTS AND DISCUSSION

In this section, we present and discuss our results. And before we delve in the results we give a brief summary about it.
Summary of our results

- Fractional Greedy outperforms LDG and Fennel in most cases with exact balanced clusters and lower edge cut.
- Results that were obtained show that by augmenting beta, or by augmenting the size of the portion to be restreamed, we obtain better quality partition.
- By restreaming the first loaded half of a graph, we obtain partitions of similar quality than those yielded by restreaming the whole graph.
- The partial selective restreaming method preserve the quality of a partition unlike the partial methods which depend on the stream order. The reason is that, no matter the order, the selective methods always pick the good portions to be restreamed.
- Restreaming only a portion of a graph dataset incurs lower running time than restreaming the whole graph, for example, restreaming only the half of the graph takes the half running time taken by restreaming the whole graph while the results are almost the same.

Performance discussion

Our proposed heuristic Fractional Greedy outperforms LDG and Fennel in terms of balance and also of edge cut. In Table II we show our results, we see that FG yields partitions of exact balance $\rho = 1$ and lower edge cut. However, LDG outperforms FG in web-google (0.310 vs 0.308) and web-berkstan (0.368 vs 0.342). Our proposed heuristic is the most adapted for the setting of constrained graph partitioning, it yields exact balanced partitions and also minimize the edge cut.
In figure 1 we show average results of comparison between $FG, LDG, FENNEL$ and $METIS$ in terms of fraction of edge cut over 10 graph datasets. METIS

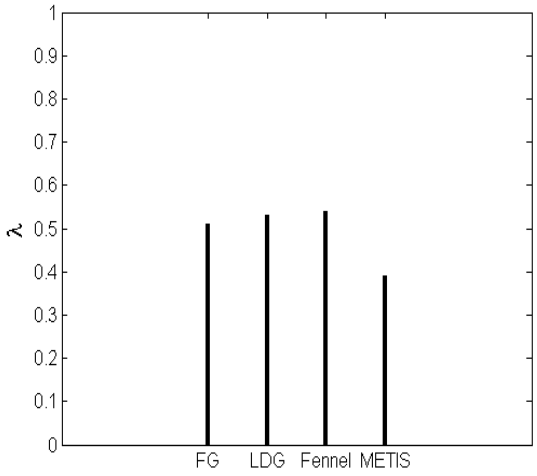

Fig. 1. Average fraction of edge cut λ for $FG, LDG, FennelandMETIS$ over ten graph datasets.

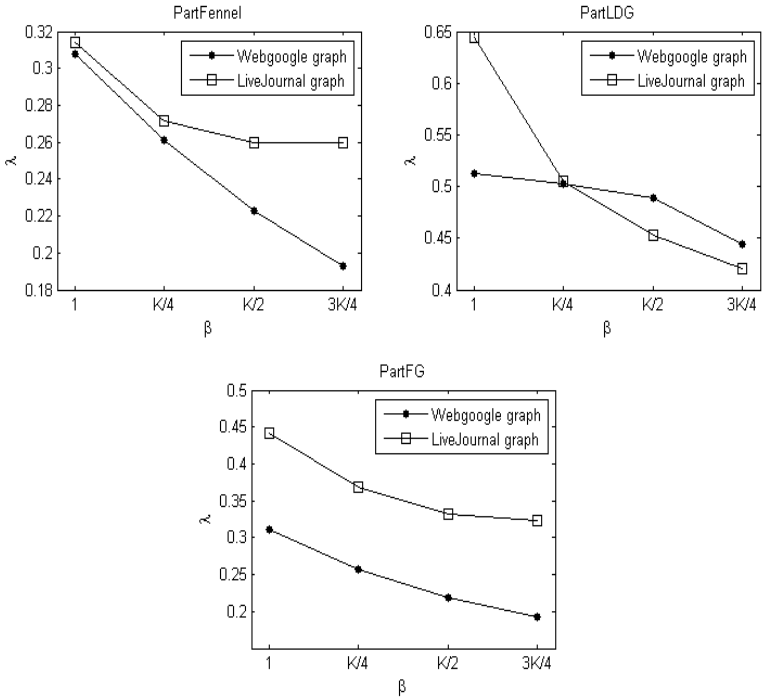

Fig. 2. Variation of λ with the growing size of portion being restreamed represented by β for Webgoogle graph and LiveJournal. On the left $PartFennel$ and on the right $PartLDG$ and at the bottom $PartFG$.

is the best performing heuristic, FG is the second bestt heruistic outeerforming LDG and Fennel.

