Aegypti v0.4.3: Triangle Detection via Lookahead Clique-Constrained Union-Find and a Residual Matching Fallback

Frank Vega *

Posted Date: 25 May 2026

doi: 10.20944/preprints202511.2197.v5

Keywords: triangle-free graphs; triangle detection; union-find; matching; graph algorithms; computational complexity

 doi:10.20944/preprints202511.2197.v5

Article

Aegypti v0.4.3: Triangle Detection via Lookahead Clique-Constrained Union-Find and a Residual Matching Fallback

Frank Vega

Information Physics Institute, 840 W 67th St, Hialeah, FL 33012, USA; vega.frank@gmail.com

Abstract

We present AEGYPTI v0.4.3, a *hybrid heuristic* for detecting a single triangle in an undirected graph $G = (V, E)$ with $n = |V|$ vertices and $m = |E|$ edges. The algorithm has a fast Phase 1 based on a sparse clique-constrained Union-Find structure (FASTCLIQUEUF). It first unions a maximal matching M, then streams over the remaining edges. A rejected non-matching edge is not deleted immediately: it is kept alive for a fixed lookahead batch of size 32, allowing nearby residual edges to complete a triangle before the failed edge is hidden from future clique tests. Phase 1 also uses a bounded common-neighbour probe on promising residual edges. These changes preserve linear Phase 1 work, $\mathcal{O}(n + m)$, while increasing the probability of early detection on triangle-rich graphs. If Phase 1 fails, Phase 2 computes a greedy vertex cover C of the residual graph $G' = (V, E \setminus M)$ and performs adjacency-intersection enumeration only from that cover, while intersections still use the original graph so triangles touching a matching edge remain visible. Thus the fallback is not the full Chiba–Nishizeki scan over all vertices: its cost is $\mathcal{O}(r + \Phi_C)$, where $r = m - |M|$ and Φ_C is the sum of smaller-side intersection costs over residual base edges scanned from C. Since $\Phi_C \leq \mathcal{O}(r^{3/2})$, the coarse worst-case upper bound is $T(G) = \mathcal{O}(n + m + (m - |M|)^{3/2})$, while the implementation-specific bound is $T(G) = \mathcal{O}(n + m + \Phi_C)$. The worst-case exponent is still inherited from the classical Chiba–Nishizeki fallback; AEGYPTI is not claimed as a new asymptotic upper bound. Its contribution is practical: early exits on dense triangle-rich graphs, explicit witness triples, and a residual matching-edge and vertex-cover saving in the fallback. We evaluate v0.4.3 on an expanded 31-instance DIMACS benchmark suite. The smart algorithm wins 14 individual instances, including 7 of 8 deliberately triangle-rich adversarial instances, while sparse matrix multiplication remains fastest in total time because dense triangle-free bipartite graphs force the fallback.

Keywords: triangle-free graphs; triangle detection; union-find; matching; graph algorithms; computational complexity

MSC: 05C69; 68Q25; 68Q17; 68Q15

1. Introduction

Triangle detection—determining whether an undirected graph G contains three mutually adjacent vertices—is one of the most fundamental problems in graph algorithms. It arises as a primitive in network analysis [1], sparse-matrix computations [2], database join processing [3], and as the base case of exact clique solvers [4].

Known complexity.

Let $n = |V|$, $m = |E|$. The classical results are:

- $\mathcal{O}(m^{3/2})$ **via adjacency intersections.** Chiba and Nishizeki [5] showed that listing all triangles costs $\mathcal{O}(a(G) \cdot m)$, where $a(G)$ is the arboricity of G; since $a(G) = \mathcal{O}(\sqrt{m})$, detection costs $\mathcal{O}(m^{3/2})$.

 doi:10.20944/preprints202511.2197.v5

- $\mathcal{O}(n^\omega)$ **via matrix multiplication.** Itai and Rodeh [6] observed that triangle detection reduces to Boolean matrix multiplication; with the current best exponent $\omega < 2.372$ [7], this gives $\mathcal{O}(n^{2.372})$. Alon, Yuster, and Zwick [8] sharpened the combination to $\mathcal{O}(m^{2\omega/(\omega+1)})$.
- **Conditional lower bound.** Under the k-Clique Conjecture and related hypotheses, triangle detection requires $\Omega(n^{4/3})$ time [9] in the combinatorial model.

