Linear-Time Graph Triples Census Algorithm
Under Assumptions Typical of Social Networks

Daniel McEnnis

August 20, 2009

©Daniel McEnnis
Department of Computer Science
The University of Waikato
Private Bag 3105
Hamilton, New Zealand
1. Introduction

A graph triples census is a histogram of all possible sets of three vertices (called a triple) from a graph. Graph triples census have been in active use in sociology for over 50 years. The earliest paper using this approach is by Holland and Leinhardt [1]. This gives a general description of the structure of directed graphs in a fixed length vector. Since this time, this analytic tool has been widely used in social network analysis. A summary of important papers using this approach, both as end product and as a component of further analysis, are in [2].

Graph Triples Census is also an important tool in machine learning for capturing information about relational structure of a data set in a form that can be fed to non-relational machine learning algorithms. These approaches are still in their infancy largely because of a lack of effective, time efficient algorithms for describing large scale structure—especially for large networks such as on-line friendship networks and the structure of the Internet with its underlying communities. All of these graphs have the property that the average number of links per node is small compared to the number of vertices and, likewise, the max degree is small compared to the number of vertices. In many cases, both average and max degree are explicitly limited to a small constant by the structure of the source data. One example of this is the sociological data collected by Harris et al. [3]—widely used in social network analysis. Similar patterns have been identified by the author in LiveJournal and LastFM friendship networks.

Existing algorithms are discussed in related work. This followed by definitions, the algorithm description, proof of correctness, proof of time complexity, and proof of space complexity.

2. Definitions

Throughout this paper, we are concerned with a graph $G = (V, E)$ with a finite set $V$ of vertices and a finite set $E$ of ordered pairs of distinct vertices
called edges denoted $e(v_i, v_j)$. Vertex $w$ is considered a neighbor of vertex $v$ iff $(v, w) \in E$

1. Let $v_i$ denote an arbitrary ordering of vertices in $V$ from 0 to $|V| - 1$
2. Let $N(v_i)$ denote the set of all neighbors of $v_i$.
3. Let $G N(v_i)$ denote the set of all $v_f \in N(v_i)$ such that $v_f > v_i$
4. Let $G_3(G)$ denote a graph of three ordered vertices $i, j, k \in G$ where $i < j < k$ with undirected edges present if $e(v_l, v_m)$ exists $\forall l, m \in i, j, k$.
5. Let $L G_3(G)$ denote a graph of three ordered vertices $i, j, k \in G$ where $i < j$ and $k \neq i, j$ with undirected edges present if $e(v_l, v_m)$ exists $\forall l, m \in i, j, k$.
6. Let $A(G)$ be the set of all $G_3$ such that $\forall v_i, v_j, v_k \in G$ where $i < f < g$ $G_3(v_i, v_j, v_k) \in A$
7. Let $E_n(G)$ be the subset of $A(G)$ such that $\forall G_3 \in A$ where there are $n$ edges present.
8. Let $E_2a(G)$ be the subset of $A(G)$ such that $\forall G_3 \in A$ with 2 edges present and $e(v_i, v_j)$ or $e(v_j, v_i)$ exists.
9. Let $E_2b(G)$ be the subset of $A(G)$ such that $\forall G_3 \in A$ with 2 edges present and $e(v_i, v_j)$ and $e(v_j, v_i)$ does not exist.
10. Let $addToCensus(e_1, e_2, e_3, x)$ define a procedure that increments the graph triple equivalence class that corresponds to this combination of link types by value $x$ in O(4) time and O(0) space.
11. Let $linkType(v_1, v_2)$ define a procedure that returns one of the four link types (0-4: no link, lesser to greater, greater to lesser, bidirectional) in O(6) time and O(1) space.

3. Algorithm

The algorithm enumerates smaller census entries first, then calculates the remainder of the census entries using set compliments to avoid counting their entries individually.

let $count = 0$
for $v_i \in V(G)$ // loop 1
$G N_i, N_i = getLinks(V_i, V_i)$ // link 1
for $v_j \in G N_i$ // loop 2
$count++$
// enumerate $E_3(G)$
$G N_j, N_j = getLinks(V_j)$ // link 2
for $v_k \in (G N_i \cap G N_j)$ // loop 3a
addToCensus(linkType($v_i, v_j$), linkType($v_j, v_k$), linkType($v_i, v_k$), 1)
rof
// enumerate $E_2(G)$
for $v_k \in (N_j \cap N_i)$ // loop 3b
addToCensus(linkType($v_i, v_j$), linkType($v_j, v_k$), 0)
rof
addToCensus(linkType($v_i, v_j$), 0, $|V(G)| - |N_i \cup N_j|$) // link 3

2
4. Proof of Correctness

**Theorem 1.** The census of triples of $G$ can be calculated using neighbor properties and set compliments.

**Proof of Theorem.** By definition of graph triple census, the census is a count of the size of the set of each equivalence class of $G^3 \in G$. Subdivide the set of all $G^3$ into the subsets $E_n$ and prove that each member of each subset is counted.

