Parallel Counting of Triangles in Large Graphs: Pruning and Hierarchical Clustering Algorithms

Chun-Yen Kuo*, Ching Nam Hang*, Pei-Duo Yu* and Chee Wei Tan†,*
*City University of Hong Kong, †Tencent AI Lab
{chunykuo, cnhang3-c, peiduoyu2-c, cheewtan}@cityu.edu.hk

***Abstract*—As part of the 2018 MIT-Amazon Graph Challenge on subgraph isomorphism, we propose a novel joint hierarchical clustering and parallel counting technique called the PHC algorithm that can compute the exact number of triangles in large graphs. The PHC algorithm consists of first pruning followed by hierarchical clustering based on geodesic distance and then triangle counting in parallel. This allows scalable software framework such as MapReduce/Hadoop to count triangles inside each cluster as well as those straddling between clusters in parallel. We characterize the performance of the PHC algorithm mathematically, and its performance evaluation using representative graphs including random graphs demonstrates its computational efficiency over other existing techniques.**

[1]

I. Introduction

The subgraph isomorphism problem is a longstanding and challenging task in computer science. This paper studies its special case of pattern matching when the subgraph is particularly a triangle as part of the 2018 MIT-Amazon Graph Challenge [1]. We focus on scalable algorithm design for parallel computation to handle large graphs. The problem statement is given as follows.

Definition 1 (Triangle Counting)**.** Given a simple graph $G = (V(G), E(G))$, if there exists three vertices v_i, v_j and v_k such that (v_i, v_j), (v_i, v_k) and (v_j, v_k) are all in $E(G)$ then we say v_i, v_j, v_k form a triangle. The goal is to compute exactly the size of the set in G characterized by

$$|\{(v_i, v_j, v_k)|v_i, v_j, v_k \text{ form a triangle where } i < j < k\}| \tag{1}$$

Beyond the computational challenge, there are useful applications of triangle counting techniques found in graph analysis applications such as web spamming detection [2], community detection, web recommendation. Furthermore, since triangles are the most basic structural units in a graph, triangle counting can be adapted to address other graph-theoretic problems such as the maximum clique problem or the k-truss computation problem [3] that are practically relevant to biological network analysis [4].

[1]This work was supported in part by the Hong Kong Innovation and Technology Fund Project (ITS/180/16), in part by the Science, Technology, and Innovation Commission of Shenzhen Municipality under Project JCYJ20170307090810981, in part by the National Natural Science Foundation of China under Grant 61771017, and in part by the 2018 Tencent AI Lab Visiting Scholars Program.

We briefly discuss some of the recent work on triangle counting. For a given graph G with N vertices, a naive method that exhaustively checks every group of three vertices has a time complexity $O(N^3)$ since there are $\binom{N}{3}$ groups altogether. This is however impractical for large graphs that often have more than hundreds of millions of vertices. Even if there is an efficient method to list down all triangles, this is still time consuming when the graph is dense, because there are overwhelmingly many triangles needed for storage. There are work that use graph embedding with $O(N)$ triangles [5] and also approximately counting methods [6]. On the other hand, to address scalability, some works proposed algorithms to compute an exact or approximate solution [7], [8]. Other work study advances in linear algebraic computational methods that use the adjacency matrix A of the given graph, whereupon the number of triangles in a graph can be obtained from the diagonal of A^3. TABLE I summarizes the key mathematical notations used in this paper.

TABLE I
Key mathematical notations used in this paper.

Notation	Description
G	The original simple graph.
G'	The graph obtained after pruning G.
G_i	The ith connected component of G'.
$V(G)$	The set of vertices of G.
$E(G)$	The set of edges of G.
N	The number of vertices in G, which equals to $\lvert V(G)\rvert$.
N'	The number of vertices in G', which equals to $\lvert V(G')\rvert$.
M	The threshold used in the pruning step.
$\mathsf{deg}(v)$	The degree (the number of neighbors) of v.
$\mathsf{dist}(u, v)$	The distance between vertices u and v.
$N_G(v)$	The vertex set which contains all neighbors of v.
$S_{i,j}$	The jth cluster of G_i.
$G_{i,j}$	The induced subgraph of G with $V(G_{i,j}) = S_{i,j}$.
$\mathsf{PHC}(G)$	The number of triangles in G counted by the PHC algorithm.

