PRIORITIZED RESTREAMING ALGORITHMS FOR BALANCED GRAPH PARTITIONING

Co-Authors: Amel Awadelkarim and Johan Ugander

SIAM Workshop on Network Science 2020
July 9–10 · Toronto

Summary

We introduce a class of scalable, iterative, graph partitioning algorithms, called *prioritized restreaming algorithms*, and show that restreamed Linear Deterministic Greedy [3] with a custom prioritized stream ordering, *ambivalence* order, results in higher quality partitions (in terms of edge cut) than other recent scalable algorithms [5, 3, 2, 1], including non-streaming algorithms. Motivation for this contribution comes from an extensive comparison of the class of restreaming algorithms [3] and a class of iterative techniques based on label propagation [5, 2].

Background

Balanced graph partitioning is a critical step for many large-scale distributed computations with relational data. As graph datasets have grown in size and density, a range of highly-scalable balanced partitioning algorithms have appeared to meet varied demands across different domains. These algorithms seek to find an approximate solution to the following optimization problem: given an undirected graph $G = (V, E)$ on $|V| = n$ nodes and $|E| = m$ edges, an integer k, and an imbalance parameter ϵ, find a *partitioning* $P = \{V_1, ..., V_k\}$ of the node set into k disjoint *shards* V_i such that the number of cross-shard edges is minimized, and $\lceil(1-\epsilon)\frac{n}{k}\rceil \le |V_i| \le \lceil(1+\epsilon)\frac{n}{k}\rceil$ for all i.

As the starting point for our work, we observe that two recently introduced families of iterative partitioners—those based on restreaming and those based on balanced label propagation—can be viewed through a common modular framework of design decisions. We build this framework around three recent algorithms regarded as at or near the state-of-the-art: Balanced Label Propagation (BLP) [5], Restreamed Linear Deterministic Greedy (reLDG) [3], and Facebook's Social Hash partitioner (SHP) [2].

BLP and SHP belong to the family of approximation algorithms based on label propagation, beginning from an initial assignment, iteratively conducting node relocations to achieve higher quality partitions. Specifically, nodes are relocated to maximize the *gain* of relocation: for a node u, its gain g_u is defined as the maximum improvement in co-located neighbor count between an external and its current shard:

$$g_u = \max_{i\in[k]} |N(u) \cap V_i| - |N(u) \cap V_{P(u)}|, \qquad (1)$$

where $N(u)$ is u's neighbor set, and $P(u)$ denotes u's shard assignment under partition P. To handle the balance constraints, BLP solves a linear program at each iteration to determine the optimal number of nodes to move from shard i to j, for all $i, j \in [k]$, whereas SHP simply makes maximal balanced pairwise swaps between all shard pairs.

For SHP, we study three variations of the original algorithm to understand the roles of its underlying modules. One, we denote by KL-SHP, swaps nodes across shard pairs in decreasing order by gain, allowing for nodes with negative gain to be swapped if the net gain is positive. A simplification, SHP-II, also swaps nodes in order of decreasing gain, but does not allow nodes with negative gain to be swapped. A final simplification, SHP-I, swaps only nodes with positive gain, and in a random order.

ReLDG is an example of what are called restreaming algorithms, processing the node set serially in repeated streams, with each node placed according to an assignment rule designed to achieve balance. Streaming algorithms are commonly motivated by a highly restricted computational framework where one is attempting to make node assignments while the graph is in transit, being moved and/or loaded (during ETL, in the language of data warehousing). As such, the only *stream orderings* of the node set tested prior to this work are random, breadth-first-search (BFS), and depth-first-search (DFS) to mimic the order obtained by a web-crawler or equivalent process [4]. By considering strategic stream orderings (the order in which the node set is considered by the algorithm) that import notions of priority from SHP/BLP, we contribute a new class of algorithms that we call *prioritized restreaming algorithms*.

Modular design decisions

Restreaming and label propagation-based algorithms can be viewed through a unified lens in terms of how they handle a small number of design decisions. We identify four

Table 1: Internal edge fraction, 16 shards, 10 iterations, exact balance ($\epsilon = 0$). Highest quality partition in **bold**.