**Note 1.** the sum of all entries in the graph triple census is $|V(G)^3|/6$

**Definition 1.** $\forall v_i, v_j \in G$ such that $i < j$ define $A_{ij}$ as the set of all $LG^3(G)$ where $i = v_i$ and $j = v_j$

**Note 2.** $|A| = |V(G) - 2|$ consider $E_0$

**Note 3.** $\forall edge \in G, \exists exactly|V(G) - 2|triples \notin E_0$

$$\Rightarrow |E_0| = |E(G)||(|V(G)| - 2)$$ as in Link 4, enumerating $E_0$

consider $E_3$

**Definition 2.** The set $Triple$ as the set of all $i, j$ such that $Triple$ contains the members of all $A_{ij}$ where $k \in GN(v_i) \cap GN(v_j)$

**Note 4.** $Triple$ is equivalent to $E_3$

**Note 5.** $Triple$ is enumerated by Loop 3a

**Definition 3.** define the set $Double$ as the set of all $i, j$ such that $Double$ contains the members of all $A_{ij}$ where $k \in N(v_j)$ and $k \notin N(v_i)$

**Note 6.** $Double$ is enumerated by Loop 3b

**Definition 4.** The set $Double_a$ as the subset of $Double$ such that $i < k, j < k$

**Note 7.** Note that this set enumerates $E_{3a}$

**Definition 5.** The set $Double_b 1$ as the subset of $Double$ such that $i < k < j$

**Definition 6.** The set $Double_b 2$ is the subset of $Double$ such that $k < i < j$

**Note 8.** $Double_b 1 \cup Double_b 2 = E_{3b}(G)$ and $Double_b 1 \cap Double_b 2 = \{\}x1$
Loop 3b enumerates $E_2(G)$
Consider $E_1(G)$

**Note 9.** Given $i, j \in G$ with an edge between them, the size of the subset of $A_{ij}$ with $k > j$ and $\in E_3(G)$ or $\in E_2(G) = N_i \cup N_j$

**Note 10.** $\forall v_i, v_j \in G$ with an edge between them, there exists $|V(G)| - j$ G3 containing $v_i, v_j$.

\[ \Rightarrow \forall i, j \in G \text{ where } e(i, j) \text{ exists or } e(j, i) \text{ exists } \sum_{i=1}^{|E(G)|} \sum_{j=1}^{|N_i|} |V(G)| - N_i \cup N_j = |E_1(G)| \]
\[ \Rightarrow \text{Link 3 enumerates } E_1(G) \]
\[ \Rightarrow \text{the algorithm enumerates all graph triples of } G. \]

5. Worst-Case Time Complexity Under Assumptions

**Theorem 2.** The time complexity is $|V(G)|$ for all $G$ where the assumptions hold.

**Proof of Theorem.** Consider the time complexity of each statement as the time-complexity of the operation times the maximum number of times the statement could be executed.

**Assumption 1.** $G$ has a vertices index with $O(3)$ access time using hashtables

**Assumption 2.** $G$ has two edge indeci by both source and destination with $O(3)$ access time using hashtables

**Assumption 3.** $|E(G)| = m|V(G)|$ where $m \in R$ is a small constant.

**Assumption 4.** $\max(|N_i|) = n$ where $n \in N$ is a small constant.

**Note 11.** $\forall iGN_i$ is at most $O(3|N_i|)$

**Note 12.** $\forall iN_i$ is at most $O(3|N_i|)$

**Note 13.** $3$ executes $|V(G)|$ times

\[ \Rightarrow GN_i \text{ and } N_i \text{ are } O(\sum_{i=1}^{|V(G)|} 3|N_i|) \]
\[ \Rightarrow GN_j \text{ and } N_i \text{ are } O(3|E(G)|) \]
\[ \Rightarrow GN_j \text{ and } N_i \text{ are } O(3m|V(G)|) \]

**Note 14.** $3$ is executed $|E(G)|$ times which is $\sum_{i=1}^{|V(G)|} |N_i|$

by Lemma 1 $GN_j$ and $N_j = O(\sum_{i=1}^{|V(G)|} \sum_{j=1}^{|N_i|} 3|N_j|)$

**Note 15.** $\text{addToCensus(linkType}(v_i, v_j), \text{linkType}(v_j, v_k), \text{linkType}(v_i, v_k), x) = O(9) \forall x \in N$
Note 16. The number of iterations of Loop 3a are $O(\sum_{i=1}^{V(G)} \sum_{j=1}^{N_i} 3|N_i \cap N_j|)$
by Lemma 2 Loop 3a has $O(mn|V(G)|)$ iterations
by definition of addToCensus and linkType, Loop 3a has $O(9mn|V(G)|)$

Note 17. The number of iterations of Loop 3b are $O(\sum_{i=1}^{V(G)} \sum_{j=1}^{N_i} 3|N_j \not\cap N_i|)$
by Lemma 3, the number of iterations of Loop 3b is $O(9/4mn\sqrt{n}|V(G)|)$
by definition of addToCensus and linkType, Loop3b is less than $O(9/4mn|V(G)|)$

