Agent-Based Triangle Counting and its Applications in Anonymous Graphs

Extended Abstract

Prabhat Kumar Chand
Indian Statistical Institute
Kolkata, India
pchand744@gmail.com

Apurba Das
BITS Pilani
Hyderabad, India
apurba@hyderabad.bits-pilani.ac.in

Anisur Rahaman Molla
Indian Statistical Institute
Kolkata, India
molla@isical.ac.in

ABSTRACT

Triangle counting in graphs is a fundamental problem with a diverse application domain. In this paper, we propose a solution to the triangle counting problem in an anonymous graph using autonomous mobile agents. We further use the triangle count to address the Truss Decomposition problem which involves finding maximal subgraphs with strong interconnections. Truss decomposition helps in identifying maximal, highly interconnected subgraphs, or trusses, within a network. Additionally, the triangle count is also used to compute two important metrics - Triangle Centrality and Local Clustering Coefficient for the nodes of the graph. Our goal is to devise algorithms that effectively solve these problems minimizing both the overall time complexity and the memory usage at each agent.

KEYWORDS

Mobile Agents; Triangle Counting; $k$-Truss; Truss Decomposition; Triangle Centrality; Local Clustering Coefficient; Time Complexity; Space Complexity; Network Algorithms; Distributed Algorithms

2 INTRODUCTION AND RELATED WORK

Triangle counting in graphs has received significant attention in recent decades, serving as a building block of complex network analysis. It is used for computing the clustering coefficient, one of the most used metrics for network analysis [5, 19, 25], and triangle centrality [1, 6, 15]. Triangle counting also plays a pivotal role in the hierarchical decomposition of a graph such as truss decomposition [24], an important hierarchical subgraph structure in community detection [2, 11]. In addition, Becchetti et al. [4] used triangle counts in detecting web spam and estimating the content quality of a web page. Other applications include query optimization in databases [3], link prediction in social networks [22], and community detection in system biology [12]. A detailed account of related works on triangle counting for various model set-ups may be found in [3, 4, 10, 14, 16, 20–23].

On the other hand, our agent-based model has been gaining significant attention recently. For example, there have been some recent works on positioning the agents on nodes of the graph $G$ such that each agent’s position collectively form the maximal independent set (MIS) [17, 18] or they identify a small dominating set [8] of $G$. Another related problem is of dispersion in which $k \leq n$ agents are positioned on $k$ different nodes of $G$, see [13] and the references therein. A solution to the dispersion problem guarantees that $k$ agents are positioned on $k$ different nodes; which is a requirement for the triangle counting problem defined in this paper. Exploration problem on graphs using mobile agents refers to solving a graph analytic task using one or more agents [9].

In this work, we consider triangle counting in a simple, undirected, anonymous graph using mobile agents and then use it as a sub-routine to solve the (i) Truss Decomposition Problem and compute (ii) Triangle Centrality and (iii) Local Clustering Coefficient. Our solution to the truss decomposition problem is based on $h$-index based parallel truss decomposition algorithm described in [26]. We study these problems from a theoretical perspective and aim to solve them while minimizing both time and memory-per-agent as much as possible. The full version of the paper can be found in [7].

2 PRELIMINARIES, PROBLEM AND RESULTS

2.1 Model

We have $G(V, E)$ - a connected, undirected, unweighted and anonymous graph with $n$ nodes and $m$ edges. The nodes of $G$ do not have any identifiers and are memoryless. Edges incident on $v$ are locally labelled using port numbers. The edges of the graph serve as routes through which the agents can commute.

We have a collection of $n$ agents residing on the nodes of the graph in such a way that each node is occupied by a distinct agent at the start (known as dispersed configuration in literature). Each agent has a unique ID and memory to store information. Two or more agents can be present (co-located) at a node or pass through an edge in $G$. The agents operate in a synchronous system where they are synchronised to a common clock. We consider the local communication model where only co-located agents (i.e., agents at the same node) can communicate among themselves. An agent can perform a Communicate – Compute – Move task in a time unit, called round. The time complexity of an algorithm is the number of rounds required to achieve the goal. The space complexity is the number of bits required by each agent to execute the algorithm.
2.2 Definitions

Definition 2.1 (support [26]). For a given graph \( G(V,E) \), the support of an edge \( e \in E \) is the number of triangles in \( G \) that contain \( e \).

Definition 2.2 (k-truss [26]). A k-truss is the largest sub-graph \( T_k \) of \( G(V,E) \) in which every edge has support \( \geq k-2 \) with respect to \( T_k \). In case, \( T_k \) is a null graph, we say \( k \)-truss for \( G \) does not exist.

Definition 2.3 (trussness [26]). The trussness of an edge \( e \), is defined as the maximum \( k \) such that \( e \) belongs to \( T_k \) but not to \( T_{k+1} \).

Definition 2.4 (Triangle Centrality [6]). Triangle Centrality, \( TC(v) \) of a node \( v \in G \) is given by the equation:

\[
TC(v) = \frac{1}{2} \sum_{u \in N^2_v} T(u) + \sum_{w \in (N(v) \setminus N_T(v)} T(w)
\]

where, where \( N(v) \) is the neighborhood set of \( v \), \( N_T(v) \) is the set of neighbors that are in triangles with \( v \), and \( N^2_v \) is the closed set that includes \( v \). \( T(v) \) and \( T(G) \) denote the respective triangle counts with \( v \) as a vertex and the total triangle count in \( G \).

