Computer Science, Texas A&M University

On Asymptotic Cost of Triangle Listing in Random Graphs

Di Xiao, Yi Cui, Daren B.H. Cline, Dmitri Loguinov

Internet Research Lab
Department of Computer Science and Engineering
Texas A&M University

May 16, 2017

Agenda

- Introduction
- Background
- Unifying framework
- Main results
- Evaluation

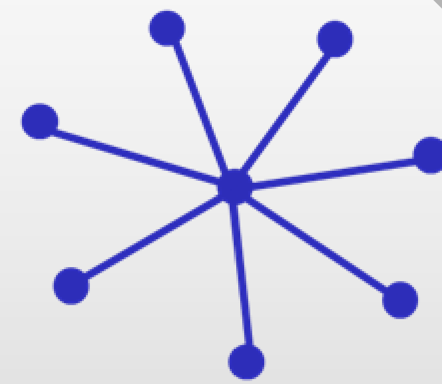
Introduction

- Triangle listing: given a simple undirected graph $G=(V,E)$, identify all 3-node cycles Δ_{xyz}
- Numerous applications
  - Network analysis: clustering coefficient, transitivity
  - Web/social networks: spam/community detection
  - Bioinformatics, graphics, databases, theory of computing
- Many open problems
  - Impact of degree distribution on CPU cost, deciding which neighbor traversal order is best, finding the optimal acyclic orientation for a given method, comparing different strategies under their optimal node permutations
  - We study these issues in random graphs

Agenda

- Introduction
- Background
- Unifying framework
- Main results
- Evaluation

Background



- Triangle listing visits each node and verifies edge existence between each pair of neighbors
  - A star graph with 40M nodes requires at least 800T checks
  - This is the CPU cost we are interested in studying
- Acyclic orientation: choose a direction along each edge such that the resulting graph has no cycles
  - Triangle listing now involves checks among only out-neighbors, only in-neighbors, or some combination thereof
  - Orienting edges towards the center and using only out-neighbors reduces verification cost to zero!

Background

- Given a graph with n nodes, each acyclic orientation can be viewed as some permutation θ_n
  - Shuffle the nodes and assign sequential labels $1, \ldots, n$
  - Direct edges from larger labels to smaller
  - List only triangles Δ_{xyz} such that $x < y < z$
- Suppose the node with a new ID i has out-degree $X_i(\theta_n)$, in-degree $Y_i(\theta_n)$, and total degree $d_i(\theta_n)$
- Then, the CPU cost of all known methods $\mathcal{M}$ is can be expressed by one formula

$$c_n(\mathcal{M}, \theta_n) = \frac{1}{n} \sum_{i=1}^{n} f(X_i(\theta_n), d_i(\theta_n))$$

  - where f is some non-linear function that depends on $\mathcal{M}$

Computer Science, Texas A&M University

Background

- Assuming m edges, prior work has shown there exist neighbor search orders where $c_n(\mathcal{M}, \theta_n)$ is $O(m^{1.5}/n)$
  - This bound is loose in sparse graphs and has seen no improvement in ~40 years
  - Still unclear how to select the best permutation and neighbor traversal pattern so as to minimize the runtime
  - Main obstacle: for a given graph G, finding θ_n that optimizes cost is likely an NP-hard problem
- Instead, we seek insight from random graphs
  - Suppose $F_n(x)$ is a CDF on integers that represents the degree distribution of the graph
  - Assume $F_n(x) \to F(x)$ as $n \to \infty$
  - Concerned with expected cost over all graph realizations

Background

- Berry 2015 obtained the limiting cost for a method we call T_1 under descending-degree permutation θ_D

$$\lim_{n\to\infty} E[c_n(T_1,\theta_D)|\mathbf{D}_n] = \frac{E[(Z_1^2 - Z_1)Z_2Z_3\mathbf{1}_{\min(Z_2,Z_3)>Z_1}]}{2E^2[D]}$$

  - where D_n is the random degree sequence and Z_1, Z_2, Z_3, D are iid with distribution $F(x)$
  - Given Pareto degree with $F(x) = 1–(1+x/\beta)^{\alpha}$, the limit is finite iff $\alpha > 4/3$

- Open issues: which permutations/methods are fundamentally better for a given $F(x)$, under what conditions, and does θ_n and neighbor search order change the asymptotics or just constants inside $O(.)$?

