A subquadratic triad census algorithm for large sparse networks with small maximum degree

Vladimir Batagelj and Andrej Mrvar
University of Ljubljana

Abstract

In the paper a subquadratic ($O(m)$, m is the number of arcs) triad census algorithm for large and sparse networks with small maximum degree is presented. The algorithm is implemented in the program **Pajek**.

Keywords: large networks, triad census, algorithm

Math. Subj. Class. (2000): 91D30, 93A15, 68R10, 05C85, 68W40, 05C90

1 Introduction

James Moody (1998) proposed a quadratic ($O(n^2)$, n is the number of vertices) algorithm for determining the triad census of a network. For (very) large and sparse networks (tens or hundreds of thousands of vertices) a subquadratic algorithm is needed (Batagelj, Mrvar 1998). In this paper we present such an $O(m)$, m is the number of arcs, triad census algorithm.

Let $G = (V, R)$ be a directed network (relational graph); V is the set of vertices and $R \subseteq V \times V$ is the set of arcs. We assume that G has no loops – the relation R is irreflexive. We denote by R' the inverse relation, $xR'y \Leftrightarrow yRx$, and by $\hat{R} = R \cup R'$ the symmetrized relation. $\hat{R}(v)$ is the set of all neighbors of vertex v, $\hat{R}(v) = \{u : v\hat{R}u\}$, and $\hat{d}(v)$ its cardinality $\hat{d}(v) = |\hat{R}(v)|$. Note that $\sum_{v \in V} \hat{d}(v) \leq 2m$. We also define the *maximum degree* in network G

$$\hat{\Delta} = \max_{v \in V} \hat{d}(v)$$

Most large networks are sparse – the number of lines (edges or arcs) m is subquadratic: $m = O(k(n) \cdot n)$, where $k(n)$ is small with respect to n, $k(n) << n$ (for example $k(n) = \text{const}$, or $k(n) = \log n$, or $k(n) = \sqrt{n}$). For some interesting ideas about the structure of large networks see Barabasi et al. (1999).

Graph G determines a function

$$Link(u, v) = \begin{cases} 1 & uRv \\ 0 & \text{otherwise} \end{cases}$$

that we shall use in our algorithm.

