Online Partitioning of Multi-Labeled Graphs*

Ioanna Filippidou
Athens University of Economics and Business
76 Patission Street
Athens, Greece
filippidoui@aueb.gr

Yannis Kotidis
Athens University of Economics and Business
76 Patission Street
Athens, Greece
kotidis@aueb.gr

ABSTRACT

Graph partitioning is an old problem that is finding renewed interest in the era of big, complex datasets and parallel computing frameworks that can benefit from a proper partitiong of big graph data across multiple nodes in a cluster. In this paper we look into a specific instance of the problem termed online graph partitioning that addresses the need to partition large graphs that do not fit in main memory. A neglected aspect of modern graph datasets is that real graphs have labels! Node labels may, for instance, correspond to categorical attributes (such as country, profession, participating groups, etc.) of the entities depicted by the vertices of the graph. Edge labels may represent different relationship types (e.g. "friend-of", "likes", etc.). In this work we first revisit the formulation of the graph partitioning problem for graphs with labels on both nodes and edges. We introduce "relation-cut", as a new metric that extends the traditional "edge-cut" metric used in graph partitioning in order to take into account the existence of different edge-types. Then, we combine this metric with a novel "label-cut" metric that takes into consideration the displacement of related nodes with similar labels across partitions. In our experiments we adapt two recent online partitioning algorithms for the new proposed metric and provide a thorough evaluation on a variety of real and synthetic graphs. Our experiments demonstrate that the proposed technique balances the generated cuts on both relations and labels on the resulting partitions.

1. INTRODUCTION

Many modern applications generate data that is naturally described via a graph model consisting of a large number of entities (depicted as graph nodes) with complex and various relation types (depicted as labeled edges) between them. Furthermore, nodes in such graphs often belong to one or more groups (denoted via node labels) that specify their behavior in a global manner. Graphs of this type can be found in several domains, including social, citation, collaboration and biological networks. With the big data explosion these graphs cannot be easily maintained in a single machine. A prevalent approach is to utilize a cluster of machines so that maintenance of and computations on these graphs can be performed in parallel. A standard solution is to split the graph equally across a number of partitions, each assigned to a different node in the cluster [11, 16, 17, 23].

The solution to this problem can be traced to the classical balanced graph partitioning problem where the goal is to minimize the number of cross partition edges (often referred to as the "edge-cut"), while keeping the number of nodes in every partition approximately even. The intuition of minimizing the cross partition edges is based on the observation that edges represent relationships between nodes from which most queries on the graph will are derived (e.g. "find the friends-of-friends of user X" in Facebook). By minimizing the cross partitions edges most queries will be performed locally, reducing inter-node communication.

The graph partitioning problem is NP-hard [2] and many algorithms utilize heuristics [6, 7, 10, 13, 14, 20] that often work well in practice. Most graph partitioning algorithms are static, meaning that they assume that the whole graph is available and fits in main memory. A recent breed of *online* graph partitioning algorithms [18, 24, 25] have emerged in order to allow partitioning of graphs that do not fit in a single machine and, thus, a static partitioner is not applicable.

Unfortunately, online partitioning algorithms concentrate on minimizing the edge-cut and ignore the various labels that exist on the graph nodes and edges. By ignoring that edges have different labels (corresponding to different relationship types), these algorithms often produce partitions that have significantly different edge-cuts, when these are broken down based on the different edge types (e.g. cross-partition edges for "friend of" and "likes" relationships). Moreover, node labels are completely ignored in this process. Node labels represent global attributes (e.g. location, age, gender, group, etc.) that are often used in queries such as "find the friends-of-friends of user X that belong to group Y". Queries on real graph networks often pose relationship and node constraints on the associated labels and it is, thus, essential that the distribution of both node and edge labels across the partitions is taken into account by the partitioning algorithm.

In this paper, we explore a partitioning scheme that is suitable for labeled graphs. We first introduce the relation-cut as an extension of the edge-cut. The new metric takes into consideration the labels (relationship types) that are associated with each edge. Then, we further introduce the notion of the "label-cut" as a metric to penalize partitioning schemes that split connected nodes with similar labels across partitions. Finally, we fuse both metrics into a formulation that takes into consideration both components. The

*This research has been co-financed by the European Union (European Social Fund ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: RECOST.