Our contribution.

We introduce AEGYPTI, a *hybrid heuristic* for triangle detection whose design is driven by the following observation:

> *If a graph is triangle-rich, an edge-streaming clique-constrained Union-Find with bounded lookahead can encounter a size-3 component early, before the classical fallback is needed.*

We emphasise the precise nature of the contribution:

- The *conceptual* novelty is the use of a clique-constrained Union-Find as a triangle-detection oracle; to our knowledge this specific data structure has not been used in this role before.
- The *worst-case* guarantee is inherited from the classical $\mathcal{O}(m^{3/2})$ adjacency-intersection fallback of Itai and Rodeh [6] and Chiba and Nishizeki [5]. We do *not* claim a new asymptotic upper bound on triangle detection.
- The *empirical* contribution is a constant-factor speedup on triangle-rich and clique-core inputs together with explicit witness triples at no extra cost.

When the graph is triangle-free or triangle-poor, the fast path certifies nothing useful and the fallback dominates the running time, so the algorithm should be understood as a hybrid heuristic with standard worst-case protection rather than a fundamentally new combinatorial algorithm.

The algorithm is named AEGYPTI after *Aedes aegypti*, a mosquito whose compound eye detects rapid visual patterns by aggregating local signals—analogous to the way our algorithm aggregates local adjacency information to detect the local pattern of a triangle.

Paper organisation.

Section 2 establishes notation and reviews the component data structures. Section 3 describes FASTCLIQUEUF and the full AEGYPTI algorithm. Section 4 proves correctness and analyses the complexity of each phase. Section 5 discusses related work in detail. Section 6 reports experimental results on DIMACS benchmarks. Section 7 concludes.

2. Preliminaries

2.1. Graph Notation

All graphs $G = (V, E)$ are simple and undirected. We write $n = |V|$, $m = |E|$, $N(v)$ for the open neighbourhood of v, and $N[v] = N(v) \cup \{v\}$ for the closed neighbourhood. The degree of v is $d(v) = |N(v)|$, and $\Delta = \max_v d(v)$.

Definition 1 (Triangle). *A* triangle *in G is a set $\{u, v, w\} \subseteq V$ of three distinct vertices such that $\{u, v\}, \{v, w\}, \{u, w\} \in E$.*

Definition 2 (Triangle-free). *G is* triangle-free *if it contains no triangle.*

2.2. Sparse Edge-Membership Representation

Version v0.4.3 uses a sparse implementation of FASTCLIQUEUF. Each component root stores the explicit list of vertices currently in the component. The underlying NetworkX graph supplies constant expected-time edge membership tests via `graph.has_edge`, and a local hash set stores non-matching edges that Phase 1 has deleted from the FASTCLIQUEUF view. Thus the data structure does not allocate

 doi:10.20944/preprints202511.2197.v5

dense n-bit closed-neighbourhood arrays; its initialisation is linear in the number of vertices, and all graph adjacency is reused from the input representation.

Proposition 1. *For the v0.4.3 sparse representation, initialising parent, size, component-membership, and deleted-edge tables costs $\mathcal{O}(n)$ time and space beyond the input graph. A live-edge query costs expected $\mathcal{O}(1)$ time.*

2.3. Union-Find

A Union-Find (disjoint-set) structure [10] maintains a partition of a ground set U under FIND (return root) and UNION (merge two sets). With path compression and union by size, both operations cost $\mathcal{O}(\alpha(n))$ amortised, where α is the inverse-Ackermann function [10].

3. The Aegypti Algorithm

3.1. FastCliqueUF: Sparse Clique-Constrained Union-Find

FASTCLIQUEUF augments a standard Union-Find with explicit component membership lists and a local set of deleted edges. A component is allowed to merge with another component only when the union remains a clique in the *live* graph: the original graph minus the non-matching edges that have been deleted inside the data structure.

Definition 3 (Clique invariant)**.** *At all times, for every root r, the component $C_r = \{v : \text{FIND}(v) = r\}$ is a clique in the live graph maintained by* FASTCLIQUEUF*.*

For two roots $r \neq s$, because C_r and C_s are already cliques by invariant, the union $C_r \cup C_s$ is a clique if and only if every cross pair $(a, b) \in C_r \times C_s$ is a live edge. In v0.4.3 this is tested directly by hash lookup in the original graph and in the deleted-edge set.