In this paper, we present a novel (PHC) algorithm consisting of pruning, hierarchical clustering and parallel counting for large graphs. The main contributions of this paper are as follows:

- The pruning step in our PHC algorithm depends on a suitable threshold tuning to decompose a large graph effectively into smaller components to accelerate counting.
- The hierarchical clustering step builds a geodesic distance-based hierarchy of clusters by leveraging Tar-

978-1-5386-5989-2/18/$31.00 ©2018 IEEE

jan's breadth-first search (BFS) graph decomposition technique that yields highly-parallelizable structure for parallel implementation in MapReduce software framework.
- We perform extensive performance evaluations using graph dataset from the public domain (e.g., Stanford SNAP website) and also synthetic random graphs generated from Barabási-Albert model. We then establish the optimal tuning in our PHC algorithm to demonstrate its improvement over the state of the art.

This paper is organized as follows. In Section II, we present the algorithmic approach to count triangles in a given graph, and we use an illustrative example (Fig. 1) to illustrate how the PHC algorithm works and to analyze its computational complexity. Performance evaluation results and MapReduce software implementation can be found in Section III, and the performance for random graphs are found in Section IV. We conclude the paper in Section V.

II. Algorithms and Analysis

Let $G = (V(G), E(G))$ be a simple graph without loops or multiple edges. The PHC algorithm has three main steps:

> ***Step 1.*** PRUNING
> Decompose the given graph G into connected components. In this step, we set a parameter M, and then remove vertices from this graph until
> $$\forall\ u \in V(G'), M < \mathsf{deg}(u) < N' - 1. \qquad (2)$$

> ***Step 2.*** HIERARCHICAL CLUSTERING
> Cluster vertices in a connected component in a hierarchical manner using the breadth-first search algorithm (BFS) and according to a choice of the root of the BFS tree ([9], [10]).

> ***Step 3.*** COUNTING
> Count the triangles within the same cluster and the triangles straddling across different clusters in each connected component.

A. Pruning Step

In the first step, we remove those vertices with degree less than or equal to M as well as the ones with degree $N' - 1$. When a vertex u with degree less than or equal to M is removed from G, then for each pair $(a, b) \in N(u)^2$, check if $(a, b) \in E(G)$. Once $(a, b) \in E(G)$, a triangle $\{u, a, b\}$ is counted. Hence, there are $\binom{\mathsf{deg}(u)}{2}$ pairs need to be checked for each removed vertex u. As a vertex connecting to all other vertices is removed, there are $|E(G)| - (|V(G)| - 1)$ triangles counted. Note that this pruning step is continued until the equation (2) is satisfied.

Data: Graph $G = (V(G), E(G))$
Result: (G', t), where G' is the resultant graph obtained by pruning G, and t is the number of triangles counted during pruning

```
t = 0;
N = |V(G)|;
m = |E(G)|;
%Pruning
while exists v with deg(v) ≤ M or = N − 1 do
    if deg(v) == N − 1 then
        t ← t + m − (N − 1);
        m ← m − (N − 1);
    end
    else if deg(v) ≤ M then
        for a < b & a, b ∈ N(v) do
            if (a, b) ∈ E(G) then
                t ← t + 1;
            end
            m ← m − deg(v);
        end
    end
    remove vertex v and its adjacent edges;
    N = N − 1;
end
return (G, t)
```

Step 1: Pruning step with input G.

Let $G' = (V(G'), E(G'))$ be the graph after pruning with N' vertices. Note that G' may not be a connected graph even if G were. Moreover, for all $v \in V(G')$, we have $M < \mathsf{deg}(v) < N' - 1$. If $N' = 0$ then the PHC algorithm ends here. For general graphs, N' is often greater than zero after the pruning step, hence we go to the next step.