GRADES'15, May 31 - June 04 2015, Melbourne, VIC, Australia

ACM 978-1-4503-3611-6/15/05...$15.00
DOI: http://dx.doi.org/10.1145/2764947.2764950.

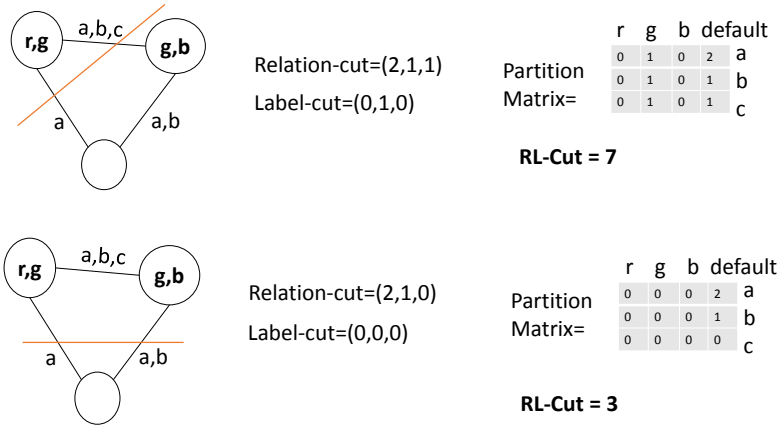

Figure 1: Example of partitioning matrix and relation-cut, label-cut calculations

proposed calculations can be easily tuned to favor the label or relation cut distribution based on the application needs.

In our experimental evaluation we adapt two well known online partitioning algorithms so that their calculations take into account our proposed metrics. Our results on various graphs indicate that our proposed technique produces partitions that reduce the induced cut, when both the relation-cut and the label-cut are taken into account.

The rest of the paper is organized as follows. In Section 2 we revisit the graph partitioning problem and introduce our proposed metrics for labeled graphs. In Section 3 we first briefly describe two online graph partitioning algorithms and then discuss the required modifications on them in order to incorporate our proposed metrics. Finally, in Section 4 we present our experiments and in Section 6 we provide concluding remarks.

2. PROBLEM DEFINITION

A graph with labels on nodes and types on edges is denoted as $G = \{V, E, L, T\}$, where V is the set of nodes, E is the set of edges, $L = \{L^1, \ldots, L^d\}$ is the set of d labels associated with the nodes in V and $T = \{T^1, \ldots, T^n\}$ is the set of n types associated with the edges in E. Each node v_i, is associated with a binary label vector $n_u = (x_i^1, \ldots, x_i^d)$, $x_i \in \{1, 0\}$, where $x_i = 1$ if the corresponding label exists in this node. Also, each edge (u, v) is associated with a type vector $e_{uv} = (t_i^1, \ldots, t_i^n)$, $t_i \in \{1, 0\}$ where $t_i = 1$ if the respective relationship type exists in this edge.[1]

We seek to partition the graph G into k disjoint groups (partitions), where $V = \bigcup_{i=1}^{k} V_i$ and $V_i \cap V_j = \emptyset$, $\forall i \neq j$ with the following objectives:

Balance Criterion: Demands that all k partitions have about equal size. It requires that, $\forall i \in \{1, \ldots k\}$ $|V_i| \leq (1 + \epsilon)\lceil |V|/k \rceil$ for some imbalance parameter $\epsilon \geq 0$. In the case of $\epsilon = 0$, the term perfectly balanced is used.

Relation-cut: A node v is a neighbor of node u if there exists an edge $(u, v) \in E$. If a node $v \in V_i$ has a neighbor $w \in V_j$, $i \neq j$ then it is called boundary node. An edge that runs between partitions is also called cut edge. The edge-cut between two subsets V_a and V_b is defined as the number of cut edges between these two sets:

$$Edge\text{-}cut(V_a, V_b) = |\sum_{u \in V_a} \sum_{v \in V_b} (u, v)|$$

In our model, where each edge is associated with a type vector $e_{uv} = (t_i^1, \ldots, t_i^n)$ we extend the notion of the edge-cut and compute the relation-cut between two subsets V_a and V_b using the norm of the sum of the vectors e_{uv} of the corresponding cut edges as follows:

$$Rel\text{-}cut(V_a, V_b) = \| \sum_{u \in V_a} \sum_{v \in V_b} e_{uv} \|_p \quad (1)$$

The sum of the e_{uv} vectors produces a new vector, where each coordinate depicts the edge-cut for the respective relationship type. The L_p norm $\|.\|_p$ of a n-dimensional vector x is defined as

$$\|x\|_p = (\sum_{i=1}^{n} |x_i|^p)^{\frac{1}{p}}$$

Common values of p are 1, 2 and ∞. L_1-norm can be used to order to minimize the mean absolute deviation, while L_2-norm minimizes the sum of square differences. L_∞-norm can be used to minimize the maximum cut value.