Agenda

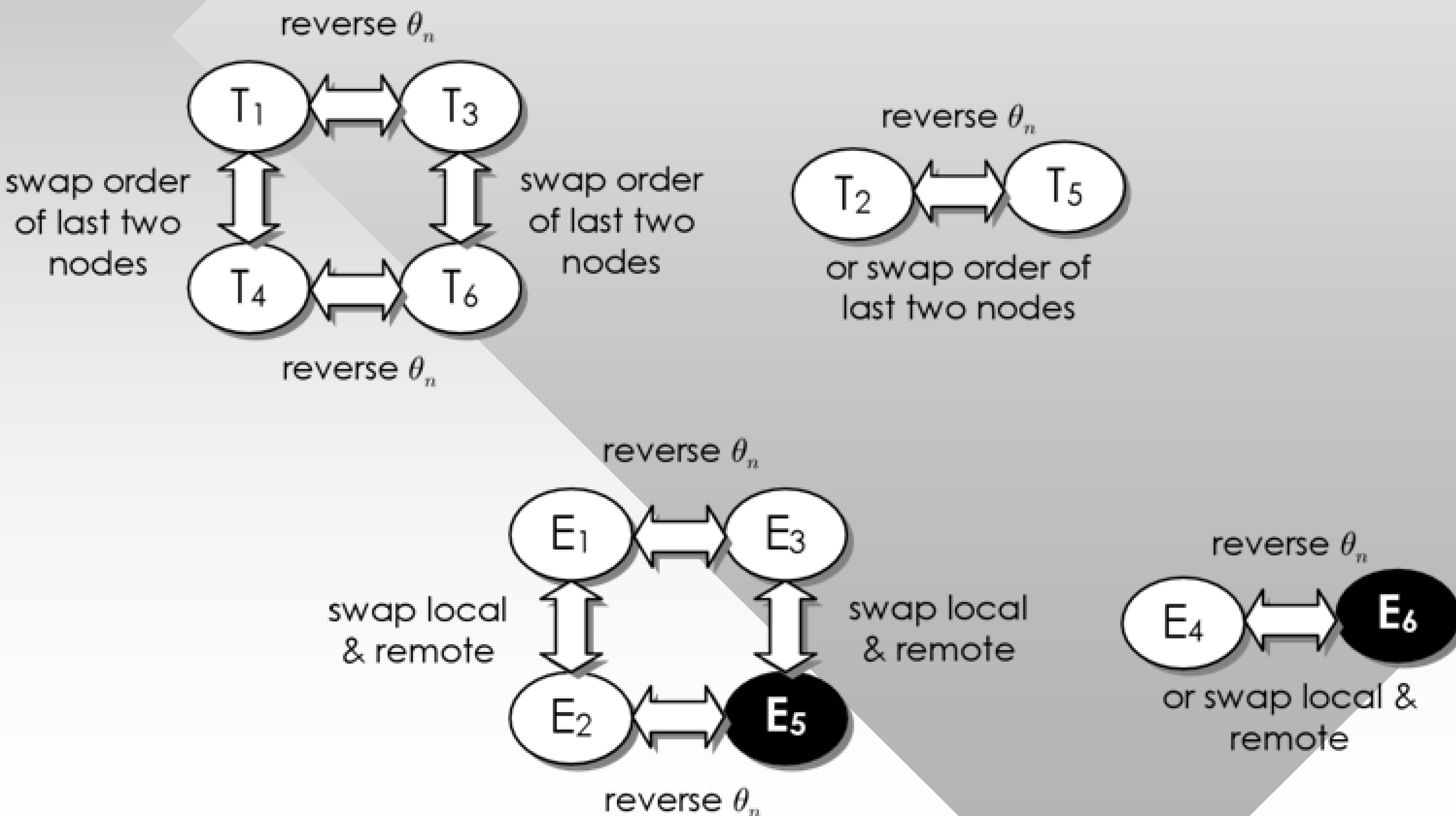
- Introduction
- Background
- Unifying framework
- Main results
- Evaluation