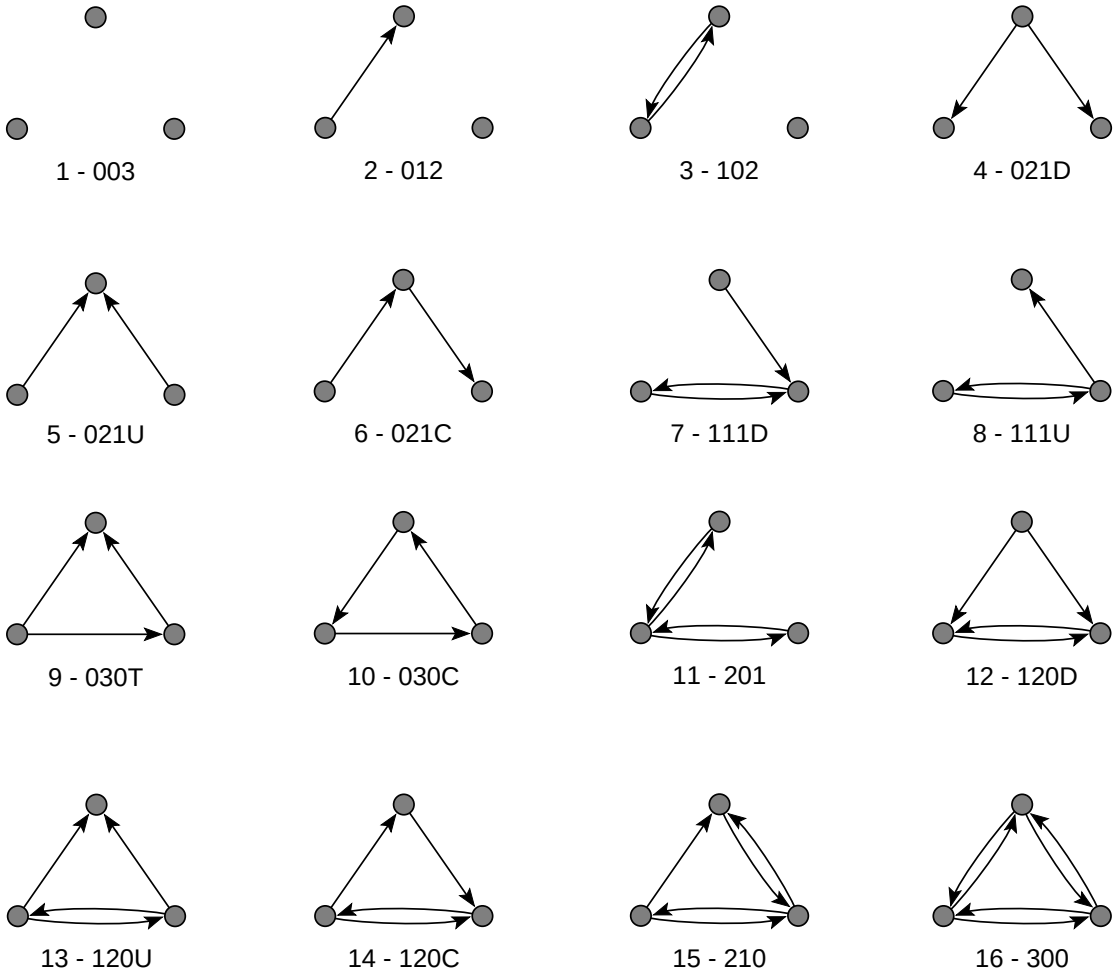

Figure 1: Types of Triads.

2 The algorithm

2.1 Basic idea

All possible triads (Wasserman, Faust 1994, page 244) can be partitioned into three basic types (see Figure 1):

- the *null* triad 003;
- *dyadic* triads 012 and 102; and
- *connected* triads: 111D, 201, 210, 300, 021D, 111U, 120D, 021U, 030T, 120U, 021C, 030C and 120C.

In a large and sparse network most triads are null triads. Since the total number of triads is $T = \binom{n}{3}$ and the above types partition the set of all triads, the idea of the algorithm is as follows:

- count all dyadic T_2 and all connected T_3 triads with their subtypes;
- compute the number of null triads $T_1 = T - T_2 - T_3$.

In the algorithm we have to assure that every non-null triad is counted exactly once. A set of three vertices $\{v, u, w\}$ can be in general selected in 6 different ways (v, u, w), (v, w, u), (u, v, w), (u, w, v), (w, v, u), (w, u, v). We solve the problem by introducing the ***canonical*** selection that contributes to the triadic count; the other, noncanonical selections need not to be considered in the counting process.

In the following, to simplify the presentation, we shall assume that the vertices are represented by integers $V = \{1, 2, 3, \ldots, n\}$.

2.2 Counting dyadic triads

Every connected dyad forms a dyadic triad with every vertex both members of the dyad are not adjacent to.

Each pair of vertices (v, u), $v < u$ connected by an arc contributes

$$n - |\hat{R}(u) \cup \hat{R}(v) \setminus \{u, v\}| - 2$$

triads of type 3 – 102, if u and v are connected in both directions; and of type 2 – 012 otherwise. The condition $v < u$ determines the canonical selection for dyadic triads.

2.3 Counting connected triads

A selection (v, u, w) of connected triad is canonical iff $v < u < w$. The triads isomorphism problem can be efficiently solved by assigning to each triad a code $Tricode(v, u, w)$ – an integer number between 0 to 63, defined by

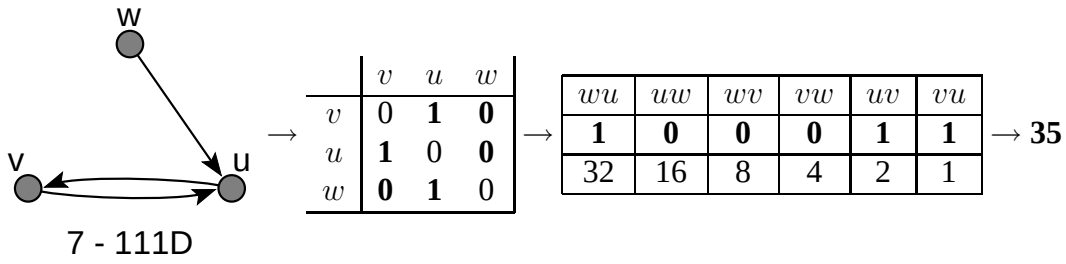

Figure 2: Triad Code.