Lemma 1 (Triangle witness)**.** *If* FASTCLIQUEUF.UNION(u, v) *causes a component of size at least* 3 *to form, that component contains a triangle of the original graph.*

Proof. By Definition 3, the component is a clique in the live graph. Deleted edges are only removed from the live graph; no non-edge of the original graph is ever inserted. Hence any three vertices of a live clique of size at least 3 are mutually adjacent in the original graph and form a triangle. □

3.2. Aegypti v0.4.3

AEGYPTI v0.4.3 is a two-phase hybrid. Phase 1 first computes a maximal matching M using `nx.approximation.min_maximal_matching`, unions all edges of M, and then streams over $E \setminus M$. Before trying a residual edge, it spends a fixed linear budget on bounded common-neighbour probes. If a residual edge is rejected by FASTCLIQUEUF, it is not deleted immediately: it remains live for a fixed lookahead batch of 32 rejected edges. This lookahead gives nearby residual edges an opportunity to complete a triangle before a failed edge is hidden from later clique tests.

If Phase 1 fails, Phase 2 builds a greedy 2-approximation vertex cover of the residual graph $G' = (V, E \setminus M)$ and scans only edges incident to this cover, skipping matching edges as base edges. The intersections themselves are taken in the original graph, so a triangle that touches a matching edge remains visible.

 doi:10.20944/preprints202511.2197.v5

Algorithm 1 FASTCLIQUEUF core operations in v0.4.3.

procedure INITIALIZE($G = (V, E)$)
  for each $u \in V$ **do**
    parent[u] $\leftarrow u$; size[u] $\leftarrow 1$; members[u] $\leftarrow [u]$
  end for
  deleted $\leftarrow \emptyset$
end procedure
function LIVEEDGE(u, v)
  return $\{u, v\} \in E$ and $\{u, v\} \notin$ deleted
end function
function UNION(u, v)
  $r_u \leftarrow$ FIND(u); $r_v \leftarrow$ FIND(v)
  if $r_u = r_v$ **then**
    return size[r_u]≥ 3
  end if
  if size[r_u] $<$ size[r_v] **then**
    swap r_u, r_v
  end if
  for each $a \in$ members[r_u] **do**
    for each $b \in$ members[r_v] **do**
      if not LIVEEDGE(a, b) **then**
        return FALSE
      end if
    end for
  end for
  parent[r_v] $\leftarrow r_u$; size[r_u] $\leftarrow$ size[r_u] + size[r_v]
  members[r_u] $\leftarrow$ members[r_u] concatenated with members[r_v]
  delete members[r_v]
  return size[r_u]≥ 3
end function
procedure DELETEEDGE(u, v)
  deleted $\leftarrow$ deleted $\cup\{\{u, v\}\}$
end procedure

 doi:10.20944/preprints202511.2197.v5

Algorithm 2 AEGYPTI v0.4.3: lookahead matching-first triangle detection.

Require: Simple undirected graph $G = (V, E)$, no self-loops
Ensure: A triangle $\{u, v, w\}$ if one exists, else NONE

$B \leftarrow 32$ ▷ fixed delete-lookahead batch size
$D \leftarrow$ FASTCLIQUEUF.INITIALIZE(G)
$d(v) \leftarrow |N(v)|$ for every $v \in V$
$A \leftarrow$ empty cache of adjacency sets
$M \leftarrow$ MINMAXMATCH(G)
for each $(u, v) \in M$ **do**
    if D.UNION(u, v) **then**
        return any three vertices from a component of size ≥ 3
    end if
end for
rejected $\leftarrow [\,]$; probeBudget $\leftarrow \max(16, n)$
for each $(u, v) \in E \setminus M$ **do**
    if probeBudget > 0 and $\min(d(u), d(v)) \geq 2$ **then**
        probeBudget $\leftarrow$ probeBudget -1
        if a bounded common-neighbour probe of (u, v) finds w **then**
            return $\{u, v, w\}$
        end if
    end if
    if D.UNION(u, v) **then**
        return any three vertices from a component of size ≥ 3
    end if
    append (u, v) to rejected
    if $|\text{rejected}| = B$ **then**
        for each (x, y) in rejected **do**
            D.DELETEEDGE(x, y)
        end for
        rejected $\leftarrow [\,]$
    end if
end for
for each (x, y) in rejected **do**
    D.DELETEEDGE(x, y)
end for
▷ Phase 2: residual vertex-cover fallback
$C \leftarrow \emptyset$; covered $\leftarrow \emptyset$
for each $(u, v) \in E \setminus M$ **do**
    if $u \notin$ covered and $v \notin$ covered **then**
        append u, v to C; covered $\leftarrow$ covered $\cup \{u, v\}$
    end if
end for
$\rho \leftarrow$ order induced by C
for each $u \in C$ **do**
    for each $v \in N(u)$ with $(u, v) \notin M$ and not previously scanned **do**
        for each w in the smaller of $N(u)$ and $N(v)$ **do**
            if w belongs to the other neighbourhood **then**
                return $\{u, v, w\}$
            end if
        end for
    end for
end for
return NONE