Given a partitioning scheme P_k that splits G into k parts, we wish to minimize the norm of the vector resulting from adding all e_{uv} across all k partitions:

$$Rel\text{-}cut(P_k) = \| \sum_{i<j} \sum_{u \in V_i} \sum_{v \in V_j} e_{uv} \|_p \quad (2)$$

Label-cut: In addition to relation-cut minimization, we wish to minimize the number of common labels between boundary nodes. The common labels between two boundary nodes $u \in V_i$ with label vector $n_u = (x_i^1, \ldots, x_i^d)$ and $w \in V_j$ with label vector $n_w = (x_i^1, \ldots, x_i^d)$ are described by the vector resulting by taking the product of n_u and n_w. This vector has 1 in each dimension that is common in both nodes.

$$(n_u * n_v)[i] = n_u[i] \times n_v[i], i = 1, \ldots d$$

The sum of these product vectors is also a vector with each dimension counting the number of a instances where the partitioning scheme creates a pair of boundary nodes with the respective label. Thus, the label-cut function between two subsets can be defined as:

$$Label\text{-}cut(V_a, V_b) = \| \sum_{u \in V_a} \sum_{w \in V_b, e_{uw} \in E} n_u * n_w \|_p \quad (3)$$

[1]Our model replaces multiple edges of different types between the same two nodes with a single edge with a compound labeling vector, thus transforming a multigraph into a graph. It is obvious that no information is lost via this (virtual) mapping.

In the general case of a partitioning P_k of the graph into k parts, we wish to minimize the norm of the sum of products between all parts using the objective function:

$$Label\text{-}cut(P_k) = \| \sum_{i<j} \sum_{u \in V_i} \sum_{w \in V_j, e_{uw} \in E} n_u * n_w \|_p \quad (4)$$

Combined relation and label cut metric: In order to combine both relationship type and node label cut functions, we construct the total partitioning matrix $P(n \times (d+1))$, where each row represents a relationship type and each column a specific label. We keep an extra column for storing the edge-cut for each relationship type so as to account for cuts between nodes with no common label. We can think of this column as representing a default label present in all nodes. A cell $x_{i,j}$ of the matrix stores the total number of cut edges between i^{th} relationship type and j^{th} node label that have occurred from a P_k partitioning. The objective function we wish to minimize in order to consider both edge types and node labels is:[2]

$$RL\text{-}cut(P_k) = \|vectorize(P)\|_p \quad (5)$$

In Figure 1, we present an example of the use of our metrics for a small graph. Each node in the depicted graph has a label vector selected from three labels (red, green, blue) and each edge has between one and three relationship types, namely (a, b, c). We calculate two different cut matrices, one for each cut presented with red lines. The figure depicts the vectors for the relation-cut and the label-cut, in each case. At the right, we calculate the corresponding RL-cut matrices and the L_1-norms, which are 7 and 3, respectively. As a result, our technique will favor the second cut that minimizes the norm.

Distribution balance criterion: There are many cases of graphs where the distribution of labels and relationship types are imbalanced, or from a symmetric point of view, the query frequencies for each relationship type and node label are uneven. We wish during the partitioning phase to consider these imbalances in order to produce a solution more effective to the user perspective or to the graph distribution. Our model can be easily extended to capture both cases.