B. Hierarchical Clustering Step

The second step is a hierarchical clustering algorithm ([9], [10]) based on the BFS tree traversal. Note that G' may not be a connected graph. Let G_i denote the ith connected component of G'. Then, we have $G' = \bigcup_{i=1}^{t} G_i$, and $t = 1$ when G' is a connected graph. Let v^i be the vertex with the maximum degree (i.e., degree center) in the connected component G_i for $i = 1, \ldots, t$. For each G_i, vertices are in the same cluster if their distance to v^i are equivalent. Hence, each cluster in G_i is defined as $S_{i,j} = \{v \in V(G') | \mathsf{dist}(v, v^i) = j\}$. Note that $S_{i,0} = \{v^i\}$ and $S_{i,1} = N_{G'}(v^i)$. Since all clusters in G_i can be constructed by the BFS tree traversal starting from the root v^i, we call the cluster $S_{i,j}$ the jth cluster of G_i.

Lemma II.1. For each triangle $\{a, b, c\}$ in G', either three vertices $\{a, b, c\}$ are in the same cluster say $S_{i,j}$, or two of them are in $S_{i,j}$ and the remaining one is in a neighboring cluster $S_{i,j-1}$ or $S_{i,j+1}$.

Proof. First we can observe that for each edge $(u, v) \in E(G')$ where $u \in S_{i_1,j_1}$ and $v \in S_{i_2,j_2}$, i_1 must be equal to i_2 as well

```
Data: Graph G'
Result: vertex sets S_{i,j}
S = V(G);
i = 1;
while S ≠ ∅ do
    Let v^i be the vertex with the maximum degree in S.
    S_{i,0} = {v^i};
    S ← S − S_{i,0};
    j = 1;
    while {u|dist(u, v^i) = j} ≠ ∅ do
        S_{i,j} = {u|dist(u, v^i) = j};
        S ← S − S_{i,j};
        j ← j + 1;
    end
    i ← i + 1;
end
return vertex sets S_{i,j}
```

Step 2: Hierarchical clustering ([9], [10]) step with input G'

as $|j_1 - j_2| \leq 1$ according to the definition of $S_{i,j}$. Hence, for a given triangle $\{a, b, c\}$ if we assume that $a \in S_{i_1,j_1}$, $b \in S_{i_2,j_2}$ and $c \in S_{i_3,j_3}$, then we have $i_1 = i_2 = i_3$, $|j_1 - j_2| \leq 1$, $|j_2 - j_3| \leq 1$ and $|j_1 - j_3| \leq 1$ since (a, b), (b, c) and (a, c) are edges. If $|j_1 - j_2| = |j_2 - j_3| = |j_1 - j_3| = 0$, then all of them are in the same cluster $S_{i,j}$. Otherwise, two of their second indices are equivalent to "j" and the other one is either $j - 1$ or $j + 1$, which implies that $\{a, b, c\}$ are in two neighboring clusters. □

This proof of Lemma II.1 is based on the fact that for any two clusters S_{i_1,j_1} and S_{i_2,j_2}, if $i_1 \neq i_2$ or $|j_1 - j_2| \geq 2$ then there is no edge straddling across S_{i_1,j_1} and S_{i_2,j_2}. Hence, when counting triangles in $S_{i,j}$, only all vertices in $S_{i,j}$ and some vertices in its neighboring clusters need to be considered.

C. Counting Step

The remaining triangles are counted as follows. According to Lemma II.1, for each triangle $\{v_i, v_j, v_k\}$ in G, there are only two possible structures. The first structure is that v_i, v_j and v_k scatter in two neighboring clusters, and we call $\{v_i, v_j, v_k\}$ an inter-cluster triangle. In this case, one of the three vertices may be the root. The second structure is that v_i, v_j and v_k are in the same cluster, and we call it an intra-cluster triangle. In the following, we describe how different structures of triangles are counted in the counting step.

1) Inter-cluster triangles with one vertex as the root: Since each edge in $E(S_{i,1})$ forms a triangle with the root v^i, there are altogether $\sum_{i=1}^{t} |E(S_{i,1})|$ first type inter-cluster triangles in G'.