Unifying Framework

- We consider three families of algorithms and propose a generalized framework subsumes all previous efforts
  - Vertex Iterator (VI): methods T_1-T_6 that check neighbor pairs against a hash table
  - Scanning Edge Iterator (SEI): methods E_1-E_6 that run intersection of neighbor lists using sequential scans
  - Lookup Edge Iterator (LEI): methods L_1-L_6 that offers no CPU-cost benefits over VI, but have higher I/O
- It may seem that the order in which neighbors are visited (along in/out edges) is unimportant
  - However, this makes a noticeable difference!
  - Furthermore, improvement in cost is not limited to just constants, but asymptotics as well

Unifying Framework

- A total of 18 distinct methods, but many have identical cost; need to prune the result



Unifying Framework

- Four competing algorithms

- To minimize the runtime, need to consider the ratio of cost to speed

Family of algorithms	Operations	Speed
Vertex iterator	Hash table	19
Lookup edge iterator (LEI)	Hash table	19
Scanning edge iterator (SEI)	SIMD intersection	1,801

Table 3: Single-core speed (million nodes/sec) using an Intel i7-3930K @ 4.4 GHz.

- The speed can be easily benchmarked, what remains is to decide the optimal cost for each method
  - Note that E_1/E_2 have strictly more operations that T_1/T_2

Agenda

- Introduction
- Background
- Unifying framework
- Main results
- Evaluation

T_1	T_2	E_1	E_4
$\frac{x^2}{2}$	$x(1-x)$	$\frac{x(2-x)}{2}$	$\frac{x^2+(1-x)^2}{2}$

Main Results

- Many assumptions and details omitted (see the paper)
- Theorem: the cost of all 18 methods can be summarized by a sum of functions of order statistics

$$E[c_n(\mathcal{M},\theta_n)|\mathbf{D}_n] \approx \frac{1}{n}\sum_{i=1}^{n} g(d_i(\theta_n))h(q_i(\theta_n))$$

  - where $g(x) = x^2 - x$, $h(x)$ is given by the table above, and $q_i(\theta_n)$ depends only on the permuted degree sequence
- Since the degree $d_i(\theta_n)$ is sorted (e.g., $d_1(\theta_D)$ is the largest), this sum has some peculiar properties
  - Asymptotic behavior of averages in the above form is studied in a field of L-estimators

Main Results

- We leverage Glivenko-Cantelli results for functions of order statistics [Wellner 1978, van Zwet 1980]
- Theorem:

$$\lim_{n\to\infty} E[c_n(\mathcal{M},\theta_A)|\mathbf{D}_n] = E[g(D)h(J(D))]$$

$$\lim_{n\to\infty} E[c_n(\mathcal{M},\theta_D)|\mathbf{D}_n] = E[g(D)h(1-J(D))]$$

  – where $J(x)$ is the spread distribution of $F(x)$

- In particular,

$$\lim_{n\to\infty} E[c_n(T_1,\theta_D)|\mathbf{D}_n] = \frac{E[g(D)(1-J(D))^2]}{2}$$

$$\lim_{n\to\infty} E[c_n(E_1,\theta_D)|\mathbf{D}_n] = \frac{E[g(D)(1-J^2(D))]}{2}$$

Main Results

- However, non-monotonic permutations require a different approach and new theory
- Suppose θ_n converges to a random map $\xi(u)$
  - Random variable $\xi(u)$ specifies the new (permuted) location of nodes i that originate in the vicinity of $u = i/n$
- Theorem: the limiting cost under any convergent sequence of permutations is given by

$$\lim_{n\to\infty} E[c_n(\mathcal{M},\theta_n)|\mathbf{D}_n] = E[g(D)h(\xi(J(D)))]$$

- Both the model and derivations are much simpler than in prior work, even though we can handle a much wider class of methods/permutations

Main Results

- This allows us to establish optimal permutations for certain families of functions $h(x)$
- Theorem: T_1 and E_1 are both optimized by θ_D, T_2 by round-robin θ_{RR}, and E_4 by complementary RR θ_{CRR}
  - RR is a new permutation that places large degree towards the outside of the range $[1, n]$
  - CRR is another new permutation that does the opposite (large degree in the center)
- We can finally compare these methods under their respectively optimal θ_n
  - Theorem: $c_n(T_1, \theta_D) < c_n(T_2, \theta_{RR})$ for all $F(x)$
  - Theorem: $c_n(E_1, \theta_D) < c_n(E_4, \theta_{CRR})$ for all $F(x)$