Let $W_{rel}(n \times n)$ be a diagonal matrix where $x_{ij} = 0$ if $i \neq j$. This matrix in its main diagonal contains a positive weight indicating the importance of each relationship type. Similarly, let $W_{lab}((d+1) \times (d+1))$ be a diagonal matrix, which in its main diagonal contains the importance of each different label. The $d+1$ entry in the main diagonal of this matrix corresponds to the weight of the edge-cut between boundary nodes. In cases where the user wants to specify the importance of each relationship and label to the partitioning scheme, these two weight matrices can be included to the objective function as follows:

$$WeightedRL\text{-}cut(P_k) = \|vectorize(W_{rel} * P * W_{lab})\|_p \quad (6)$$

3. ADAPTING ONLINE ALGORITHMS FOR LABELED GRAPHS

The partitioning scoring function described in this work, can be used and adapted by any partitioning algorithm (offline or online) that wishes to produce partitions that will be used on queering property graphs. For the purposes of our work, we adapt the scoring functions of two state-of-the-art online partitioning algorithms LDG and FENNEL, respectively, that will be used throughout our experimental section.

[2]Function $vectorize(A)$ converts a $A_{n,m}$ matrix into a vector with $n \times m$ dimensions.

Online algorithms use a streaming scenario, where nodes are incoming, one at a time, each with its adjacency list and the decision of placing them in a partition is based sorely on the adjacent nodes that have already been placed in partitions (no global graph information is required) from previous streams. The decision for each algorithm is based on a scoring function that it uses, which we briefly describe here for completeness.

Linear Deterministic Greedy (LDG): The algorithm places a newly arrived node u to the partition V_i that maximizes:

$$|N(u) \cap V_i| \times (1 - \frac{|V_i|}{n/k})$$

where $N(u)$ denotes the number of neighbors of node u (incoming with each node in the stream).

FENNEL: It places a newly arrived node u to the partition V_i that maximizes:

$$|N(u) \cap V_i| - \alpha \frac{\gamma}{2} |V_i|^{\gamma - 1}$$

The first part of the scoring function for both algorithms evaluates the locality of each node to every formed partition and the second part enforce the balance restriction. Unlike LDG, where multiplicative weights enforce exact balance, FENNEL only ensures approximate balance.

In order to adjust the evaluation function for both algorithms to work with our multi-objective function, we have replaced the first part of their equation with our combined relation and label cut function. The balance restriction parts for both functions remain the same. The transformed relation and label aware partitioning functions for both algorithms are described next. We refer to the adapted, label/relation-aware versions of LDG and FENNEL as RL-LDG and RL-FENNEL, respectively.

RL-LDG: The algorithm places a newly arrived node u to the partition V_i that maximizes:

$$\|vectorize(P_{V_i})\|_p \times (1 - \frac{|V_i|}{n/k})$$

where $\|P_{V_i}\|$ denotes the norm of the combined relation and label matrix of each partition V_i.

RL-FENNEL: It places a newly arrived node u to the partition V_i that maximizes:

$$\|vectorize(P_{V_i})\|_p - \alpha \frac{\gamma}{2} |V_i|^{\gamma - 1}$$

Thus, for an incoming node in the stream, we calculate for every candidate partition a matrix that combines relation and label cuts. Then, we place the node to the partition that maximizes the norm of the matrix.

4. EXPERIMENTS

In this section we present experimental results on labeled graphs using our suggested metrics. We first describe our experimental setup in Section 4.1.Then, we present results using both synthetic and real datasets in Section 4.2.

4.1 Experimental Setup

Table 1 summarizes the basic statistics for each graph used in our experiments. The first eight graphs were chosen to balance both size and variety and are frequently used in evaluating graph partitioning algorithms [3, 15, 22]. We mention that this work doesn't aim to measure the effectiveness of the online algorithms against alternative offline techniques, but to demonstrate the gains of the transformed evaluation functions. For this reason, although most graphs used fit in memory and an offline algorithm could be