2) Non-root inter-cluster triangles: For each vertex $v \in S_{i,j}$, we define its upper neighbors as $N^{\uparrow}(v) = N_{G'}(v) \bigcap S_{i,j-1}$ and its lower neighbors as $N^{\downarrow}(v) = N_{G'}(v) \bigcap S_{i,j+1}$. For a fixed vertex $v \in S_{i,j}$, for each $u_1^{\uparrow}$ and $u_2^{\uparrow}$ belonging to $N^{\uparrow}(v)$ as well as $v_1^{\downarrow}$ and $v_2^{\downarrow}$ belonging

```
Data: Graph G', vertex sets S_{i,j}, number t
Result: answer, the number of triangles in the given
        graph G.
answer= t;
for i do
    for j do
        G_{i,j} = (S_{i,j}, E(S_{i,j}));
        PHC(G_{i,j}) is the output we get if we put G_{i,j}
         into algorithm PHC.
         answer←answer+PHC(G_{i,j});
        for v ∈ S_{i,j} do
            N↑(v) = N(v) ⋂ S_{i,j−1};
            N↓(v) = N(v) ⋂ S_{i,j+1};
            for (a, b) ∈ N↑(v)^2 + N↓(v)^2, a > b do
                if a and b are adjacent then
                    answer←answer+1;
                end
            end
        end
    end
    answer←answer+|E(S_{i,1})|;
end
return answer
```

Step 3: Counting step with inputs G' and $S_{i,j}$

to $N^{\downarrow}(v)$, we check if $(u_1^{\uparrow}, u_2^{\uparrow}) \in E(G)$ and $(v_1^{\downarrow}, v_2^{\downarrow}) \in E(G)$ or not. If so, then it can be $u_1^{\uparrow}$, $u_2^{\uparrow}$ and v form a triangle or $v_1^{\downarrow}$, $v_2^{\downarrow}$ and v form a triangle. Hence, there are at most $\sum_{v \in V(G'),\ v \text{ is not the root}} \binom{|N^{\uparrow}(v)|}{2} + \binom{|N^{\downarrow}(v)|}{2}$ second type inter-cluster triangles in G'.

3) Intra-cluster triangles: Let $G_{i,j} = (S_{i,j}, E(S_{i,j}))$ be an induced subgraph of G'. Then, all triangles in $G_{i,j}$ can be counted by a recursive approach, i.e., we apply the PHC algorithm with the input $G_{i,j}$ again. Let $\mathsf{PHC}(G)$ denote the number of triangles counted in G by the PHC algorithm, then there are $\sum_{i,j} \mathsf{PHC}(G_{i,j})$ triangles counted altogether.

D. Algorithm Implementation

We use an example to illustrate how the PHC algorithm works on a given graph G and the pseudocode of each step of the PHC algorithm is also shown in this section.

Example II.1. We build a simple graph G as shown in Fig. 1. In the pruning step, we prune the graph G with the threshold $M = 2$, then we obtain G' shown in Fig. 2. Note that, during the pruning step, we have 1 triangle counted. In the hierarchical clustering step, let v^1 denote the vertex with the maximum degree in the first connected component G_1, and we can construct each cluster of G_1 by its definition $S_{1,j} = \{v \in V(G') | \mathsf{dist}(v, v^1) = j\}$. Also, we can find v^2 and construct $S_{2,j}$ for each possible j in the second connected component G_2. The hierarchical clustering step is shown in Fig. 2. Let $G_{i,j}$ be defined as an induced subgraph of G with $V(G_{i,j}) = S_{i,j}$. Then, we have that there are

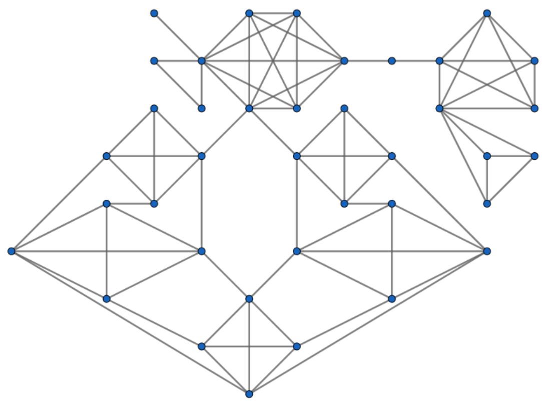


Fig. 1. The simple graph G considered in Example II.1.

