Erschienen in: Advances in Network Science : 12th International Conference and School, NetSci-X 2016, Wroclaw, Poland, January 11-13, 2016 ; Proceedings / Wierzbicki, Adam; Brandes, Ulrik; Schweitzer, Frank; Pedreschi, Dino (Hrsg.). - Cham : Springer, 2016. - (Lecture Notes in Computer Science ; 9564). - S. 1-13. - ISSN 0302-9743. - eISSN 1611-3349. - ISBN 978-3-319-28360-9
https://dx.doi.org/10.1007/978-3-319-28361-6_1

Quad Census Computation: Simple, Efficient, and Orbit-Aware

Mark Ortmann(✉) and Ulrik Brandes(✉)

Department of Computer and Information Science,
University of Konstanz, Konstanz, Germany
{Mark.Ortmann,Ulrik.Brandes}@uni-konstanz.de

Abstract. The prevalence of select substructures is an indicator of network effects in applications such as social network analysis and systems biology. Moreover, subgraph statistics are pervasive in stochastic network models, and they need to be assessed repeatedly in MCMC sampling and estimation algorithms. We present a new approach to count all induced and non-induced 4-node subgraphs (the quad census) on a per-node and per-edge basis, complete with a separation into their non-automorphic roles in these subgraphs. It is the first approach to do so in a unified manner, and is based on only a clique-listing subroutine. Computational experiments indicate that, despite its simplicity, the approach outperforms previous, less general approaches.

1 Introduction

The $\mathcal{F}$-census of a graph is the frequency distribution of subgraphs from a family $\mathcal{F}$ of non-isomorphic graphs in an input graph. In this work we focus on four node subgraphs, i.e. *quads*.

Discrimination of graphs by a subgraph census was proposed already by Holland and Leinhardt [7,8] in the context of social networks and it is of utmost importance for the effects of exponential random graph models [20]. While there is extensive work on determining the subgraph census for varying subgraph sizes [9,10,12] and also for directed graphs [4], the focus is almost always on the global distribution, i.e., say, the number of triangles a graph contains but not how often a given node is part of a triangle. However, for many properties describing nodes and edges, respectively, it is necessary to know the subgraph census on a node or edge level basis. For example to calculate a node's *clustering coefficient* we need to know in how many triangles it is contained. The same holds for the *Jaccard index* computed with respect to an edge. Although, for these two examples it is not necessary to calculate the frequencies of all non-isomorphic induced 3-node subgraphs, the triad census, there exist edge weights that take different subgraph configurations into account [1] and the running time for most edge metrics [14] is dominated by calculating the frequencies of particular subgraphs.

We gratefully acknowledge financial support from Deutsche Forschungsgemeinschaft under grant Br 2158/11-1.

While k-subgraph censuses specific for nodes and edges are not used widely in social network analysis, this is different already for bioinformatics. So far, however, even here the use is restricted to connected k-node subgraphs, so called *graphlets* [19] or *motifs* [17].

A further differentiation of subgraph censuses consist in the distinction of node and edge automorphism classes (orbits) in each graphlet. For example, in a diamond (i.e. a complete 4-node graph with one missing edge), there are two node and edge orbits, see Fig. 1. The node orbits are defined by the nodes with degree 2, and those with degree 3, respectively. The edge orbits are determined by the edge connecting the nodes with degree 3, and all remaining edges, respectively. Milenković and Pržulj [16] and Solova et al. [21] utilize the characterization of each node and edge respectively by it's orbit-aware connected subgraph census, which they call graphlet degree vector, for clustering purposes.

Due to the importance of subgraph enumeration and censuses in bioinformatics, various computational methods [6,13,15,23] were proposed.

The general approach to determine a subgraph census on the global level is to solve a system of equations that relates the non-induced subgraph frequency of each non-isomorphic k-node subgraph with the number of occurrences in other k-node subgraphs [4,5,9,10,12]. It is known that the time needed to solve the system of equations for the 4-node subgraph census, which we refer to as the *quad census*, on a global level is $\mathcal{O}(a(G)m + i(G))$ [12], where $i(G)$ is the time needed to calculate the frequency of a given 4-node induced subgraph in G, and $a(G)$ is the *arboricity*, i.e., the minimum number of spanning forest needed to cover E. Following the idea of relating non-induced and induced subgraph counts, Marcus and Shavitt [13] present a system of equations for the orbit-aware connected quad census on a node level that runs in $\mathcal{O}(\Delta(G)m + m^2)$ time with $\Delta(G)$ denoting the maximum degree of G. Because of the larger number of algorithms invoked by Marcus and Shavitt's approach, Hočevar and Demšar [6] present a different system of equations, again restricted to connected quads, that requires fewer counting algorithms and runs in $\mathcal{O}(\Delta(G)^2 m)$ time, but does not determine the non-induced counts.