 doi:10.20944/preprints202511.2197.v5

4. Correctness and Complexity Analysis

4.1. Correctness

Theorem 1 (Correctness of Aegypti v0.4.3)**.** AEGYPTI(G) *returns a triangle if and only if G contains a triangle.*

Proof. Soundness. Phase 1 returns either a bounded-probe witness $w \in N(u) \cap N(v)$ for an edge (u, v), or a component of size at least 3 produced by FASTCLIQUEUF. The first case is immediately a triangle, and the second is valid by Lemma 1. Phase 2 returns only after finding $w \in N(u) \cap N(v)$ for an actual edge (u, v), again giving a triangle.

Completeness. If Phase 1 finds no triangle, Phase 2 remains. Let M be the matching used by Phase 1. By Lemma 2, no edge of M can belong to a triangle after Phase 1 terminates without a witness. Thus every remaining triangle has at least one base edge in $E \setminus M$. The greedy set C is a vertex cover of $G' = (V, E \setminus M)$, so that base edge is incident to some vertex in C and is scanned by Phase 2. Since intersections are taken in the original graph, the third edge may be a matching edge and still be detected. Therefore any triangle missed by Phase 1 is found by Phase 2. □

Lemma 2 (Matching edges are useless for Phase 2)**.** *Let M be the matching used in Phase 1 and assume Phase 1 terminated without finding a triangle. Then no edge of M is contained in any triangle of G.*

Proof. Suppose $(u, v) \in M$ and $\{u, v, w\}$ is a triangle. Since M is a matching, (u, w) and (v, w) are not in M. After the matching pass, u and v lie in a 2-clique component and w is not in that component. When the first residual edge from w to this component is processed, the cross-edge test succeeds because the other residual edge exists. Hence Phase 1 would create a size-3 clique and return a triangle, contradicting the assumption. □

4.2. Running Time

Lemma 3 (Phase 1 cost)**.** *Phase 1 of v0.4.3 runs in $\mathcal{O}(n + m)$ expected time.*

Proof. Initialising the sparse FASTCLIQUEUF tables costs $\mathcal{O}(n)$. The matching routine and the scan over E are linear in the graph representation. The common-neighbour probe has fixed candidate limit 8 and total budget $\max(16, n)$, hence costs $\mathcal{O}(n)$ plus adjacency-cache construction for probed vertices. Each rejected edge is appended once and deleted at most once; the lookahead batch size is the fixed constant 32. Before a successful size-3 merge, the components tested by UNION remain bounded in size in the intended fast path, and every failed residual edge incurs only a constant number of live-edge checks. The expected hash-table costs are constant, so the phase is $\mathcal{O}(n + m)$. □

Theorem 2 (Phase 2 cost)**.** *Let $r = m - |M|$, where M is the matching computed in Phase 1. Let C be the greedy vertex cover computed on the residual graph $G' = (V, E \setminus M)$, and let E_C be the residual base edges actually scanned by Phase 2, oriented so that each edge is scanned once from an endpoint in C. Define*

$$\Phi_C = \sum_{(u,v)\in E_C} \min\{d_G(u), d_G(v)\}.$$

Then Phase 2 runs in $\mathcal{O}(r + \Phi_C)$ time. In particular, $\Phi_C = \mathcal{O}(r^{3/2})$, so the coarse worst-case bound is $\mathcal{O}(r^{3/2})$.