Definition 2.5 (Local Clustering Coefficient [19]). The Local Clustering Coefficient (LCC) of a node \( v \in G \) is written as \( LCC(v) = \frac{T(v)}{\delta(v)(\delta(v) - 1)} \), where \( T(v) \) is the number of triangles with \( v \) as a vertex and \( \delta(v) \), the degree of the node \( v \).

2.3 Problem Statements

Triangle Counting using Mobile Agents: Consider an undirected, simple, connected anonymous \( n \)-node graph \( G = (V,E) \) and a collection \( \mathcal{R} = \{r_1, r_2, \ldots, r_n\} \) of \( n \) agents, each of which is initially placed distinctly at each node of \( G \). The \( n \) autonomous agents coordinate among themselves to solve the following problems:

(a) Node-Based Triangle Counting: To count the number of triangles with a given node as a vertex.

(b) Edge-Based Triangle Counting: To count the number of triangles based on a given edge.

(c) Total Triangle Counting: To count the total number of triangles in the graph \( G \).

Truss Decomposition, Triangle Centrality and Local Clustering Coefficient: Consider an undirected, simple, connected anonymous \( n \)-node graph \( G = (V,E) \) and a collection \( \mathcal{R} = \{r_1, r_2, \ldots, r_n\} \) of \( n \) agents, each of which is initially placed distinctly at each node of \( G \). The \( n \) autonomous agents coordinate among themselves to solve the (i) Truss Decomposition Problem and compute (ii) Triangle Centrality and (iii) Local Clustering Coefficient of a given node.

2.4 Our Results

Let \( G \) be an \( n \) node arbitrary, anonymous, simple, connected graph with \( m \) edges, maximum degree \( \Delta \) and diameter \( D \). Let \( n \) mobile agents with distinct IDs with the highest ID \( \lambda \), be placed at each of the \( n \) nodes of \( G \) in an initial dispersed configuration. Then, we have the following results:

**Theorem 2.6 (Triangle Counting).** Each agent \( r_i \) with \( O(\Delta \log n) \) bits of memory, can calculate the number of triangles with \( r_i \) as a vertex in \( O(\Delta \log \lambda) \) rounds, the number of triangles based on each of its adjacent edges in \( O(\Delta \log \lambda) \) rounds and the total number of triangles in \( G \), in \( O(D \Delta \log \lambda) \) rounds.

**Theorem 2.7 (Truss Decomposition).** The Truss Decomposition Problem for \( G \) can be solved by the mobile agents in \( O(m \Delta D \log \lambda) \) rounds with \( O(\Delta \log n) \) bits of memory per agent.

**Theorem 2.8 (Triangle Centrality).** The Triangle Centrality of each node \( v \in G \) can be calculated in \( O(\Delta \log \lambda) \) rounds if \( T(G) \) is known and in \( O(D \Delta \log \lambda) \) rounds, if \( T(G) \) is unknown. \( T(G) \) is the total triangle count of the graph \( G \).

**Theorem 2.9 (Local Clustering Coefficient).** The Local Clustering Coefficient of each node \( v \in G \), \( LCC(v) \) can be calculated in \( O(\Delta \log \lambda) \) rounds.

3 TRIANGLE COUNTING VIA MOBILE AGENTS

In this section, we formulate algorithms for \( n \) mobile agents that are initially dispersed among the \( n \) nodes of the graph \( G \) to count the number of triangles within \( G \). Due to the indistinguishable nature of the nodes, the algorithm relies on the memory and IDs of the mobile agents stationed on these nodes. Moreover, the limitation of communication only within the co-located agents and synchronising their movement emerges as an additional challenge.

Our algorithm operates in three phases:

- **Phase 1** (Neighbourhood Discovery): Agents, symbolically representing their respective nodes, initially scan their neighbourhoods to gather information about adjacent nodes.

- **Phase 2** (Common Neighborhood Counting): Once the neighbourhood information is collected, agents count the number of common neighbours with each adjacent agent. Additionally, each agent \( r_i \) tallies local triangles with itself as a vertex and triangles with \( \{r_i, r_j\} \) as an edge, where \( r_j \) is an adjacent agent to \( r_i \).

- **Phase 3** (Total Triangle Counting): In the final phase, each agent consolidates the local triangle count from every other agent, enabling the determination of the total number of triangles in the graph \( G \).

The results attained throughout the three phases are summarized in Theorem 2.6. For details, refer to the full version [7].

4 APPLICATIONS

**Truss Decomposition:** In truss decomposition, we compute the trussness for each edge in \( G(V,E) \), to obtain a partition (equivalence classes) of \( E \), thereby obtaining the \( k \)-trusses of \( G \) for any \( k \) by taking union of the equivalent classes. Using methods from Section 3, we obtain trussness values for each edge, thus solving the TRUSS DECOMPOSITION PROBLEM. The main result of this section is provided in Theorem 2.7. For more details, refer to [7].

**Triangle Centrality and Local Clustering Coefficient:** Triangle Centrality, formulated in [6], identifies key vertices in a graph by assessing the concentration of triangles around each vertex. The Local Clustering Coefficient of a node measures the proximity of its neighbours to forming a clique, indicating the network’s connectivity around that node. We employ the mobile agents to compute these metrics. Our results on Triangle Centrality and Local Clustering Coefficient have been summarized in Theorem 2.8 and Theorem 2.9, respectively.
REFERENCES