In Figure 2, we see that the bigger β is the lower the fraction of edge cut. This shows that the information about the precedent stream iteration allows vertices to be more oriented toward the best cluster leading to lower edge cut. In other words, the more information we have about the precedent streaming iteration the better is the edge cut.

Table III shows results in terms of fraction of edge cut and balance. The difference between edge cut of fully restreaming methods and partial restreaming methods are minimal in the most cases.

Table IV shows the diffrence between the performances of partial restreaming methods and selective partial methods on seven different datasets. The results shows that in most cases, selective partial restreaming methods yields better quality partitions by leading low edge cut. However, in some datasets, it may be that portions which are conform to the criteria doesn't exist, then the selective partial method would not give the best results, that's why the simple partial restreaming method could be a good alternative for both speeding up computations and imporving partitions quality by restreaming several portions.

The Running time gain is represented in Table V. In all our graph datasets, PartFennel has an average gain of 49.93% and PartLDG 48.85% and PartFG 50.26%. It shows that partial restreaming reduces the runtime by half.

VI. Conclusion and future work

In this work, we provide new methods for the problem of balanced graph partitioning. Specifically, we introduce a new online heuristic for streaming partitioning and we show that we improve the partitions quality by partially restreaming several portions of the graph. We showed that our proposed heuristic produces partitions of significantly enhanced quality in terms of balance and edge cut: partitions that were produced are exactly balanced and have a lower edge cut.

We also introduced the partial restreaming graph partitioning that aims to take several portions of the graph and restream it several times in order to improve the partitions quality. We evaluate our proposed methods on ten different graph datasets and compare it with the state-of-the-art partitioning heuristics (Fennel and LDG). We showed that our proposed partial restreaming methods produce partitions of similar quality than those produced by fully restreaming methods with the advantage of incurring lower run time and memory cost.

We also present another variant of the partial restreaming partitioning called selective partial restreaming partitioning and showed that selecting relevant portions of the graph using degree and density as selection criteria improve the quality of the partitions.

There are several future work directions for our work. We comple this empirical study of partial restreaming partitioning by theoritical study of convergence of this method. We also envisage extending our proposed heuristic FG to handle partitioning of portions not only of vertices. We are also looking for other selection criteria to refine the partition quality in the selective partial restreaming setting.

TABLE II

FRACTION OF EDGE CUT λ AND MAXIMUM LOAD NORMALIZED ρ FOR 3 STREAMING HEURISTICS LDG AND $FENNEL$ AND FG AND METIS,(1.001) INDICATES THAT THE SLACKNESS ALLOWED IS 001. RESULTS ARE OBTAINED FOR 10 GRAPH DATASETS WHERE $k = 40$.

Graphs	FG		LDG		Fennel		Metis(1.001)	
	λ	ρ	λ	ρ	λ	ρ	λ	ρ
wikivote	0.844	1	0.867	1	0.862	1	0.822	1.001
enron	0.589	1	0.610	1	0.612	1	0.855	1.001
astro ph	0.555	1	0.619	1	0.578	1	0.535	1.001
slashdot	0.758	1	0.787	1	0.777	1	0.711	1.001
webnd	0.249	1	0.261	1	0.270	1	0.036	1.001
stanford	0.349	1	0.392	1	0.347	1.043	0.123	1.001
webgoogle	0.310	1	0.308	1	0.313	1.023	0.009	1.001
web berkstan	0.386	1	0.342	1	0.367	1.023	0.117	1.001
live journal	0.442	1	0.462	1	0.546	1.009	0.309	1.001
orkut	0.627	1	0.639	1	0.696	1.076	0.376	1.001

TABLE III

COMPARISON OF FRACTION EDGE CUT AND NORMALIZED MAXIMUM LOAD ρ FOR REFG AND PARTFG, RELDG AND PARTRELDG, REFENNEL AND PARTFENNEL. RESULTS ARE OBTAINED FOR TEN GRAPH DATASETS WHERE $k = 40$ AND $s = 10$ AND $\beta = k/2$.