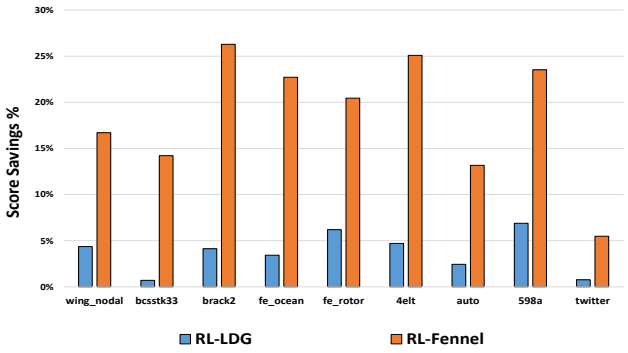

Figure 2: RL-cut savings

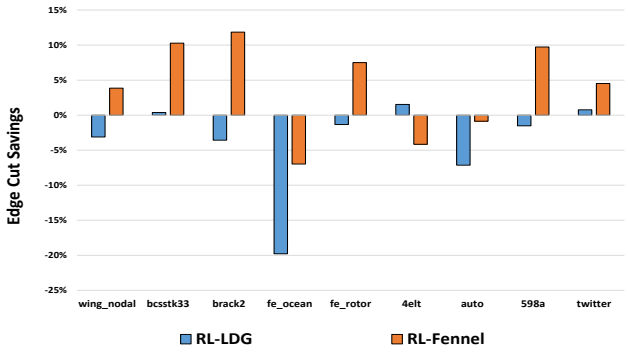

Figure 3: Edge-cut savings

used, the presented evaluation concentrates on the effects of using the new proposed metrics in the online scenario. Also since these graphs do not contain labels nor relationship types, we added both using a custom generator as described in the following sections.

Name	V	E	Avg. Degree
bcsstk33	8,738	291,583	33.4
wing_nodal	10,937	75,488	6.9
4elt	15,606	45,878	2.9
brack2	62,631	366,559	5.8
fe_rotor	99,617	662,431	6.6
598a	110,971	741,934	6.7
fe_ocean	143,437	409,593	2,8
auto	448,695	3,314,611	7.4
twitter	34,062,759	910,526,369	26.7

Table 1: Statistics of Evaluation Graphs

The data for the Twitter graph was collected by using the API provided by the social network. The nodes of this graph are users of Twitter. Each node includes (when available) the user nationality, gender and language. We have used 8 labels for the most common nationalities and languages included in the dataset and 2 labels for the gender. The edges of the graph depict the "follow" relationship and are directed. For our experiments we treated the edges of Twitter graph as undirected.

All experiments presented in this paper were performed in a single machine, with Intel Core CPU i7-4700MQ, and 8GB of main memory. We evaluate all algorithms by measuring the edge-cut, the relation-cut, the label-cut and, finally, the relation-label-cut, as defined in the previous sections. Due to lack of space we only present results for the L_1-norm. Unless otherwise specified, the number of partitions used was k=8. The objective function parameters for FENNEL for each graph, were selected in order to allow the balance restrictions to relax to an upper bound of 5%. All presented figures are averages computed from ten repetitions of each respective experiment.

4.2 Online Algorithms with Label and Relations Awareness

In this section of the experimental evaluation we measure the effectiveness of the proposed metrics in the case of the two online partitioning algorithms mentioned above. We notice that we do not provide partitioning times for both algorithms, since the transformed evaluation function doesn't affect the algorithms performance. With the exception of the labeled Twitter graph, in all other datasets we first generate a random number from 1-10 labels for each node and then distribute those labels uniformly across each node label vector. This way the resulting distribution of label counts is about the same for each label. We also used ten different relationship types and assigned them to edges of each graph using zipf distribution.

In Figure 2, we evaluate the relative reduction in the RL-cut produced by RL-LDG and RL-FENNEL compared to the standard LDG and FENNEL algorithms. We notice that in all graphs, the adapted algorithms generate partitions with significantly lower RL-cut values. By tracing the works of each algorithm we noticed that RL-LDG sometimes places an arriving node to a non-optimal partition, in order to keep them balanced. RL-FENNEL, which has a small tolerance for imbalance during partitioning, manages to achieve lower scores of up to 27%. The relative improvement in the Twitter graph is lower but this is explained by the fact that a large number of users (about 35% of them) did not provide appropriate labels in their profiles.

In Figure 3, we evaluate the relative reduction in the standard edge-cut metric typically used in graph partitioning algorithms. As expected, in some cases the edge-cut is increased by our adapted algorithms, since the node label distribution is also taken into account when making placement decisions. It is worth noting though that RL-FENNEL, in most cases manages to reduce the edge-cut (as is indicated by positive saving values in the figure), eventhough this is not its primary objective. Moreover, the relationship-cut is reduced in most graphs as is depicted in Figure 4. Last, in Figure 5, we measure the reduction in the label-cut obtained by the adapted algorithms. Compared to the original algorithms that are oblivious to the labels on the graph vertices, the adapted algorithms provide significant savings of up to 32%.