	Synchronous				Streaming (reLDG)					
Graph	SHP-I	SHP-II	KL-SHP	BLP	Random	CC	BFS	Degree	Ambivalence	Gain
`pokec`	0.578	0.595	0.585	0.532	0.675	0.681	0.698	**0.716**	0.712	0.618
`livejournal`	0.626	0.648	0.625	0.617	0.674	0.666	0.731	0.745	**0.749**	0.671
`orkut`	0.535	0.555	0.534	0.531	0.650	0.628	0.665	**0.689**	0.679	0.626
`notredame`	0.783	0.635	0.652	0.612	0.882	0.864	**0.929**	0.902	0.924	0.878
`stanford`	0.737	0.711	0.697	0.629	0.856	0.844	0.891	0.900	**0.916**	0.793
`google`	0.670	0.603	0.616	0.606	0.848	0.814	0.868	0.959	**0.964**	0.799

such key decisions: synchronous vs. streaming assignment, flow-based vs. pairwise constraint handling, incumbency preference or not, and how node priority is implemented. BLP is synchronous, flow-based, prefers incumbent nodes, and prioritized. KL-SHP is synchronous, pairwise, without incumbency, and prioritized. Vanilla ReLDG is streaming, without incumbency, and without priority.

Priority in restreaming

From this framework, we develop *prioritized* restreaming algorithms, motivated by the two synchronous algorithms. We introduce several new stream orderings for consideration, both *static* and *dynamic* in nature. The orderings investigated in prior work are "static", as they are defined based on graph properties alone and are not updated between iterations. We add two additional static orderings for consideration: degree and local clustering coefficient, prioritized by streaming nodes in decreasing order.

Decreasing gain is the first prioritized dynamic order to consider given its role in the synchronous algorithms. However, nodes with a gain of 0 would end up tied for last in the stream, likely be evicted from their previous assignment. Moreover, nodes with a gain of 0 may actually incur significant *loss* in being relocated. To mitigate this, we introduce an analogous prioritized dynamic stream order, *ambivalence order*, defined as

$$a_u = - \max_{i \in [k] \setminus P(u)} \left| |N(u) \cap V_i| - |N(u) \cap V_{P(u)}| \right| .$$

The higher (less negative) the ambivalence score, a_u, the smaller the gap in neighbor co-location count between the node's current assignment and the best external shard, and the more "ambivalent" node u is to reassignment.

Results

We report the partition qualities of all methods—BLP, KL-SHP, its restricted forms (SHP-I, SHP-II), and reLDG with six stream orders (random, BFS, local clustering coefficient, degree, gain, ambivalence)—on several networks in Table 1. We see that reLDG with a random stream order outperforms the synchronous methods in all networks by a sizable margin, a surprising result considering that reLDG is generally regarded as further constrained by its online design. Furthermore, we see considerable improvement in quality from utilizing prioritized stream orders. Of these, ambivalence order results in the highest (or nearly highest) quality partition on the large-scale graphs we test. From studying the rank correlation of each pair of orders, we find that degree and ambivalence orders are highly correlated at every iteration, suggesting degree as the preferred static ordering.

We also observe that while BLP boasts the most bells and whistles of the synchronous algorithms—sorting move queues by gain, and satisfying flow-based balance using an LP—it generally performs the worst of the synchronous methods. Furthermore, the restricted forms of SHP, SHP-II and SHP-I, very often do better than KL-SHP.

References

[1] K. Aydin, M. Bateni, and V. Mirrokni. Distributed balanced partitioning via linear embedding. *Algorithms*, 12(8):162, 2019.

[2] I. Kabiljo, B. Karrer, M. Pundir, S. Pupyrev, and A. Shalita. Social hash partitioner: a scalable distributed hypergraph partitioner. *VLDB*, 10(11):1418–1429, 2017.

[3] J. Nishimura and J. Ugander. Restreaming graph partitioning: Simple versatile algorithms for advanced balancing. In *KDD*, pages 1106–1114, 2013.

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

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