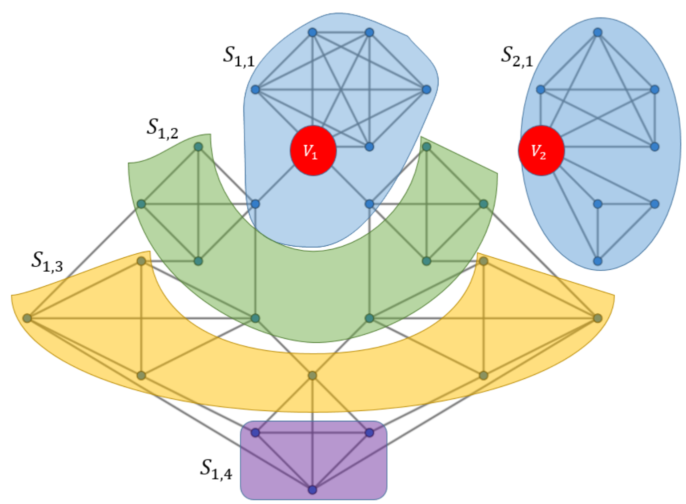


Fig. 2. The pruned graph G'. The left cluster constructed in the first connected component G_1 with four clusters and G_2 on the right-hand side has a single cluster cluster.

$|E(G_{1,1})| + |E(G_{2,1})| = 10 + 9 = 19$ inter-cluster triangles (will be counted in the next step) with one vertex as the root which are shown in Fig. 3. In the counting step, besides the above nineteen inter-cluster triangles, there are six edges in $S_{1,2}$ such that their end vertices have a common neighbor in $S_{1,1}$, i.e., there are six inter-cluster triangles between $S_{1,1}$ and $S_{1,2}$. Also, there are six inter-cluster triangles between $S_{1,2}$ and $S_{1,3}$, and three inter-cluster triangles between $S_{1,3}$ and $S_{1,4}$. Hence, there are in total $6 + 6 + 3 = 15$ non-root inter-cluster triangles. To count the intra-cluster triangles, we only need to apply the PHC algorithm to those induced subgraphs again. Then we have altogether $10 + 2 + 2 + 1 + 5 = 20$ intra-cluster triangles. Finally, we can conclude that the total number of triangles in this example is $1 + 19 + 15 + 20 = 55$.

E. Time Complexity

The time complexity of the pruning step is dependent on the number of pairs of vertices that need to be verified whether an edge exists between them. Let P_M denote the set of vertices

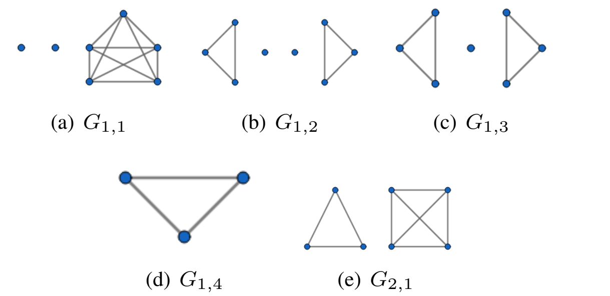


Fig. 3. Example of induced subgraphs of G considered in the counting step.

that will be pruned off when the threshold is set to be M, i.e., $P_M = \{v \in V(G) | v \notin V(G')\}$. Then, the time complexity of the pruning step is bounded above by $\binom{M}{2} \cdot |P_M|$. Assume that the threshold M is much smaller than N, where N is the number of vertices in the graph. Then, the time complexity of the pruning step is bounded above by $O(N)$ since $|P_M| \leq N$. In the hierarchical clustering step, for each G_i, we need to find the vertex v^i with the maximum degree as the root of G_i and start a BFS from v^i on G_i to obtain the distances from v^i to any other vertices in G_i. Hence, the time complexity of the hierarchical clustering step is $O(|N| + |E|) = O(|E|)$, since in the triangle counting problem we mostly consider graphs in which $|N| \leq |E|$. In the counting step, each vertex v has three types of neighbors which are described as follows, $N_G^{\uparrow}(v)$, the neighbors existing in the upper cluster, $N_G^{\downarrow}(v)$,the neighbors existing in the lower cluster, and $N_G^{\rightarrow}(v)$, the neighbors existing in the same cluster. Therefore, we only need to check at most $\binom{|N_G^{\uparrow}(v)|}{2} + \binom{|N_G^{\downarrow}(v)|}{2} + \binom{|N_G^{\rightarrow}(v)|}{2} \leq \binom{d_{max}}{2}$ pairs of vertices for each vertex v. We can conclude that the time complexity of the counting step is $O(N \cdot \binom{d_{max}}{2})$. Thus, the total time complexity of the PHC algorithm is $O(|E| + N \cdot \binom{d_{max}}{2})$.