Table 1: Triad Types.

code	type	code	type	code	type	code	type
0	1	16	2	32	2	48	3
1	2	17	6	33	5	49	7
2	2	18	4	34	6	50	8
3	3	19	8	35	7	51	11
4	2	20	5	36	6	52	7
5	4	21	9	37	9	53	12
6	6	22	9	38	10	54	14
7	8	23	13	39	14	55	15
8	2	24	6	40	4	56	8
9	6	25	10	41	9	57	14
10	5	26	9	42	9	58	13
11	7	27	14	43	12	59	15
12	3	28	7	44	8	60	11
13	8	29	14	45	13	61	15
14	7	30	12	46	14	62	15
15	11	31	15	47	15	63	16

function $Tricode(v, u, w : vertex) : integer$;
begin

$$Tricode := Link(v,u) + 2 * (Link(u,v) + 2 * (Link(v,w) + 2 * (Link(w,v) + 2 * (Link(u,w) + 2 * Link(w,u)))));$$

end;

The function $Tricode$ essentially treats the out-diagonal entries of triad adjacency matrix as a binary number (see Figure 2).

Each triad code corresponds to a unique triad type as described by Table 1.

2.4 The triad census algorithm

INPUT: relational graph $G = (V, R)$ represented by lists of neighbors
OUTPUT: table $Census$ with frequencies of triadic types

```
1       for i := 1 to 16 do Census[i] := 0;
2       for each v ∈ V do
2.1         for each u ∈ R̂(v) do if v < u then begin
2.1.1           S := R̂(u) ∪ R̂(v) \ {u, v};
2.1.2           if vRu ∧ uRv then TriType := 3 else TriType := 2;
2.1.3           Census[TriType] := Census[TriType] + n − |S| − 2;
2.1.4           for each w ∈ S do if u < w ∨ (v < w ∧ w < u ∧ ¬vR̂w) then begin
2.1.4.1             TriType := TriTypes[Tricode(v, u, w)];
2.1.4.2             Census[TriType] := Census[TriType] + 1;
                end
            end;
3       sum := 0;
        for i := 2 to 16 do sum := sum + Census[i];
        Census[1] := 1/6 n(n − 1)(n − 2) − sum;
```

For a connected triad we can always assume that v is the smallest of its vertices. So we have in 2.1.4 to determine the canonical selection from the remaining two selections (v, u, w) and (v, w, u). If $v < w < u$ and $v\hat{R}w$ then the selection (v, w, u) was already counted before. Therefore we have to consider it as canonical only if $\neg v\hat{R}w$.

In an implementation of the algorithm we must take care about the range overflow in the case of T and T_1. For example, already $T(10000) = \binom{10000}{3} \approx 1.67\ 10^{11}$. It is easy to see that $|T_2| < n \cdot m$ and $|T_3| < \hat{\Delta} \cdot m$. Therefore, using in Delphi for counters the type *Int64* with 19-20 significant decimal digits, there should be no problem also with sparse networks on some millions of vertices.

2.5 Complexity of the algorithm

Assuming that the sets $\hat{R}(v)$ are represented by ordered lists, the count of dyadic triads (2.1.1 – 2.1.3) for each arc (v, u) can be done in time $O(\hat{\Delta})$. Therefore, the total complexity for counting dyadic triads is $T(T_2) = O(m\hat{\Delta})$.

The body (2.1.4.1 – 2.1.4.2) of the connected triads counting loop requires constant time. Therefore the complexity of counting connected triads $T(T_3)$ is of the same order as the number of all connected triads. Since every connected triad contains a vertex that is an 'origin of an angle' we have

$$|T_3| \leq \sum_{v \in V} \binom{\hat{d}(v)}{2} = \frac{1}{2} \sum_{v \in V} \hat{d}(v)(\hat{d}(v) - 1) \leq \frac{1}{2}(\hat{\Delta} - 1) \sum_{v \in V} \hat{d}(v) = (\hat{\Delta} - 1)m$$

Counting the null triads (3) can be done in constant time. Therefore the total complexity of the algorithm is $O(\hat{\Delta}m)$ and thus, for networks with small $\hat{\Delta} << n$, since $2m \leq n\hat{\Delta}$, of order $O(n)$.

