Partial Restreaming Approach for Massive Graph Partitioning.

Ghizlane Echbarthi, Hamamache Kheddouci

Partial Restreaming Approach 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 a challenging and highly important problem when performing computation tasks over large distributed graphs; the reason is that a good partitioning leads to faster computations. In this work, 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)* [1], 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 trilions of unique links [2], Facebook contains over one billion of active users [3] and Twitter consists of millions of users, biological networks are also of similar sizes, like protein interaction networks, to name but a few. However, the growing size of graph datasets makes it challenging when performing usual computations over graphs, such as community detection, counting triangles, identifying protein associations as well as many other computation tasks. One solution to this problem is to divide the graph over several clusters and then run parallel algorithms to perform computations. Graph partitioning is an NP-hard problem aiming to divide a graph dataset into distinct sets under the constraints of equilibrating the cluster sizes and minimizing the number of edges crossing the clusters, this is the balanced graph partitioning problem. As seen in previous works [4], [5] , 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 the minimized crossing edges minimize the overhead network. In the setting of dynamic graphs, this problem is called streaming graph partitioning where the graph is serially processed, each vertex loaded is assigned to a cluster.

Our contribution. In this paper, we introduce the partial restreaming partitioning; it is a hybrid model of data streaming where a portion of size C is restreamed several times, and the rest of the graph dataset is streamed once, like in the streaming model used in [4], [5]. The main strength of our proposed method avoids restreaming the whole graph dataset to minimize runtime and memory cost. We show that by restreaming at most half of the graph dataset we can obtain a good partition as in the setting of full restreaming [1]. As partitioning heuristics, we use LDG and Fennel. As an extension of partial restreaming, we introduce the selective partial restreaming partitioning, in which several portions of the graph dataset are selected for restreaming according to their average degree and density. We show that the selective method improves the partition quality.

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

II. Related work

Graph partitioning is a well studied problem. Many of methods and heuristics have been proposed in the litterature, each one presenting strengths and weaknesses. But in the setting of streaming partitioning, only few heuritics are proposed, and were first introduced by Stanton *et al.* in 2012 [4]. Stanton *et al.* proposed ten different heuristics and found that the best performant heuristic was Linear Deterministic Greedy LDG.

Since then, streaming partitioning methods become the trend in partitioning graphs especially when dealing with huge datasets, the reason is that these methods are well suited for big graphs since they process nodes on the fly and do not incur as high computation cost as in the classic partitioning methods.

After Stanton *et al.*, Tsourakakis *et al.* came up with a new streaming partitioning heuristic called FENNEL [5]. FENNEL relaxes the balance constraint unlike LDG, producing partitions with lower edges cut but higher balance slackness.

In 2013, Nishimura *et al.* [1] proposed the restreaming partitioning, which improves partitions quality while preserving the balance constraint.

Though classical methods are not suited anymore for processing huge graphs, we cite METIS [6], a multilevel method for graph partitioning yielding high quality partitions. METIS have been usually used in order to compare online and offline methods.

1) The streaming model: Recent work in streaming partitioning [1], [4], [5] adopts almost the same streaming graph model in which vertices arrive, each with its adjacency list. The vertices arrive in a certain order: Random, Breath First

Search or Depth First Searh [7], [5], 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 a vertex is loaded, a partitioner must decide in which one of the k machines the vertex must be placed.

III. Proposed Approach

In this section, we introduce the partial restreaming partitioning. In Section 3.1 we first present the partial restreaming model used. In Section 3.2 we present the instance of the partial restreaming partitioning model using Linear Deterministic Greedy LDG [4] and Fennel [5] as heuristics for partitioning. In Section 3.3, 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. $S_i, \cdots, S_k$ is 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 $\lambda = \frac{|e(S,S-V)|}{m}$ as the fraction of edge cut. We note that λ should be minimized during partitioning. Each cluster S_i is of size C. In our work we set $C = \lceil \frac{n}{k} \rceil$. 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. Partial Restreaming Model

We consisder 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, multi-passes of the stream are allowed only for a graph subset. 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 in a single pass of data stream. In the section below we describe the partitioning heuristics used in our model.

B. Partial Restreaming Partitioning

As a partitioning strategy, we use the state-of-the-art heuristics [4], [5] . In this section we describe how we adapt these heuristics to our partial restreaming model.