III. PERFORMANCE EVALUATION

In this section, we evaluate the PHC algorithm on undirected graphs with different degree distributions provided by the Stanford SNAP in TABLE II.

TABLE II
THE LIST OF GRAPHS FROM SNAP WITH THEIR SIZES, AND THE NUMBER OF TRIANGLES IN EACH GRAPH IS DENOTED BY $|T|$.

Graph	$\|V\|$	$\|E\|$	$\|T\|$
facebook	4039	88234	1612010
com-youtube	1134890	2987624	3056386
com-dblp	317080	1049866	2224385
com-amazon	334863	925872	667129
email-Enron	36692	183831	727044
ca-HepPh	12008	118521	3358548
CA-CondMat	23133	93497	173435
roadNet-CA	1965206	2766607	120676
as-skitter	1696415	11095298	28769868
loc-gowalla	196591	950327	2273138

TABLE III
COMPARISON OF OUR ALGORITHM WITH THAT IN [11].

Graph	Time ($M=1$)	Time ([11])	Time (M)
facebook	3.78	2.312	1.979 (79)
com-youtube	145.21	46.668	25.147 (51)
com-dblp	19.49	13.46	5.695 (15)
com-amazon	18.26	12.718	3.047 (5)
email-Enron	5.28	2.907	1.667 (43)
ca-HepPh	3.27	4.047	1.52 (90)
CA-CondMat	1.72	1.138	0.456 (16)
roadNet-CA	15.2	91.216	8.867 (3)
as-skitter	3359.05	182.63	133.983 (75)
loc-gowalla	36.28	14.802	9.371 (51)

The time in TABLE III is given in seconds. The parenthesis in the last column indicates the choice of the parameter M.

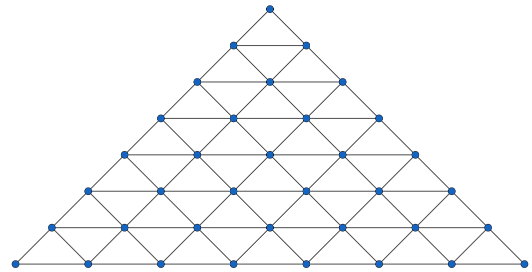


Fig. 4. An example in which the threshold M is set to be 3 and the pruned graph G' becomes a null graph.

A. Experimental Setup

All the experiments are conducted on a 64-bit computer running a Windows 10 system with Intel(R) Core(TM) i7-2600 CPU 3.40GHz and 8.00 GB RAM configuration.

B. Comparison Between Different Algorithms

1) Time Complexity: We compare the PHC algorithm with the method in [11] published in Graph Challenge 2017 [1] using the SNAP datasets. The work in [11] assigns direction to each edge according to their degree. This implies that, for each vertex v, there are $\binom{\text{outdegree}(v)}{2}$ pairs of vertices to be checked. Hence, the time complexity of the algorithm provided in [11] is $O(|E|+N\cdot\binom{d_{\max}}{2})$, where $|E|$ is the time complexity of the direction assignment, and $d_{\max}$ is the maximum out-degree. Note that this time complexity is equivalent to the time complexity of the PHC algorithm. However, if we can find a suitable threshold M that leverages the structure of the graph in advance, then the PHC algorithm can complete the triangle counting task at the pruning step with a time complexity of $O(N)$ and outperform the algorithm in [11].

C. Algorithm on Large Graphs in Public Domain

In practice, we leverage the SNAP library and tools [12] to implement the PHC algorithm. We evaluate the performance of the PHC algorithm on various graphs provided by SNAP datasets [13]. In TABLE II, we list out the information of each graph in our experiments. Let $|V|$ and $|E|$ denote the size of the vertex set and the edge set of a given graph respectively, and let $|T|$ denote the number of triangles in the graph.