Contribution: We present the first algorithm to count both induced and non-induced occurrences of all 4-node subgraphs (quads). It is based on a fast algorithm for listing a single quad type and capable of distinguishing the various roles (orbits) of nodes and edges. While this simplifies and generalizes previous approaches, our experimental evaluation indicates that it is also more efficient.

In the following section we provide basic notation followed by an introduction of the system of equations and the algorithm utilized in Sect. 3. In Sect. 4 we present a running time comparison on real world and synthetic graphs showing that our approach is more efficient than related methods. We conclude in Sect. 5.

2 Preliminaries

We consider finite simple undirected graphs $G = (V, E)$ and denote the number of nodes by $n = n(G) = |V|$ and the number of edges by $m = m(G) = |E|$. The *neighborhood* of a node $v \in V$ is the set $N(v) = \{w : \{v, w\} \in E\}$ of all adjacent nodes, its cardinality $d(v) = |N(v)|$ is called the *degree* of v, and $\Delta(G) = \max_{v \in V}\{d(v)\}$ denotes the maximum degree of G.

For finite simple directed graphs $G = (V, E)$ we denote the *outgoing neighborhood* of a node $v \in V$ by $N^+(v) = \{w : (v, w) \in E\}$. The *incoming neighborhood* $N^-(v)$ is defined analogously.

A complete graph with k nodes is called K_k and K_3 is also called a *triangle*. We use $T(u) = \{\binom{N(u)}{2} \cap E\}$ to refer to the set of node pairs completing a triangle with u and $T(\{u, v\}) = N(u) \cap N(v)$ for the set of nodes completing a triangle with the edge $e = \{u, v\}$. For the cardinality of these sets we write $t(u) = |T(u)|$ and $t(e) = |T(e)|$. A *triad* or a *quad* are any graphs on exactly three or four nodes, respectively.

A subgraph $G' = (V', E')$ of $G = (V, E)$, $V' \subseteq V$, is called (node-)*induced*, if $E' = \binom{V'}{2} \cap E$, and it is called *non-induced*, if $E' \subseteq \binom{V'}{2} \cap E$.

Two undirected graphs G and G' are said to be *isomorphic*, if and only if there exists a bijection $\pi : V(G) \to V(G')$ such that $\{v, w\} \in E(G) \iff \{\pi(v), \pi(w)\} \in E(G')$. Each permutation, including identity, of the node set V, such that the resulting graph is isomorphic to G is called an *automorphism* and the groups formed by the set of automorphisms is denoted *automorphism class* or *orbit*.

3 Determining the Orbit-Aware Quad Census

The k-node subgraph census is usually computed via a system of linear equations relating the non-induced and induced k-subgraph frequencies, as the non-induced frequencies are easier to compute. Lin et al. [12] show that for $k = 4$ all non-induced frequencies, except for K_4, can be computed in $\mathcal{O}(a(G)m)$ time. This implies that the total running time to calculate the quad census at the level of the entire graph is in $\mathcal{O}(a(G)m + i(G))$, where $i(G)$ is the time needed to compute the induced frequencies for any induced quad.

The approach of Lin et al., however, is not suitable to answer questions such as how often a node or an edge is contained in a K_4. Furthermore, the automorphism class of the node/edge in the quad is sometimes of interest. All non-isomorphic graphs with four nodes are shown in Fig. 1 and the node/edge labels refer to their automorphism classes (orbits). For example in a diamond all edges of the C_4 belong to the same orbit while the diagonal edge belongs to another. Analogously the orbits of the nodes can be distinguished.

As our approach also relies on relating non-induced and induced frequencies we will start by presenting how the non-induced frequencies for a node/edge in a given orbit relate to the induced counts. Thereafter, we will present formulas to compute the respective non-induced frequencies and prove that our approach