Graph	ReFG		PartFG		ReLDG		PartLDG		ReFENNEL		PartFENNEL	
	λ	ρ	λ	ρ	λ	ρ	λ	ρ	λ	ρ	λ	ρ
wikivote	0.812	1	0.828	1	0.835	1	0.850	1	0.813	1.023	0.826	1.022
enron	0.479	1	0.509	1	0.475	1	0.507	1	0.476	1.098	0.482	1.087
astro ph	0.433	1	0.475	1	0.418	1	0.501	1	0.413	1.019	0.443	1.019
slashdot	0.711	1	0.705	1	0.713	1	0.722	1	0.703	1.041	0.692	1.106
webnd	0.164	1	0.207	1	0.113	1	0.207	1	0.143	1.048	0.193	1.056
stanford	0.200	1	0.267	1	0.204	1	0.319	1	0.193	1.025	0.216	1.109
webgoogle	0.163	1	0.219	1	0.161	1	0.217	1	0.160	1.087	0.222	1.012
web berkstan	0.241	1	0.276	1	0.212	1	0.276	1	0.254	1.037	0.282	1.073
live journal	0.325	1	0.331	1	0.313	1	0.331	1	0.330	1.006	0.319	1.018
orkut	0.398	1	0.503	1	0.395	1	0.503	1	0.410	1.005	0.451	1.0173

TABLE IV

COMPARISON OF FRACTION OF EDGE CUT λ AND MAXIMUM LOAD NORMALIZED ρ FOR PARTIAL RESTREAMING METHODS AND SELECTIVE PARTIAL RESTREAMING METHODS $PartFG\ vs\ PSelectFG$, $PartLDG\ vs\ PSelectLDG$, $PartFennel\ vs\ PSelectFennel$. RESULTS ARE OBTAINED FOR 7 GRAPH DATASETS WHERE $k = 40$.

Graphs	PSelectFG		PartFG		PSelectLDG		PartLDG		PSelectFENNEL		PartFENNEL	
	λ	ρ	λ	ρ	λ	ρ	λ	ρ	λ	ρ	λ	ρ
wikivote	0.826	1	0.828	1	0.849	1	0.850	1	0.845	1.005	0.826	1.022
enron	0.503	1	0.509	1	0.502	1	0.507	1	0.509	1.003	0.482	1.008
astro ph	0.475	1	0.475	1	0.501	1	0.501	1	0.468	1.008	0.443	1.019
slashdot	0.703	1	0.705	1	0.722	1	0.722	1	0.716	1.011	0.692	1.106
webnd	0.213	1	0.207	1	0.214	1	0.207	1	0.217	1.013	0.193	1.056
stanford	0.271	1	0.267	1	0.339	1	0.319	1	0.258	1.024	0.216	1.109
webgoogle	0.189	1	0.219	1	0.188	1	0.217	1	0.187	1.009	0.222	1.012

TABLE V

RUNNING TIME GAIN COMPUTED FOR $PartLDG$ AND $PartFennel$ AND $PartFG$ OVER EXECUTIONS ON TEN GRAPHS WITH $k = 40$ AND $s = 10$.

Graphs	**PartLDG Gain**	**PartFennel Gain**	**PartFG Gain**
Wikivote	47.2%	58.5%	45.9%
Enron	50%	48.3%	47.6%
Astro ph	44.5%	44%	47.4%
Slashdot	39.6%	47.5%	45.2%
Web nd	51.4%	49.5%	56.2%
Stanford	54.5%	52.4%	56.8%
Web google	57.5%	52.7%	56.1%
Web berkstan	50.6%	49.6%	65%
Live journal	40.5%	45.6%	42.1%
Orkut	52.7%	51.2%	39.9%

REFERENCES

[1] "http://googleblog.blogspot.co.uk/2008/07/we-knew-web-was-big.html."