1) Linear Deterministic Greedy : $LinearDeterministicGreedyLDG$ is the best performing heuristic in [4]. It greedily assigns the vertices to clusters while adding a penalization for big clusters to emphasize balance. LDG assigns vertices to clusters that maximize:

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

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:

$$PartLDG = \underset{i}{argmax}\ (P_i^{t-1} \cap N(v)) * (1 - \frac{|P_i^t|}{C}) \quad (2)$$

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:

$$PartLDG = \underset{i}{argmax}\ (P_i^t \cap N(v)) * (1 - \frac{|P_i^t|}{C}) \quad (3)$$

2) Fennel: Like done for LDG, we adapt Fennel function to the two phases of our streaming model. In the first restreaming phase, $Fennel$ is as follows:

$$PartFennel = \underset{i}{argmax}\ (P_i^{t-1} \cap N(v)) - \alpha\gamma(|P_i^t|)^{\gamma-1} \quad (4)$$

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

$$PartFennel = \underset{i}{argmax}\ (P_i^t \cap N(v)) - \alpha\gamma(|P_i^t|)^{\gamma-1} \quad (5)$$

C. Selective Partial Restreaming Partitioning

Instead of restreaming the first loaded portion of the graph dataset, we select *relevant* portions of size C that would lead to a good quality partitioning. We set degree and density parameters as 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 criterions. 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.

The average degree and density of the whole graph aren't 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 influence the partitioning of a large number of 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 in ten graph datasets listed in Table I, all obtained from the SNAP repository [8]. Vertices with 0 degree and self-loops were removed. All the graphs were made undirected by reciprocating the edges.

2) Methodology:
We run our methods on the ten graph datasets, we choose to fix expriments parameters as follows: $k = 40$, $s = 10$ and $\beta = k/2$ (which means that we are restreaming the half of the graph). Notice that the ordering of vertices is done at random. We first compare our methods to the single pass methods and then to the full restreaming methods [1]. Afterwards, we compare the partial restreaming methods $(PartLDG, PartFennel)$ and the selective methods $(PSelectLDG, PSelectFennel)$ on five graph datasets. We run 5 executions on 5 different random orders and show the results obtained.
In order to see how the partition quality reacts to the changing portion size being restreamed, we examine the results for different values of β. We run $PartLDG$ and $PartFennel$ (partially restreaming LDG and partially restreaming $Fennel$ respectively) on WebGoogle and LiveJournal for different values of β.
Ultimately, we show the running time gain for the partial methods $(PartLDG, PartFennel)$ 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.

V. Results and discussion

In this section, we present and discuss our results. Notice that the metrics presented in tables II, III and IV represent the fraction of edge cut λ. And before we delve in the results we give a brief summary about it.

1) **Summary of our results**
- Results that were obtained show that by augmenting β, or by augmenting the size of the portion to be restreamed, we obtain a 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 dataset.
- The partial selective restreaming method improves the partition quality.
- Restreaming only a portion of a graph dataset incurs lower run time cost. Compared with full restreaming methods, partial methods takes half of running time in case of restreaming half of the graph dataset.

2) **Performance: discussion**
In Table II, we clearly see that the partial methods outperforms the single pass methods over all the graph datasets and it improves the partition quality.
In table III we observe that our partial methods yields almost the same fraction of edge cut as in full restreaming methods, with an average difference of 4%.
In Table IV, we run PartFENNEL and PartLDG on 5 different random orders and compare it to PSelectFENNEL and PSelectLDG (Partial Selective FENNEL and Partial Selective LDG). As we expected, partial selective methods outperforms the partial methods in all the 5 graphs that we tested, and it outperforms METIS also except for Astro-ph and Webnd (0.593 vs 0.535 and 0.376 vs 0.036 respectively).
In Figure 1 we see that the size of the protion restreamed influences the partition quality: the bigger β is the lower fraction of edge cut. In other words, the more information we have about the precedent streaming iteration the better is the edge cut. For Fennel, the simple partial methods outperforms the selective ones because the selective methods enhance the balance which lead to higher edge cut.

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

TABLE II
COMPARISON OF FRACTION EDGE CUT FOR LDG AND PARTRELDG, FENNEL AND PARTFENNEL. RESULTS ARE OBTAINED FOR TEN GRAPH DATASETS WHERE $k = 40$ AND $s = 10$ AND $\beta = k/2$.

Graph	LDG	PartLDG	FENNEL	PartFENNEL
wikivote	0.867	0.835	0.813	0.826
enron	0.610	0.507	0.476	0.482
astro ph	0.619	0.501	0.413	0.443
slashdot	0.787	0.722	0.777	0.703
webnd	0.261	0.207	0.270	0.193
stanford	0.392	0.319	0.347	0.216
webgoogle	0.308	0.217	0.313	0.222
web berkstan	0.342	0.276	0.367	0.282
live journal	0.462	0.331	0.546	0.319
orkut	0.639	0.503	0.696	0.451

TABLE III
COMPARISON OF FRACTION EDGE CUT FOR RELDG AND PARTRELDG, REFENNEL AND PARTFENNEL. RESULTS ARE OBTAINED FOR TEN GRAPH DATASETS WHERE $k = 40$ AND $s = 10$ AND $\beta = k/2$.

