Shared-Memory Parallel Dynamic Louvain Algorithm for Community Detection

Subhajit Sahu*, Kishore Kothapalli*
International Institute of Information Technology Hyderabad, India
Gachibowli, Hyderabad, India, 500032.
{subhajit.sahu@research., kkishore@}iiit.ac.in

Dip Sankar Banerjee**
Indian Institute of Technology Jodhpur, India
Surpura Bypass Rd, Karwar, Rajasthan, India, 342030.
dipsankarb@iitj.ac.in

***Abstract*—Community detection refers to the identification of coherent partitions in networks. In this poster, we present a parallel dynamic Louvain algorithm that finds communities in rapidly evolving graphs. Given a batch update of edge deletions or insertions, our algorithm identifies an approximate set of affected vertices in the graph with minimal overhead and updates the community membership of each vertex. This process repeats until convergence. Our approach achieves a mean speedup of $7.3\times$, compared to our parallel and optimized implementation of Δ-screening combined with Louvain, a recently proposed state-of-the-art approach.**

I. Introduction

Communities are collections of vertices that more strongly connected to vertices within a collection, than to those outside. Mdoreover, many real-world graphs evolve with the insertion/deletion of edges/vertices. For efficiency reasons, one needs algorithms that update the results without re-computing from scratch — known as *dynamic algorithms*.

Current literature on dynamic community detection algorithms leaves open a few questions. Some algorithms [1], [2], do not match the performance of static algorithms even for modest-sized batch updates. A few works such as that of Riedy et al. [3] and Zarayaneh et al. [4], do not consider cascading changes and flag too many vertices as affected.

In this paper, we introduce our Parallel Dynamic Louvain algorithm. Given a batch update of edge deletions or insertions, our algorithm incrementally identifies an approximate set of affected vertices in the graph with minimal overhead. The algorithm updates the community membership of each vertex based on the Louvain algorithm, a high-quality static community detection algorithm. Our algorithm achieves a mean speedup of $7.3\times$ compared to a parallel and optimized implementation of Δ-screening combined with Louvain, a recently proposed state-of-the-art approach.

II. Preliminaries

Let $G(V, E, w)$ be an undirected, weighted graph where V denotes the set of vertices with $N = |V|$, E denotes the set of edges with $M = |E|$, and $w_{ij} = w_{ji}$ is a positive weight associated with each edge in the graph with $m := \sum_{i,j \in V} w_{ij}/2$. If the graph is unweighted, we assume each edge to be associated with unit weight ($w_{ij} = 1$). We denote the neighbors of a vertex i as $J_i = \{j \mid (i,j) \in E\}$, and the weighted degree of each vertex i as $K_i = \sum_{j \in J_i} w_{ij}$.

Disjoint community detection is the process of arriving at a community membership mapping, $C : V \rightarrow \Gamma$, which maps each vertex $i \in V$ to a community-id $c \in \Gamma$, where Γ is the set of community-ids. We denote the vertices of a community $c \in \Gamma$ as V_c. We denote the community that a vertex i belongs to as C_i. Further, we denote the neighbors of vertex i belonging to a community c as $J_{i \rightarrow c} = \{j \mid j \in J_i \ and \ C_j = c\}$, the sum of those edge weights as $K_{i \rightarrow c} = \{w_{ij} \mid j \in J_{i \rightarrow c}\}$, the sum of edge weights within a community c as $\sigma_c = \sum_{(i,j) \in E \ and \ C_i = C_j = c} w_{ij}$, and the total edge weight of c as $\Sigma_c = \sum_{(i,j) \in E}$ and $_{C_i = c} w_{ij}$ [4].

We use *modularity* Q to evaluate the quality of communities obtained as $Q = \sum_{c \in \Gamma} \left[\frac{\sigma_c}{2m} - \left(\frac{\Sigma_c}{2m} \right)^2 \right]$. The *delta modularity* of moving a vertex i from the community d to the community c, denoted as $\Delta Q_{i:d \rightarrow c}$, can be calculated as $\Delta Q_{i:d \rightarrow c} = \frac{1}{m}(K_{i \rightarrow c} - K_{i \rightarrow d}) - \frac{K_i}{2m^2}(K_i + \Sigma_c - \Sigma_d)$.

a) Louvain method:: Louvain method [5] is a greedy agglomerative algorithm that finds high-modularity communities in a graph with a time complexity of $O(LM)$ and a space complexity of $O(N+M)$, where L is the total number of passes performed. The algorithm runs in two phases: the *local-moving phase* and the *aggregration phase*. In the local moving phase, each vertex i greedily decides to move to the community of one of its neighbors $j \in J_i$ that gives the highest increase in modularity $\Delta Q_{i:C_i \rightarrow C_j}$. The *aggregation phase* creates a single super-vertex corresponding to all the vertices in a community. These two phases make up one pass and the passes repeat until there is no further increase in modularity beyond a set threshold.

b) Dynamic Graphs:: A dynamic graph is a sequence of graphs, where $G^t(V^t, E^t, w^t)$ denotes the graph at time step t with $t \geq 0$. The graph G^0 is the *base graph*. We consider only changes to the edges of the graph over time. We use Δ^t to denote the changes to the edges of the graphs $G^{t-1}(V^{t-1}, E^{t-1}, w^{t-1})$ and $G^t(V^t, E^t, w^t)$ at consecutive

*Partially supported by a grant from the Department of Science and Technology (DST), India, under the National Supercomputing Mission (NSM) R&D in Exascale initiative vide Ref. No: DST/NSM/R&D_Exascale/2021/16.