Proof. The residual graph $G' = (V, E \setminus M)$ has r edges. The greedy cover construction is linear in r. Unlike the full Chiba–Nishizeki scan, the fallback does not iterate over all vertices of G; it scans residual base edges only from the cover C. Since C is a vertex cover of G', every residual base edge has at least one endpoint in C and is scanned once by the rank rule. For each scanned edge (u, v), the smaller-side intersection costs $\mathcal{O}(\min\{d_G(u), d_G(v)\})$, giving the exact cover-restricted term Φ_C. The original graph degrees are used because intersections must still see triangles whose third side is a

 doi:10.20944/preprints202511.2197.v5

matching edge. Finally, $E_C \subseteq E \setminus M$, so the standard Chiba–Nishizeki smaller-side summation over the residual edge set gives $\Phi_C = \mathcal{O}(r^{3/2})$ as a worst-case upper bound. □

Theorem 3 (Total running time)**.** *Aegypti v0.4.3 runs in*

$$\mathcal{O}(n + m + \Phi_C)$$

time, where Φ_C is the cover-restricted Phase 2 intersection cost from Theorem 2. Consequently its coarse worst-case bound is $\mathcal{O}(n + m + (m - |M|)^{3/2})$. If Phase 1 detects a triangle after streaming k^* residual edges, the cost is $\mathcal{O}(n + k^*)$ expected time.

Proof. Combine Lemma 3 and Theorem 2. If Phase 1 halts early, Phase 2 is skipped and only the matching prefix, bounded probes, and the first k^* streamed residual edges are paid. □

4.3. Space Complexity

Proposition 2. *Aegypti v0.4.3 uses $\mathcal{O}(n + m)$ words of working space beyond the input graph representation.*

Proof. The sparse FastCliqueUF stores parent, size, member lists, and a deleted-edge set; these are $\mathcal{O}(n)$ plus at most $\mathcal{O}(m)$ deleted edges. Phase 2 builds cached adjacency sets and a residual vertex cover, bounded by $\mathcal{O}(n + m)$ words. □

5. Related Work

Combinatorial triangle detection.

Itai and Rodeh [6] gave the first linear-space algorithm running in $\mathcal{O}(m^{3/2})$, which remains the best combinatorial bound. Chiba and Nishizeki [5] gave an output-sensitive analysis via arboricity. Latapy [11] provided an engineering study with practical optimisations; Ortmann and Brandes [12] refined the approach for main-memory systems.

Algebraic algorithms.

Alon, Yuster, and Zwick [8] obtained $\mathcal{O}(m^{2\omega/(\omega+1)})$ by a clever combination of matrix multiplication and combinatorial techniques. For dense graphs, $\mathcal{O}(n^\omega)$ dominates [7]. These algorithms are impractical for large sparse graphs due to large constants and cache effects.

Bit-parallel and sparse engineering methods.

Vassilevska Williams [13] studied bit-parallel matrix multiplication for triangle counting. Cohen [14] used bitset-based adjacency representations for fast triangle listing in social networks. Version v0.4.3 instead uses a sparse component-membership representation for FastCliqueUF, relying on hash-based edge membership and delayed deletion to reduce setup cost and preserve a linear Phase 1 scan.

Union-Find approaches.

To our knowledge, using a clique-constrained Union-Find as a triangle-detection oracle is novel as a *data-structural* idea. The closest prior work is the use of Union-Find in minimum spanning tree algorithms [10] and in dynamic connectivity, where Union-Find maintains spanning trees under edge insertions. We emphasise that this conceptual novelty is distinct from any claim of asymptotic improvement: the *worst-case* behaviour of Aegypti is governed by a residual, cover-restricted Chiba–Nishizeki-style fallback. Its exact cost is $\mathcal{O}(r + \Phi_C)$ with $r = m - |M|$, and it is upper bounded by $\mathcal{O}(r^{3/2})$. We therefore position Aegypti as a hybrid heuristic with classical worst-case protection, not as a fundamentally new asymptotic algorithm.

 doi:10.20944/preprints202511.2197.v5

Streaming and dynamic models.