Main Results

- When is VI better than SEI?
  - Cost of T_1 is finite iff $\alpha > 4/3$; that of E_1 iff $\alpha > 1.5$
  - Consequently, there are graphs where T_1 is always faster in the limit no matter what hardware is used
  - In real-world graphs, E_1 has 2-3x more cost, but 100x faster execution using SIMD intersection (see our ICDM 2016 paper)
- Derived limits are exact for all cases
  - Numerically accurate for small n in graphs with constrained degree; for large n, whenever the asymptotic cost is finite
  - Open issue: accurate models for small n, infinite limiting cost, and unconstrained degree
- In summary, both permutation and neighbor visit order change the asymptotics of cost!

Agenda

- Introduction
- Background
- Unifying framework
- Main results
- Evaluation

Evaluation: Constrained Graphs

n	$T_1 + \theta_A$			$T_1 + \theta_D$		
	sim	(50)	error	sim	(50)	error
10^4	159.1	155.6	−2.2%	40.2	39.3	−2.2%
10^5	518.0	516.6	−0.3%	87.8	87.0	−0.9%
10^6	1,355.6	1,354.5	−0.1%	143.7	142.9	−0.6%
10^7	3,089.1	3,089.2	0.003%	196.9	196.2	−0.4%
∞	∞			356.3		

Table 6: Cost with $\alpha = 1.5$ and root truncation.

n	$T_2 + \theta_D$			$T_2 + \theta_{RR}$		
	sim	(50)	error	sim	(50)	error
10^4	102.3	103.7	1.4%	79.5	75.8	−4.6%
10^5	260.0	261.4	0.5%	186.4	181.8	−2.5%
10^6	467.0	467.4	0.1%	315.4	310.4	−1.6%
10^7	674.6	675.4	0.1%	436.1	432.4	−0.8%
∞	1,307.6			770.4		

Table 7: Cost with $\alpha = 1.7$ and root truncation.

Computer Science, Texas A&M University

Evaluation: Unconstrained Graphs

n	$T_1 + \theta_A$			$T_1 + \theta_D$		
	sim	(50)	error	sim	(50)	error
10^4	7,158	6,452	−9.9%	209.5	241.1	15.1%
10^5	25,770	24,303	−5.7%	261.0	302.1	15.8%
10^6	84,441	82,815	−1.9%	294.1	333.0	13.3%
10^7	274,876	270,125	−1.7%	317.0	346.9	9.4%
∞		∞			356.3	

Table 9: Cost with $\alpha = 1.5$ and linear truncation.

n	$T_2 + \theta_D$			$T_2 + \theta_{RR}$		
	sim	(50)	error	sim	(50)	error
10^4	499.4	854.4	71.1%	354.5	532.6	50.3%
10^5	725.4	1,096.6	51.2%	476.5	662.3	39.0%
10^6	907.7	1,216.7	34.0%	570.2	724.4	27.0%
10^7	1,041.5	1,270.0	21.9%	631.2	751.5	19.1%
∞		1,307.6			770.4	

Table 10: Cost with $\alpha = 1.7$ and linear truncation.

Evaluation: Real Graphs

- Model predictions
  - Descending degree is optimal for T_1, E_1
  - RR is optimal for T_2, CRR for E_4
  - The best cost of E_1 is double that of T_2
- Degenerate permutation minimizes the largest out-degree
  - This improves T_1 by ~10%, but increases cost of the other methods 2-3x

	Permutation					
	θ_D	θ_A	θ_{RR}	θ_{CRR}	θ_U	θ_{degen}
T_1	150B	123T	63T	31T	45T	136B
T_2	360B	360B	255B	62T	41T	815B
E_1	511B	123T	63T	93T	86T	951B
E_4	123T	123T	123T	62T	82T	123T

Table 12: CPU operations on Twitter.

Thank you!

Questions?