[2] "http://thenextweb.com/facebook/2013/10/30/facebook-passes-1-19-billion-monthly-active-users-874-million-mobile-users-728-million-daily-users/."

[3] U. Kang, C. E. Tsourakakis, and C. Faloutsos, "Pegasus: A peta-scale graph mining system implementation and observations."

[4] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski, "Pregel: A system for large-scale graph processing," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '10. New York, NY, USA: ACM, 2010, pp. 135–146. [Online]. Available: http://doi.acm.org/10.1145/1807167.1807184

[5] Y. Low, J. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin, and J. M. Hellerstein, "Graphlab: A new parallel framework for machine learning," in *Conference on Uncertainty in Artificial Intelligence (UAI)*, Catalina Island, California, July 2010.

[6] I. Stanton and G. Kliot, "Streaming graph partitioning for large distributed graphs," in *18th ACM SIGKDD -KDD '12*.

[7] C. Tsourakakis, C. Gkantsidis, B. Radunovic, and M. Vojnovic, "Fennel: Streaming graph partitioning for massive scale graphs," in *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, ser. WSDM '14. New York, NY, USA: ACM, 2014, pp. 333–342. [Online]. Available: http://doi.acm.org/10.1145/2556195.2556213

[8] "http://research.microsoft.com/trinity, jan 2012."

[9] "http://research.microsoft.com/ldg, jan 2012."

[10] J. Nishimura and J. Ugander, "Restreaming graph partitioning: Simple versatile algorithms for advanced balancing," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '13. New York, NY, USA: ACM, 2013, pp. 1106–1114. [Online]. Available: http://doi.acm.org/10.1145/2487575.2487696

[11] K. Andreev and H. Räcke, "Balanced graph partitioning," in *Proceedings of the Sixteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, ser. SPAA '04. New York, NY, USA: ACM, 2004, pp. 120–124. [Online]. Available: http://doi.acm.org/10.1145/1007912.1007931

[12] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1979.

[13] U. Feige and R. Krauthgamer, "A polylogarithmic approximation of the minimum bisection," *SIAM J. Comput.*, vol. 31, no. 4, pp. 1090–1118, Apr. 2002. [Online]. Available: http://dx.doi.org/10.1137/S0097539701387660

[14] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM J. Sci. Comput.*, vol. 20, no. 1, pp. 359–392, Dec. 1998. [Online]. Available: http://dx.doi.org/10.1137/S1064827595287997

[15] J. Ugander and L. Backstrom, "Balanced label propagation for partitioning massive graphs," in *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, ser. WSDM '13. New York, NY, USA: ACM, 2013, pp. 507–516. [Online]. Available: http://doi.acm.org/10.1145/2433396.2433461

[16] Q. Duong, S. Goel, J. Hofman, and S. Vassilvitskii, "Sharding social networks," in *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, ser. WSDM '13. New York, NY, USA: ACM, 2013, pp. 223–232. [Online]. Available: http://doi.acm.org/10.1145/2433396.2433424

[17] R. Andersen and Y. Peres, "Finding sparse cuts locally using evolving sets," in *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, ser. STOC '09. New York, NY, USA: ACM, 2009, pp. 235–244. [Online]. Available: http://doi.acm.org/10.1145/1536414.1536449

[18] G. Karypis and V. Kumar, "Multilevel graph partitioning schemes," in *Proc. 24th Intern. Conf. Par. Proc., III*. CRC Press, 1995, pp. 113–122.

[19] C.-E. Bichot and P. Siarry, "Graph Partitioning," Sep. 2011, iSTE-Wiley, 368 pages, 13 chapters, ISBN 978-1-84821-233-6. [Online]. Available: http://liris.cnrs.fr/publis/?id=5317

[20] "http://snap.stanford.edu/data/."