**Partially supported by a grant from the Ayursasthya Yojana scheme from Ministry of AYUSH vide Ref No. S/12011/12/2021-SCHEME

time steps $t-1$ and t. The set Δ^t consists of a set of edge deletions $\Delta^{t-} = E^{t-1} \setminus E^t$ and a set of edge insertions $\Delta^{t+} = E^t \setminus E^{t-1}$.

III. Approach

We now describe our Parallel Dynamic Louvain algorithm. We take as input the previous snapshot of the graph G^{t-1}, the batch update consisting of edge deletions Δ^{t-} and insertions Δ^{t+}, the previous community membership of each vertex C^{t-1}, the previous weighted degree of each vertex K^{t-1}, and the previous edge weight of each community Σ^{t-1}.

First, we mark the initial set of vertices as affected. We initialize the community membership C of each vertex, obtain the updated vertex weights K and community weights Σ, and get the graph G', community membership C', vertex weights K', and community weights Σ' at the current pass.

For each pass, we perform the *local-moving* phase of *Louvain*. In each iteration, each affected vertex i in the graph G' identifies the the best community c^* as well as the associated delta-modularity (highest) δQ^*. We use per-thread collision-free hashtables to performm this step. If, for vertex i, the best community c^* is different from the original community membership $C'[i]$ of vertex i, we update the community membership of the vertex C' and atomically update the total edge weights linked to each community Σ'. If c^* is no longer the best choice, vertex i will be re-processed in the next iteration. Otherwise, we mark i as not affected for the next iteration. At the end of each iteration, if the total delta-modularity across all vertices ΔQ is less than the specified tolerance τ, we terminate the local-moving phase and return the number of iterations performed. The algorithm stops when either the community labels converge or when the local-moving phase executes for $MAX\_ITERATIONS$.

Next, we renumber community IDs to generate the aggregated graph G'. If the community labels converge after one pass, we terminate the algorithm. We check if only a small fraction of communities merged. We calculate the ratio of the current to the original number of communities, $|\Gamma|/|\Gamma_{old}|$. If this ratio is below an a chosen aggregation tolerance, we stop to avoid the computationally expensive *aggregation* phase.

Next, we aggregate and store the result as graph G'. After aggregation, super-vertex weights K' are obtained. Each vertex in the aggregated graph forms its own singleton community, leading to identical super-community weights Σ' as K'. Subsequently, all super-vertices are marked as affected, and their community memberships are initialized. The threshold scaling optimization is applied, reducing tolerance τ using a factor `TOLERANCE_DROP`. This reduction decreases local-moving phase iterations, enhancing performance with minimal impact on community modularity. Following the completion of all necessary passes, dendrogram flattening is executed. Finally, the auxiliary information of the updated graph includes the community membership of each vertex C, as well as the weighted degree of each vertex K and community Σ.

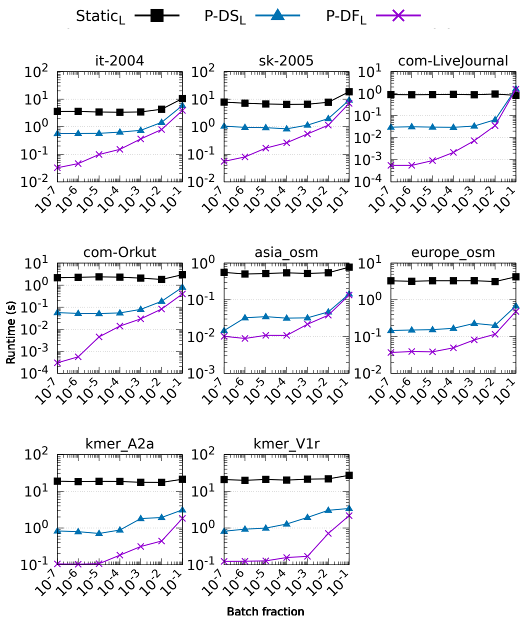

Fig. 1: Time taken with Parallel *Static*, Δ-*screening*, and our *Dynamic Louvain* on batch updates of $10^{-7}|E|$ to $0.1|E|$.

TABLE I: List of 8 graphs obtained from the *SuiteSparse Matrix Collection* (directed graphs are marked with $*$).

Graph	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert\Gamma\rvert$
it-2004*	41.3M	2.19B	5.28K
sk-2005*	50.6M	3.80B	3.47K
com-LiveJournal	4.00M	69.4M	2.54K
com-Orkut	3.07M	234M	29
asia_osm	12.0M	25.4M	2.38K
europe_osm	50.9M	108M	3.05K
kmer_A2a	171M	361M	21.2K
kmer_V1r	214M	465M	6.17K

IV. Evaluation

We use an AMD EPYC-7742 server, with graphs shown in Table I. For each graph, we generate an undirected random batch update consisting of edge deletions or insertions each with an edge weight of 1. Fig. 1 shows the results.

References

[1] M. Cordeiro, R. Sarmento, and J. Gama, "Dynamic community detection in evolving networks using locality modularity optimization," *Social Network Analysis and Mining*, vol. 6, no. 1, pp. 1–20, 2016.

[2] X. Meng, Y. Tong, X. Liu, S. Zhao, X. Yang, and S. Tan, "A novel dynamic community detection algorithm based on modularity optimization," in *Proc. Intl. Conf. Software Engg. and Service science*, 2016, pp. 72–75.

[3] J. Riedy and D. A. Bader, "Multithreaded community monitoring for massive streaming graph data," in *IEEE IPDPSW*, 2013, pp. 1646–1655.

[4] N. Zarayeneh and A. Kalyanaraman, "Delta-Screening: A Fast and Efficient Technique to Update Communities in Dynamic Graphs," *IEEE TNSE*, vol. 8, no. 2, pp. 1614–1629, 2021.

[5] V. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. Stat. Mech.*, vol. 2008, no. 10, pp. P10 008:1–12, 2008.