In Figure 6 we repeat the experiment for one of the graphs but this time we vary the number of requested partitions between 2 and 64. The savings on the RL-cut provided by RL-FENNEL seem to remain relatively stable, in the range of 20%-35%. RL-LDG provides savings but these seem to fluctuate with the number of partitions.

In Figure 7 we vary the number of maximum labels in the nodes from 1 up to 1000 and use a single relationship type. The graph indicates that both algorithms increase their savings in the RL-cut compared to their label-oblivious counterparts, as the number of node labels increases.

Finally, in Figure 8 we demonstrate the ability of our techniques to incorporate different weights on the labels and relationship-types. As an example, we used a weighting diagonal matrix W_{lab} with a different weight of importance for each of the label types. For this

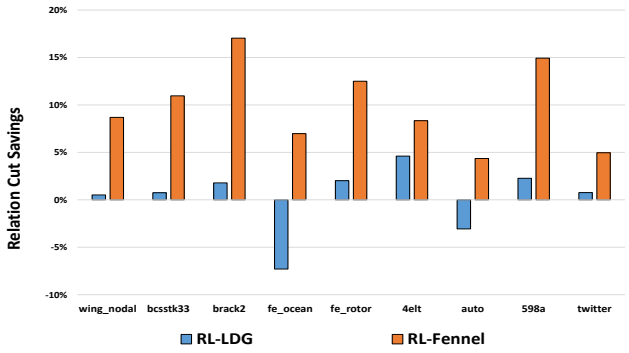

Figure 4: Relation-cut savings

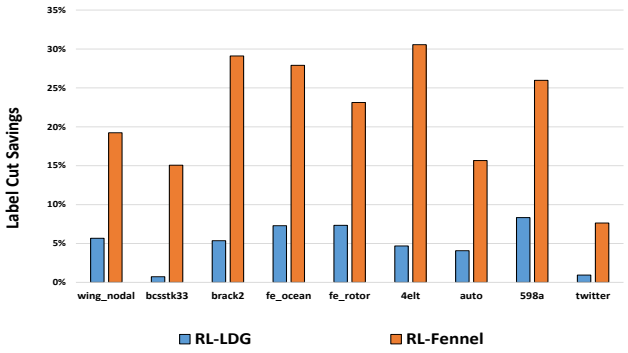

Figure 5: Label-cut savings

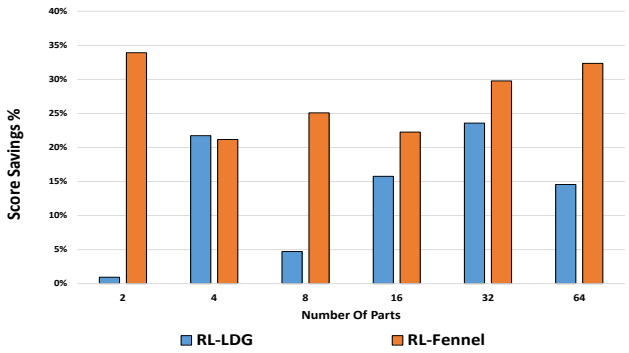

Figure 6: RL-cut savings for various values of k-parts for 4elt graph

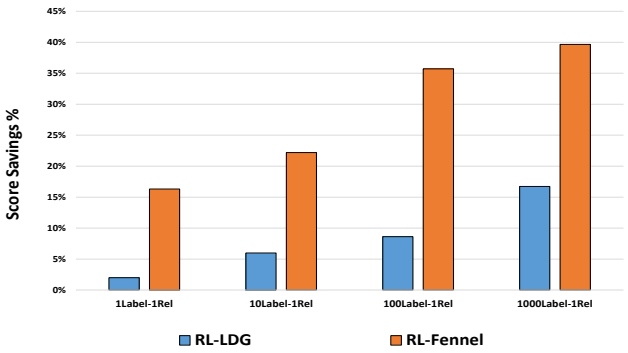

Figure 7: RL-cut savings, varying the number of of labels for brack2 graph

experiment we used 10 labels and 1 relation. The importance of each label was proportional to its position in the vector, thus label L^i had weight i, for i in $[1 \ldots 10]$. In the figure, we depict the label-cut ratio for each label on the unweighted and weighted version of RL-FENNEL. We notice that when no weights are used, RL-FENNEL cuts the same percentage of edges to all labels. When the weighting matrix is used, RL-FENNEL respects the given weights, resulting to lower cuts in labels of greater importance.