Note 18. the time complexity of $\sum_{i=1}^{V(G)} \sum_{j=1}^{N_i} |N_i \cup N_j| \in O(2n|E(G)|) = O(mn|V(G)|)$
⇒ time complexity of the algorithm is $O(3m|V(G)| + 3n|V(G)| + 9mn|V(G)| + 9/4mn|V(G)| + mn|9V(G)| + 9)$

Lemma 1. time complexity to create $N_i$ is $O(m|V(G)|)$

Note 19. $\sum_{i=1}^{V(G)} |N_i|$ is maximized, within assumptions of maximum degree
and $-E(G)$, by the graph $g$ where there are $x$ cliques of degree $n$.

⇒ $x = |E(G)|/n^2$
⇒ $\forall v_i \in$ cliques of $g$, time complexity of the clique is $\sum_{i=1}^{n} 3|N_j| = 3c^2$
⇒ $\forall v_i \not\in$ cliques of $g$, time complexity is $O(0)$
⇒ time complexity to create $|N_j| < O(3x^2 + 0)$
⇒ time complexity to create $|N_j| < O(3|E(G)|)$
⇒ time complexity to create $|N_j| < O(3m|V(G)|)$

Lemma 2. $O(\sum_{i=1}^{V(G)} \sum_{j=1}^{N_i} 3|N_i \cap N_j|) = O(mn|V(G)|)$

Note 20. $\sum_{i=1}^{V(G)} |N_i \cap N_j|$ is maximized, within assumptions of maximum degree
and $-E(G)$, by the graph $g$ where there are $x$ cliques of degree $n$.

⇒ $\forall v_i \in$ cliques of $g \sum_{i=1}^{V(G)} |N_i \cap N_j|$ is $O(3n^2)$
⇒ $\forall v_i \not\in$ cliques of $g$ is $O(0)$

Note 21. $|v_i \in$ cliques of $g| = xn$

⇒ $O(\sum_{i=1}^{V(G)} \sum_{j=1}^{N_i} 3|N_i \cap N_j|) = xn(3n^2) = \left(\frac{|E(G)|}{n^2}\right)n^3 = mn|V(G)|$

Lemma 3. $O(\sum_{i=1}^{V(G)} \sum_{j=1}^{N_i} 3|N_j \cap N_i|) = O(9/4mn|V(G)|)$

Note 22. a graph consisting of $x$ subgraphs containing only $E_2(G)$ triples maximizes $O(\sum_{i=1}^{V(G)} \sum_{j=1}^{N_i} 3|N_j \cap N_i|)$.

Definition 7. Let $H$ be the set of all subgraphs of $G$.
find $-E(h)$—
$\forall h \in H, \exists \left(\frac{V(h)}{3}\right)$ unordered triples
Note 23. $\exists \frac{|V(h)| - 2}{2}$ repetitions of an edge in unordered triples

Note 24. $\forall G \in h, \exists 2$ edges

$\Rightarrow |E(h)| = (2 \left( \frac{|V(h)|}{3} \right)) / \left( \frac{|V(h)| - 2}{2} \right)$

$\Rightarrow |E(h)| = \frac{2}{3}|V(h)|(|V(h)| - 1) \approx \frac{2}{3}|V(h)|^2$

$\Rightarrow$ given max degree $n$, $max(\frac{2}{3}|V(h)|^2) = n|V(h)|$

$\Rightarrow max(|V(h)|) = \frac{3}{2}n$

$x = \frac{|E(G)|}{|E(h)|}$

$\Rightarrow x = m|V(G)| / \frac{3}{2}n^2$

$\Rightarrow x = m|V(G)| / 2n^2$

$\Rightarrow max(|E_2(G)|) < x\left( \left( \frac{|V(h)|}{3} \right) \right)$

$\Rightarrow max(|E_2(G)|) < \frac{m|V(G)|}{3n^2/2} (\frac{3}{2})n^2$

$\Rightarrow max(|E_2(G)|) < 9/4mn|V(G)|$

6. Worst-Case Space Complexity Under Assumptions

Theorem 3. Worst case indexing is a small multiple of the total number of edges while the total space complexity excluding indeci is $O(n)$

Proof of Theorem. Assumption 5. maximum size of a hashmap is 8 times the number of entries (see Sun Java 1.6 specifications)

$\Rightarrow$ size of vertices hashtable is $O(8|V(G)|)$

$\Rightarrow$ size of edge hashtable is $O(16|V(G)| + 16|E(G)|)$

$\Rightarrow$ size of all indexing is $O(24|V(G)| + 16n|V(G)|)$

$\Rightarrow$ space complexity is $O(24|V(G)| + 16n|V(G)| + 6n)$

7. Conclusion

This algorithm enumerates all triple of a graph in linear time when the graph meets the assumptions of small average number of edges per vertices and small maximum degree.

8. Acknowledgements

This research has been funded by the Waikato Doctoral Scholarship.
References