For streaming triangle detection (where edges arrive in order and space is sublinear), Buriol et al. [15] gave sampling-based approximation algorithms. AEGYPTI naturally supports a streaming model in its Phase 1: edges are consumed one by one and the algorithm halts as soon as a triangle is confirmed.

6. Experiments

6.1. Setup

All experiments were run on an 11th Gen Intel® Core™ i7-1165G7 (2.80 GHz), 32 GB DDR4 RAM, with Python 3.12, Aegypti v0.4.3 [16], NetworkX 3.4, SciPy 1.15, and NumPy 2.2 [17]. The benchmark suite consists of 31 DIMACS-format graphs stored under `finlay/experiments/`. The first 15 instances are correctness and design cases from earlier versions. Instances `exp16`–`exp23` add dense triangle-free and planted-triangle adversaries. Instances `exp24`–`exp31` are deliberately smart-favourable triangle-rich cases: complete or multipartite dense graphs, dense clique cores with distractors, and noisy layered clique constructions. Reported times are medians of 20 repeated runs in milliseconds.

6.2. Baselines

We compare three algorithms:

- **Aegypti v0.4.3 (smart).** The lookahead matching-first FASTCLIQUEUF algorithm with the residual vertex-cover fallback described in Algorithm 2.
- **Chiba–Nishizeki (CN).** A standalone implementation of the classical adjacency-intersection algorithm with worst-case $\mathcal{O}(m^{3/2})$ behaviour.
- **Matrix baseline.** A sparse matrix-cubing check using the diagonal of A^3; this reports whether a triangle exists but does not return a witness triple.

6.3. Expanded DIMACS Benchmark Suite

Consequences of the benchmark.

The expanded suite separates the two operating regimes of v0.4.3. On dense or structured triangle-rich graphs, Phase 1 often returns before Phase 2 is reached; this is why Aegypti wins on `exp10`, `exp20`, `exp21`, `exp23`, and on seven of the eight smart-favourable instances `exp24`–`exp31`. These cases contain many local triangle witnesses near high-degree or clique-like regions, so the bounded probe and delete-lookahead mechanism frequently trigger early.

The main failure mode is also visible. Dense triangle-free bipartite graphs and crown graphs contain many high-degree edges but no triangle. Aegypti must pay Phase 1 and then the residual fallback before proving absence. The sparse matrix baseline is especially strong on these instances because its compiled sparse kernels amortise the proof of triangle-freeness better than Python-level set intersections. Thus v0.4.3 is most attractive when the input distribution is expected to be triangle-rich, clustered, or clique-core dominated, and least attractive when graphs are dense, bipartite-like, and triangle-free.

Aggregate outcome.

Across all 31 experiments, Aegypti has the most individual wins after the matrix baseline: Aegypti wins 14 instances, the matrix baseline wins 13, and Chiba–Nishizeki wins 4. In total median time, however, the matrix baseline remains fastest (62.502 ms), followed by Aegypti (96.842 ms) and Chiba–Nishizeki (149.975 ms). On the deliberately smart-favourable block `exp24`–`exp31`, Aegypti wins 7/8 and has the best subtotal (24.828 ms versus 45.161 ms for matrix and 62.337 ms for Chiba–Nishizeki). These results support the intended use case: v0.4.3 is a fast witness-finding heuristic with classical fallback protection, not a universal replacement for algebraic triangle detection.

 doi:10.20944/preprints202511.2197.v5

Table 1. Aegypti v0.4.3 on the expanded `finlay/experiments/` DIMACS suite. Times are median milliseconds over 20 runs. "T" indicates whether the graph contains a triangle. The winner is the fastest median among Aegypti, Chiba–Nishizeki, and the sparse matrix baseline.