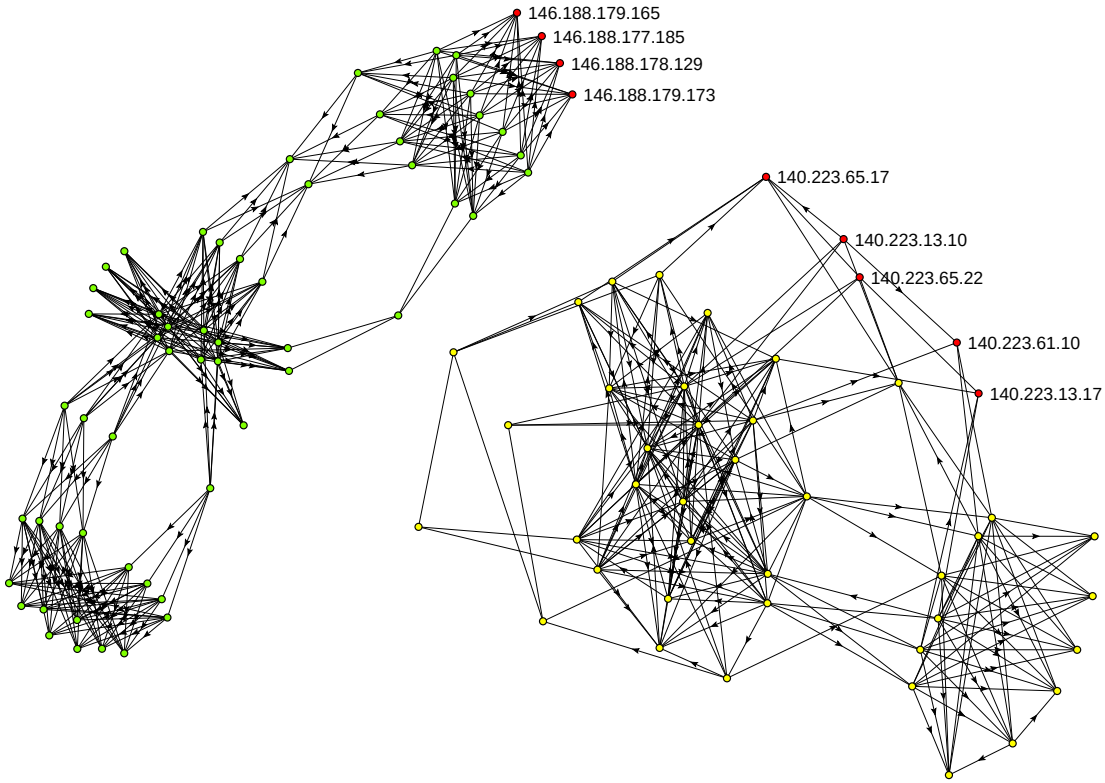

Figure 3: 6-Core of the Routing Data Network.

Note that in a large sparse network some vertices can still have large degrees. For such networks $\hat{\Delta} = O(n)$ and the proposed algorithm is quadratic.

3 Example – Internet Connections

As an example of application of the proposed algorithm we applied it to the routing data on the Internet network. This network was produced from web scanning data (May 1999) available from

`http://www.cs.bell-labs.com/who/ches/map/index.html`

It can be obtained as a **`Pajek`**'s NET file from

`http://vlado.fmf.uni-lj.si/pub/networks/data/web/web.zip`

It has $124\ 651$ vertices, $207\ 214$ arcs, $\hat{\Delta} = 153$ and average degree is 3.3. Figure 3 shows 6-core (arcs are counted in both directions) of the largest strongly connected component of the network. Using **`Pajek`** implementation of the proposed algorithm on 300 MHz PC we obtained in 10 seconds the triad census presented in Table 2.

There are only 69 complete triads (type 16 - 300). Using **`Pajek`**'s pattern searching we identified all of them and extracted the induced subnetwork from the network. The subnetwork has 18 components, 11 of them are 'triangles'. The other, nontriangular, components are presented in Figure 4.