5. RELATED WORK

Graph partitioning is a well studied NP-hard problem [8]. The static balanced graph partitioning problem (offline) remains NP even in the case of k=2 (bisection) and also if one relaxes the balanced constraint [2]. There are many approximation algorithms for this problem. Andreev and Racke give an LP-based solution that obtains a $O(logn)$ approximation [2]. There are many offline heuristics of different nature such as spectral [7, 20], geometric [21], combinatorial [6, 14] and multilevel [10, 13] that also solve this problem but with no performance guarantees. In practice, all the above heuristics are quite effective, and many tools are available, like Chaco [10], METIS [13] and SCOTCH [19]. However, because of their static nature they are not best suited for big and evolving graphs.

On the other hand, one-pass (online) streaming algorithms recently introduced [18, 24, 25], consider the graph partitioning problem where graph nodes are loaded continuously in a stream, each with its adjacency list. Online algorithms mainly address the problem of partitioning graphs that can not physically fit in memory and, therefore, offline methods cannot be used. Node placement to partitions is performed on the fly, based on a heuristic that evaluates only the current node locality. The quality of the produced partitions mostly depends on the stream order of the incoming nodes but also on the objective function that each algorithm uses.

In both cases (online and offline), partitioning algorithms aim to minimize the edge-cut between partitions with respect to the balance restrictions. The task of minimizing the edge-cut can be considered as the objective and the requirement that the partitions will be of the same size can be considered as the constraint. To the best of our knowledge this work first address the problem of a combined edge and label cut function for cases of multi-labeled graphs. Other works also consider the problem of multi-constraint [4, 12] and multi-objective [9] partitioning and compute partitions that simultaneously balance multiple weights associated with the vertices while minimizing multiple objectives associated with the edges. These works can address the problem of different type of edges using vectors but consider the nodes of the graph by a single weight indicating, depending on the application, the importance of each node to the balance of the partition. Also other works on attributed graphs exist [1, 5, 26] for graph clustering, that mostly use similarity functions in order to place nodes to partitions.

Our work is the first to consider the case of multi-labeled node and edges in a graph. The proposed partitioning metric of this work can be easily used as a replacement of the scoring function of any partitioning algorithm (offline and online), in order to balance the cut edges between partitions considering various types of relations and at the same time respect the multiple characteristics of each node, described by their labels.

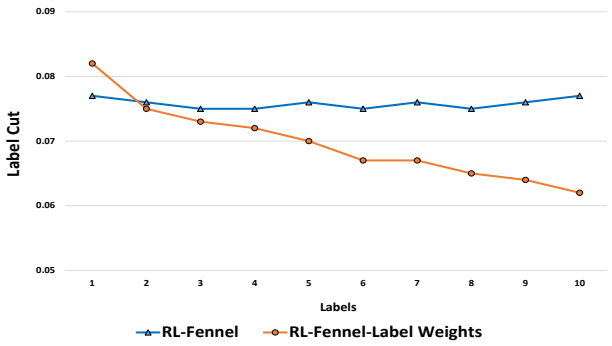

Figure 8: Label-cut, when different weights are associated with each label (fe_ocean graph)

6. CONCLUSIONS

In this paper we discussed new metrics that can be used for partitioning large graphs with multiple labels on their nodes and edges. Our techniques extend the commonly used edge-cut metric for multiple edge types and then combine it with a new label-cut metric that takes into account existing labels on the nodes. We have described the adaptation of two online algorithms for the new metrics and have presented experiments using real and synthetic graphs of realistic sizes. Our results demonstrate that the proposed metrics provide significant savings in the constituent relation and labeled cuts and can be adapted to accommodate different weights on them, depending on the application needs.

7. REFERENCES

[1] C. C. Aggarwal and H. Wang. A survey of clustering algorithms for graph data. In *Managing and Mining Graph Data*, volume 40 of *Advances in Database Systems*, pages 275–301. Springer, 2010.

[2] K. Andreev and H. Räcke. Balanced Graph Partitioning. In *Proc. of SPAA*, pages 120–124, New York, NY, USA, 2004.