Instance	n	m	T	Aegypti	CN	Matrix	Winner
exp01_k3	3	3	Y	0.029	0.017	0.115	CN
exp02_k10	10	45	Y	0.044	0.050	0.101	Aegypti
exp03_k30	30	435	Y	0.115	0.293	0.179	Aegypti
exp04_petersen	10	15	N	0.077	0.036	0.079	CN
exp05_c50	50	50	N	0.243	0.129	0.099	Matrix
exp06_k20_20	40	400	N	1.454	0.976	0.196	Matrix
exp07_decoy1	6	6	Y	0.027	0.020	0.092	CN
exp08_decoy10	60	60	Y	0.095	0.119	0.098	Aegypti
exp09_sparse_with_tri	145	210	Y	0.838	0.597	0.240	Matrix
exp10_dense_rand	200	6076	Y	1.425	7.469	4.772	Aegypti
exp11_bipartite_tf	80	252	N	0.899	0.464	0.192	Matrix
exp12_med_near_tf	287	425	Y	1.463	1.032	0.267	Matrix
exp13_cycle_chord	8	9	Y	0.030	0.022	0.085	CN
exp14_er_tf	34	36	N	0.178	0.096	0.090	Matrix
exp15_decoy30x2	270	270	Y	0.297	0.493	0.111	Matrix
exp16_k80_80	160	6400	N	33.406	39.560	2.838	Matrix
exp17_c300	300	300	N	1.533	0.887	0.165	Matrix
exp18_crown60	120	3540	N	15.543	16.328	1.447	Matrix
exp19_bipartite_dense_det	180	2833	N	11.964	10.132	1.128	Matrix
exp20_k80	80	3160	Y	0.770	4.569	1.947	Aegypti
exp21_k30_30_30	90	2700	Y	0.595	1.940	1.185	Aegypti
exp22_large_cycle_planted_tri	240	242	Y	0.345	0.435	0.104	Matrix
exp23_sparse_det_planted_tri	198	2130	Y	0.644	1.975	1.810	Aegypti
exp24_k120	120	7140	Y	1.188	6.312	4.147	Aegypti
exp25_k60_60_60	180	10800	Y	2.434	10.958	8.846	Aegypti
exp26_turan4_50	200	15000	Y	3.047	15.714	13.178	Aegypti
exp27_k80_leaf_cloud	260	3340	Y	0.679	2.652	2.820	Aegypti
exp28_four_k60_bridge	240	7084	Y	1.324	4.727	2.522	Aegypti
exp29_bipartite_plus_k20	180	6630	Y	13.281	8.874	3.367	Matrix
exp30_dense_det_k12_seed	220	6637	Y	1.429	7.230	5.220	Aegypti
exp31_three_k70_noisy_layers	210	7791	Y	1.446	5.870	5.062	Aegypti
All instances	–	–	–	96.842	149.975	62.502	Matrix
exp24–exp31 subtotal	–	–	–	24.828	62.337	45.161	Aegypti

6.4. Summary and Limitations

Aegypti v0.4.3 is correct on every benchmark instance tested and returns explicit triangle witnesses, unlike the matrix baseline. Its fallback cost is more precise than the simple $\mathcal{O}((m-|M|)^{3/2})$ expression: Phase 2 scans only from a greedy vertex cover C of the residual graph, so the implemented cost is $\mathcal{O}(r+\Phi_C)$, where $r = m-|M|$ and Φ_C sums smaller-side intersections only over residual base edges reached from C. Dense triangle-free cases still make this term large, which is why they remain adversarial. The empirical lesson is that Aegypti should be used when one expects triangles to be abundant or clustered enough for Phase 1 to terminate early; for dense triangle-free certification, sparse matrix multiplication or a low-level Chiba–Nishizeki implementation remains preferable.

7. Conclusions

We introduced AEGYPTI v0.4.3, a hybrid triangle-detection algorithm that combines a sparse clique-constrained Union-Find (Phase 1 with matching-first edge ordering, bounded common-neighbour probes, and fixed-size delete lookahead) with a residual vertex-cover adjacency-intersection fallback. The algorithm is correct, runs in expected $\mathcal{O}(n+k^*)$ time when Phase 1 detects a triangle after streaming k^* residual edges, and otherwise runs in $\mathcal{O}(n+m+\Phi_C)$ time, with $\Phi_C \le \mathcal{O}((m-|M|)^{3/2})$ as a coarse worst-case upper bound. Experiments on the expanded 31-instance DIMACS suite show that v0.4.3 is strongest on dense triangle-rich, multipartite, and clique-core inputs, while dense triangle-free bipartite graphs remain its principal adversarial regime.

Open problems.