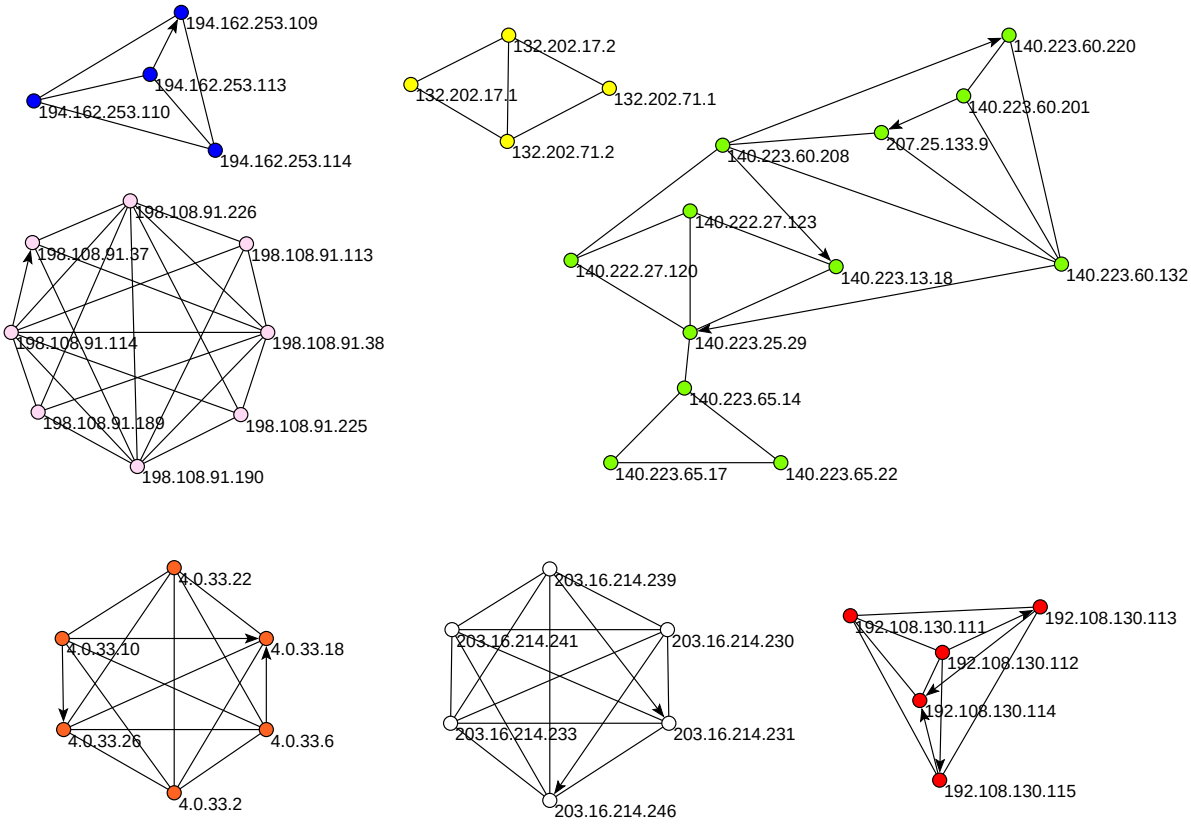

Figure 4: Nontriangular Complete Triadic Components of the Routing Data Network.

Table 2: Triad Census of the Routing Data Network.

Triad	Count
1 - 003	322 769 974 374 083
2 - 012	23 955 959 979
3 - 102	175 605 448
4 - 021D	882 596
5 - 021U	109 179
6 - 021C	444 490
7 - 111D	4 917
8 - 111U	15 508
9 - 030T	17 107
10 - 030C	111
11 - 201	1 002
12 - 120D	899
13 - 120U	1 120
14 - 120C	96
15 - 210	121
16 - 300	69

4 Acknowledgments

The algorithm was presented at 17th Leoben-Ljubljana seminar on Discrete mathematics, Leoben, April 27 - 29, 2000.

This work was supported by the Ministry of Science and Technology of Slovenia, Project J1-8532.

References

[1] Barabasi, A-L., Reka, A., and Hawoong, Jeong (1999): *Mean-field theory for scale-free random networks*. Physica A **272**, 173-187.
`http://www.nd.edu/~networks/`

[2] Batagelj, V., Mrvar, A. (1998): Pajek – A Program for Large Network Analysis. *Connections* **21 (2)**, 47-57.
`http://vlado.fmf.uni-lj.si/pub/networks/pajek/`

[3] Wasserman, S., Faust, K. (1994): *Social Network Analysis*. Cambridge University Press.

[4] Moody, J. (1998): Matrix methods for calculating the triad census. *Social Networks*, **20**, 291-299.