D. Trade-Off between Pruning and Hierarchical Clustering

The threshold M is an interesting parameter in the pruning step. Once M is set, there are at most $\binom{M}{2}\cdot|P_M|$ pairs of vertices that need to be checked. Hence, a larger M leads to a longer time spent for the pruning step. On the other hand, if more vertices are pruned off from G, then the graph becomes more fragmented. Thus, there is a trade-off between the pruning step and the hierarchical clustering step, which is dependent on the threshold M. However, finding a suitable threshold for an arbitrary graph is not a straightforward task unless we further exploit the inherent structure of the graph. For instance, in Fig. 4, a triangular graph can be totally decomposed in the pruning step with threshold $M=3$.

E. MapReduce Software Implementation

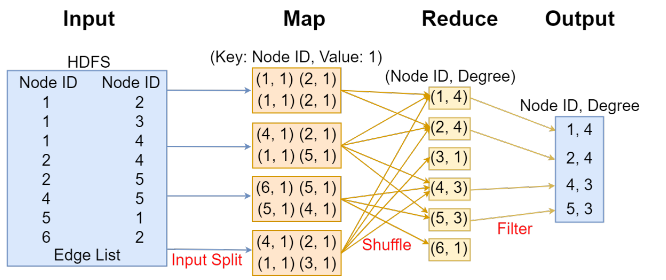
The parallel computational structure in our PHC algorithm can be naturally mapped into existing popular scalable software framework such as MapReduce. In particular, for the pruning step, a Mapper operation can count the degrees and a single Reducer identifies nodes that do not have degrees 0, 1 and $N'-1$. For the hierarchical clustering step, Mapper operations can be delegated to count triangles in each cluster and also triangles that straddle clusters. For this purpose, we create a data structure to store the nodes in each cluster, and set the custom input split size to ensure that each Mapper can access a complete cluster at any one time. The total count from the Mappers can then be obtained by a single Reducer. Counting of triangles that straddle across clusters requires setting larger input split size so that a mapper accesses two clusters at any time. Fig. 5 shows an example of using MapReduce on PHC algorithm. For more details of the MapReduce implementation, please see: `https://github.com/Graph-Challenge`.

IV. EVALUATION USING LARGE RANDOM GRAPHS

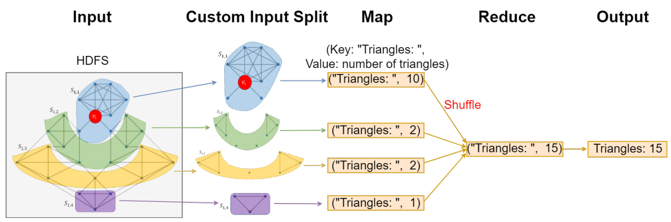
According to the experimental results of real world graphs from SNAP, the threshold M is a critical parameter in the PHC algorithm. In TABLE III, the column with the header "Time($M=1$)" shows the amount of time taken by the PHC algorithm if we prune the graph with the smallest threshold $M=1$. The next column with the header "Time([11])" shows the time taken by the algorithm provided in [11]. The last column with the headers "Time(M)" indicate an appropriate threshold M and the corresponding time taken for this M respectively. In particular, our results show that if M is carefully-chosen, then the PHC algorithm can outperform the algorithm in [11].

In the case of random graphs, we apply the PHC algorithm to a well-known random graph model, namely the Barabási-Albert (BA) model, and we conduct ten random graph simulations. The generation of the BA graph starts with K_{10}. Subsequently, new vertices connect to the previous vertices

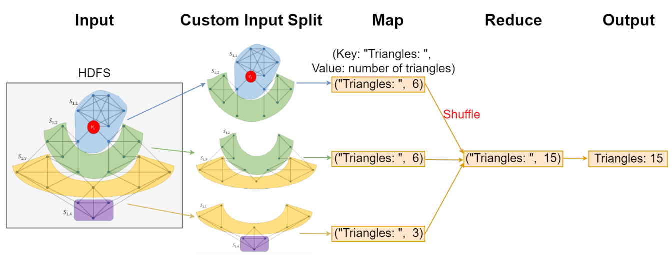
978-1-5386-5989-2/18/$31.00 ©2018 IEEE



(a) Pruning Step



(b) Intra-cluster Triangles



(c) Inter-cluster Triangles

Fig. 5. MapReduce implementation on PHC algorithm

so as to obtain a random graph with a million vertices and a total of 9999945 edges.