- **Sharper residual bounds.** Can the cover-restricted quantity Φ_C be bounded more tightly than the generic $\mathcal{O}((m-|M|)^{3/2})$ upper bound on clustered graph families?
- **Edge ordering.** Beyond the matching-first ordering used here, does a degeneracy-based or degree-based ordering further reduce the average k^* on random graphs?

 doi:10.20944/preprints202511.2197.v5

- **Triangle counting.** Can FastCliqueUF be extended to count all triangles, not just detect one?
- **Dynamic graphs.** Extending the structure to support edge insertions and deletions on the underlying graph while maintaining the clique invariant remains an open problem.

References

1. Watts, D.J.; Strogatz, S.H. Collective Dynamics of 'Small-World' Networks. *Nature* **1998**, *393*, 440–442. https://doi.org/10.1038/30918.
2. Buluç, A.; Gilbert, J.R. The Combinatorial BLAS: Design, Implementation, and Applications. 2011, Vol. 25, pp. 496–509. https://doi.org/10.1177/1094342011403516.
3. Ngo, H.Q.; Porat, E.; Ré, C.; Rudra, A. Worst-Case Optimal Join Algorithms. In Proceedings of the Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS 2012). ACM, 2012, pp. 37–48. https://doi.org/10.1145/2213556.2213565.
4. Bron, C.; Kerbosch, J. Algorithm 457: Finding All Cliques of an Undirected Graph. *Communications of the ACM* **1973**, *16*, 575–577. https://doi.org/10.1145/362342.362367.
5. Chiba, N.; Nishizeki, T. Arboricity and Subgraph Listing Algorithms. *SIAM Journal on Computing* **1985**, *14*, 210–223. https://doi.org/10.1137/0214017.
6. Itai, A.; Rodeh, M. Finding a Minimum Circuit in a Graph. *SIAM Journal on Computing* **1978**, *7*, 413–423. https://doi.org/10.1137/0207033.
7. Williams, V.V.; Xu, Y.; Xu, Z.; Zhou, R. New Bounds for Matrix Multiplication: from Alpha to Omega **2024**. Preprint available at https://arxiv.org/abs/2307.07970.
8. Alon, N.; Yuster, R.; Zwick, U. Finding and counting given length cycles. *Algorithmica* **1997**, *17*, 209–223. https://doi.org/10.1007/BF02523189.
9. Abboud, A.; Williams, V.V. Popular Conjectures Imply Strong Lower Bounds for Dynamic Problems. In Proceedings of the Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2014), 2014, pp. 434–443. https://doi.org/10.1109/FOCS.2014.53.
10. Tarjan, R.E. Efficiency of a Good but Not Linear Set Union Algorithm. *Journal of the ACM* **1975**, *22*, 215–225. https://doi.org/10.1145/321879.321884.
11. Latapy, M. Main-memory Triangle Computations for Very Large (Sparse (Power-Law)) Graphs. *Theoretical Computer Science* **2008**, *407*, 458–473. https://doi.org/10.1016/j.tcs.2008.07.017.
12. Ortmann, M.; Brandes, U. Triangle Listing Algorithms: Back from the Diversion. In Proceedings of the Proceedings of the 16th Workshop on Algorithm Engineering and Experiments (ALENEX 2014). SIAM, 2014, pp. 1–8. https://doi.org/10.1137/1.9781611973198.1.
13. Williams, V.V.; Williams, R. Subcubic Equivalences Between Path, Matrix and Triangle Problems **2010**. pp. 645–654. https://doi.org/10.1109/FOCS.2010.67.
14. Cohen, J. Graph Twiddling in a MapReduce World. *Computing in Science & Engineering* **2009**, *11*, 29–41. https://doi.org/10.1109/MCSE.2009.120.
15. Buriol, L.S.; Frahling, G.; Leonardi, S.; Marchetti-Spaccamela, A.; Sohler, C. Counting Triangles and the Curse of the Last Reducer. In Proceedings of the Proceedings of the 15th International World Wide Web Conference (WWW 2006). ACM, 2006, pp. 1345–1354. https://doi.org/10.1145/1963405.1963491.
16. Vega, F. Aegypti 0.4.3: Triangle-Free Solver. Software package, https://pypi.org/project/aegypti/, 2026.
17. Harris, C.R.; Millman, K.J.; van der Walt, S.J.; Gommers, R.; Virtanen, P.; Cournapeau, D.; et al. Array Programming with NumPy. *Nature* **2020**, *585*, 357–362. https://doi.org/10.1038/s41586-020-2649-2.