[3] D. A. Bader, H. Meyerhenke, P. Sanders, and D. Wagner, editors. *Graph Partitioning and Graph Clustering - 10th DIMACS Implementation Challenge Workshop, Georgia Institute of Technology, Atlanta, GA, USA*, 2013.

[4] U. Catalyurek and C. Aykanat. A hypergraph-partitioning approach for coarse-grain decomposition. In *Proceedings of the 2001 ACM/IEEE Conference on Supercomputing*, SC '01, pages 28–28, New York, NY, USA, 2001.

[5] H. Cheng, Y. Zhou, and J. X. Yu. Clustering large attributed graphs: A balance between structural and attribute similarities. *ACM Trans. Knowl. Discov. Data*, 5(2):12:1–12:33, Feb. 2011.

[6] C. M. Fiduccia and R. M. Mattheyses. A linear-time heuristic for improving network partitions. In *Proceedings of DAC*, pages 175–181, Piscataway, NJ, USA, 1982.

[7] M. Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Mathematical Journal*, 25(4):619–633, 1975.

[8] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified np-complete problems. In *Proceedings of STOC*, pages 47–63, New York, NY, USA, 1974.

[9] K. S. George, G. Karypis, and V. Kumar. A new algorithm for multi-objective graph partitioning. In *In Proceedings of Europar*, pages 322–331. Springer Verlag, 1999.

[10] B. Hendrickson and R. Leland. A multilevel algorithm for partitioning graphs. In *Proceedings of Supercomputing*, New York, NY, USA, 1995.

[11] U. Kang, C. E. Tsourakakis, and C. Faloutsos. Pegasus: A Peta-Scale Graph Mining System Implementation and Observations. In *Proceedings of ICDM*, pages 229–238, Washington, DC, USA, 2009.

[12] G. Karypis and V. Kumar. Multilevel algorithms for multi-constraint graph partitioning. In *Proceedings of the 1998 ACM/IEEE Conference on Supercomputing*, SC '98, pages 1–13, Washington, DC, USA, 1998.

[13] G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. *J. Parallel Distrib. Comput.*, 48(1):96–129, Jan. 1998.

[14] B. Kernighan and S. Lin. An Efficient Heuristic Procedure for Partitioning Graphs. *The Bell Systems Technical Journal*, 49(2), 1970.

[15] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection, June 2014.

[16] Y. Low, J. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin, and J. M. Hellerstein. Graphlab: A New Parallel Framework for Machine Learning. In *Proceedings of UAI*, July 2010.

[17] 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 SIGMOD*, pages 135–146, New York, NY, USA, 2010.

[18] J. Nishimura and J. Ugander. Restreaming graph partitioning: Simple versatile algorithms for advanced balancing. In *Proc. of SIGKDD*, pages 1106–1114, New York, NY, USA, 2013.

[19] F. Pellegrini and J. Roman. Scotch: A software package for static mapping by dual recursive bipartitioning of process and architecture graphs. In *Proceedings of HPCN*, pages 493–498, London, UK, 1996. Springer-Verlag.

[20] A. Pothen, H. D. Simon, and K.-P. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM J. Matrix Anal. Appl.*, 11(3):430–452, May 1990.

[21] H. D. Simon. Partitioning of unstructured problems for parallel processing, 1991.

[22] A. J. Soper, C. Walshaw, and M. Cross. A Combined Evolutionary Search and Multilevel Optimisation Approach to Graph-Partitioning. *Journal of Global Optimization*, 29(2):225–241, June 2004.

[23] V. Spyropoulos and Y. Kotidis. Dynamic partitioning of big hierarchical graphs. In *Proceedings of the First International Workshop on Big Dynamic Distributed Data, Riva del Garda, Italy, August 30, 2013*, pages 37–42, 2013.

[24] I. Stanton and G. Kliot. Streaming graph partitioning for large distributed graphs. In *Proceedings of SIGKDD*, pages 1222–1230, New York, NY, USA, 2012.

[25] C. Tsourakakis, C. Gkantsidis, B. Radunovic, and M. Vojnovic. Fennel: Streaming graph partitioning for massive scale graphs. In *Proceedings of WSDM*, pages 333–342, New York, NY, USA, 2014.

[26] Z. Xu, Y. Ke, Y. Wang, H. Cheng, and J. Cheng. A model-based approach to attributed graph clustering. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, pages 505–516, New York, NY, USA, 2012.