For each BA graph, we apply the pruning step with a suitable threshold $M = 10$ (that is equal to the parameter in the BA model) to totally decompose the BA graph. The experimental results are shown in TABLE IV. We observe that although there are some large degree vertices in each BA graph, it can still be decomposed with such M. Moreover, the time complexity is simply $O(N)$ since we only execute the pruning step with a small threshold. Accordingly, if the problem is to count the number of triangles in a BA-model-like network, then we may simply set the threshold $M = \lceil \frac{|V|}{|E|} \rceil$ in the PHC algorithm.

V. Conclusions and Future Work

We proposed the PHC algorithm that first employed careful pruning and then hierarchical clustering to efficiently count the exact number of triangles in large graphs. Interestingly, particularly for graphs with distinctive structure, e.g., when the graphs were similar to Barabási-Albert model random graph model, we showed the optimal configuration to decompose the large graph into highly parallelizable clusters with a time complexity $O(N)$. In our current work, geodesic distance is used to differentiate clusters, and it will be interesting to compare alternative graph-theoretic metrics for clustering.

TABLE IV
Barabási-Albert model

Graph	maximum degree	Time ([11])	Time ($M = 10$)
BAGraph01	5356	147.009	59.77
BAGraph02	4933	146.013	62.347
BAGraph03	6062	140.967	59.676
BAGraph04	5324	138.184	63.438
BAGraph05	5331	139.278	65.527
BAGraph06	4876	138.687	62.361
BAGraph07	5878	142.472	60.508
BAGraph08	5422	138.945	62.451
BAGraph09	4579	138.69	61.774
BAGraph10	4954	138.394	61.798

The time in TABLE IV is given in seconds.

Another future work is to extend the PHC algorithm to count more complex subgraphs and to study the optimal parameter tuning of the hierarchical clustering and more efficient parallel software implementation.

References

[1] Siddharth Samsi, Vijay Gadepally, Michael Hurley, Michael Jones, Edward Kao, Sanjeev Mohindra, Paul Monticciolo, Albert Reuther, Steven Smith, William Song, Diane Staheli, and Jeremy Kepner. GraphChallenge.org: Raising the bar on graph analytic performance. arXiv: https://arxiv.org/abs/1805.09675v1, 2018.

[2] Luca Becchetti, Paolo Boldi, Carlos Castillo, and Aristides Gionis. Efficient semi-streaming algorithms for local triangle counting in massive graphs. In *Proceedings of the 14th ACM (KDD) international conference on Knowledge discovery and data mining*, pages 16–24. ACM, 2008.

[3] Koryo Miura and Yasuyuki Miyazaki. Concept of the tension truss antenna. *AIAA journal*, 28(6):1098–1104, 1990.

[4] Duncan J Watts and Steven H Strogatz. Collective dynamics of small-world networks. *nature*, 393(6684):440, 1998.

[5] Walter Schnyder. Embedding planar graphs on the grid. In *Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms*, pages 138–148. Society for Industrial and Applied Mathematics, 1990.

[6] Comandur Seshadhri, Ali Pinar, and Tamara G Kolda. Triadic measures on graphs: The power of wedge sampling. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 10–18, 2013.

[7] Charalampos E Tsourakakis. Fast counting of triangles in large real networks without counting: Algorithms and laws. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 608–617, 2008.

[8] Rasmus Pagh and Charalampos E Tsourakakis. Colorful triangle counting and a mapreduce implementation. *Information Processing Letters*, 112(7):277–281, 2012.

[9] Robert Endre Tarjan. A hierarchical clustering algorithm using strong components. *Information Processing Letters*, 14(1):26 – 29, 1982.

[10] Richard J. LIPTON and Robert Endre Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979.

[11] Roger Pearce. Triangle counting for scale-free graphs at scale in distributed memory. In *High Performance Extreme Computing Conference (HPEC), 2017 IEEE*, pages 1–4. IEEE, 2017.

[12] Jure Leskovec and Rok Sosič. Snap: A general-purpose network analysis and graph-mining library. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(1):1, 2016.

[13] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

978-1-5386-5989-2/18/$31.00 ©2018 IEEE