Graph	ReLDG	PartLDG	ReFENNEL	PartFENNEL
wikivote	0.835	0.850	0.813	0.826
enron	0.475	0.507	0.476	0.482
astro ph	0.418	0.501	0.413	0.443
slashdot	0.713	0.722	0.692	0.703
webnd	0.113	0.207	0.143	0.193
stanford	0.204	0.319	0.193	0.216
webgoogle	0.161	0.217	0.160	0.222
web berkstan	0.212	0.276	0.254	0.282
live journal	0.313	0.331	0.330	0.319
orkut	0.395	0.503	0.410	0.451

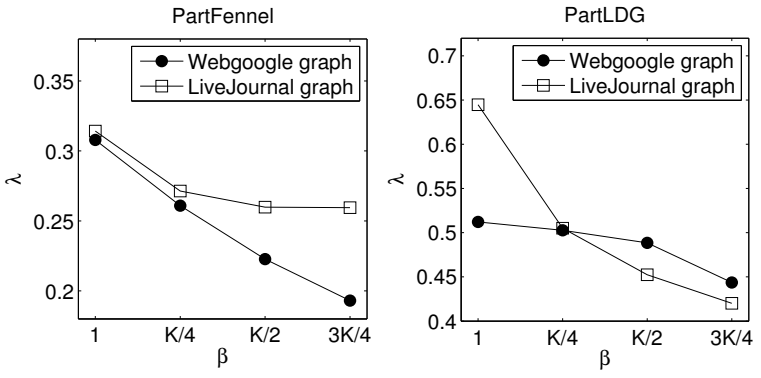

Fig. 1. Variation of λ with the growing size of portion being restreamed represented by β for Webgoogle graph and LiveJournal. On the left $PartFennel$ and on the right $PartLDG$.

Running time gain is represented in Table V, in all our graph datasets, PartFennel have a gain of 49.93% and PartLDG 48.85%. Those results are obtained while restreaming the half of the graph datasets, which is normal since the gain is equal to 50% approximately.

VI. CONCLUSION AND FUTURE WORK

We have demonstrated that by restreaming only subsets of a graph dataset we actually obtain as good quality partitions as those yielded by restreaming the whole graph dataset, while fastening the running time partitioning and minimizing the cost in memory. We also showed that by selecting portions of the graph having high average degree and density we improve the partition quality. Good partitioning is a result of having a good kernel in clusters. In future work, we plan to refine our selection criteria for selecting portions worth restreaming.

TABLE IV

FRACTION OF EDGE CUT FOR PARTIAL METHODS $PartLDGandPartFENNEL$ VERSUS PARTIAL SELECTIVE METHODS $PSelectLDGandPSelectFENNEL$ AND METIS,(1.001) INDICATES THAT THE SLACKNESS ALLOWED IS 001. RESULTS ARE OBTAINED FOR 5 GRAPH DATASETS WHERE $k = 40$ AND $s = 10$ AND $\beta = k/2$ FOR 5 DIFFERENT RANDOM STREAM ORDERS.

Graphs	PartLDG	PSelectLDG	PartFennel	PSelectFennel	Metis(1.001)
wikivote	0.875	0.875	0.676	0.667	0.822
enron	0.702	0.687	0.515	0.470	0.855
Astro ph	0.736	0.733	0.602	0.593	0.535
slashdot	0.830	0.829	0.596	0.533	0.711
Web nd	0.471	0.450	0.378	0.376	0.036

TABLE V

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

Graphs	Partial ReLDG Gain	Partial ReFennel Gain
Wikivote	47.2%	58.5%
Enron	50%	48.3%
Astro ph	44.5%	44%
Slashdot	39.6%	47.5%
Web nd	51.4%	49.5%
Stanford	54.5%	52.4%
Web google	57.5%	52.7%
Web berkstan	50.6%	49.6%
Live journal	40.5%	45.6%
Orkut	52.7%	51.2%

REFERENCES

[1] 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

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

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

[4] I. Stanton and G. Kliot, "Streaming graph partitioning for large distributed graphs," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '12. New York, NY, USA: ACM, 2012, pp. 1222–1230. [Online]. Available: http://doi.acm.org/10.1145/2339530.2339722

[5] 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

[6] 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

[7] Karypis and Kumar, "Multilevel graph partitioning schemes," pp. 113–122, 1995.

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