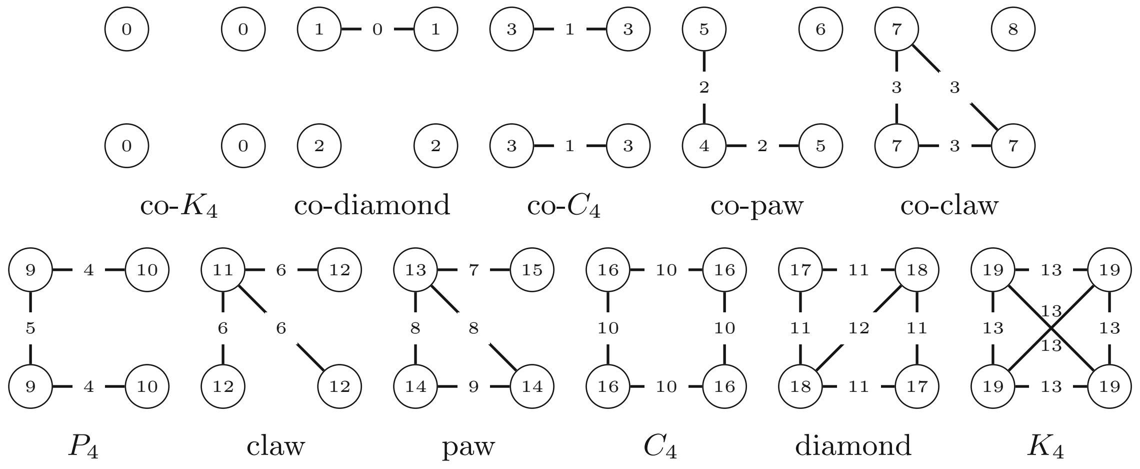


Fig. 1. All non-isomorphic subgraphs with four nodes (quads). Node and edge labels refer to the orbits and were enumerated such that each orbit is identified with a single quad

matches the running time of Lin et al., implying that it is asymptotically as fast as the fastest algorithm to compute the frequencies on a node and edge level for any induced quad. Note that in the following when we talk about non-induced frequencies we exclude those of the K_4, as it equals the induced frequency.

3.1 Relation of Induced and Non-induced Frequencies

To establish the relation between induced and non-induced frequencies, the number of times G' is non-induced in some other graph G with the same number of nodes has to be known. For instance, let us assume that G' is a P_3 and G a K_3 (co-paw and -claw without isolated node cf. Fig. 1). Having the definition of the edge set for non-induced subgraphs in mind we see that G contains three non-induced P_3, as each edge can be removed from a K_3 to create a P_3. Consequently, if we know the total number of non-induced P_3 and we subtract three times the number of K_3 we obtain the number of induced P_3 of the input graph.

Similarly, we can establish systems of equations relating induced and non-induced frequencies on a node and edge level distinguishing the orbits for quads, see Figs. 3 and 4. Note that both systems of equations are needed since we cannot compute the node from the edge frequencies and vice versa, but from both we can compute the global distribution. In the following we show the correctness for $ei_{10}(e)$.

Induced Orbit 10 Edge Census. Let us assume we want to know how often edge e is in orbit 10 or in other words part of a C_4. We know that a C_4 is a non-induced subgraph of a diamond, K_4 and of itself, cf. Fig. 2, and that there is no other quad containing a non-induced C_4. Let us first concentrate on the diamond. In a diamond we have two different edge orbits; orbit 11, i.e. the edges on the C_4, and orbit 12, i.e. the diagonal edge. Figure 2 shows that for every diamond where e is in orbit 12 there is no way to remove an edge, such that this graph becomes a C_4, but for each diamond where e is in orbit 11 we can remove the diagonal

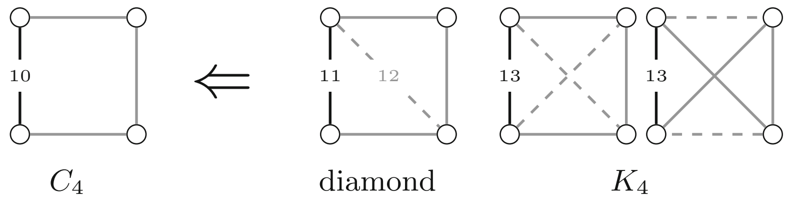


Fig. 2. The three quads containing a non-induced C_4. Dashed lines indicate that their removal creates an C_4. Edge label correspond to orbits

edge and end up with a C_4. Therefore, the non-induced number of subgraphs where e is in orbit 10 contains once the number of induced subgraphs where e is in orbit 11, but not those in orbit 12. As for the case of the C_4 in a K_4 all edges are in the same orbit. From a K_4 we can construct a C_4 in two ways. The first is to remove both diagonal edges, cf. Fig. 2; and the second to delete the two horizontal edges. As a consequence the induced number of e being in orbit 10 is given by $ei_{10}(e) = en_{10}(e) - ei_{11}(e) - 2ei_{13}(e)$.

Following this concept all other equations can be derived.

3.2 Calculating Non-induced Frequencies

The calculation of the non-induced frequencies is (computationally) easier than for the corresponding induced counts, except for K_4s. This is due to the fact that the non-induced frequencies can be constructed from smaller, with respect to the number of nodes, subgraphs. In the following we show the correctness of $nn_{14}(u)$ and $en_4(u, v)$.

Non-induced Orbit 14 Node Census. To determine $nn_{14}(u)$ we start by enumerating all triangles containing u. Let v and w form a triangle together with u. As u is in orbit 14 we know that each neighbor of v and w that is not u, v or w definitely creates a non-induced paw with u in orbit 14. While this does not necessarily hold for neighbors of u as they might not be connected to v or w (and, if they are, we already gave credit to this). Note that if x is a neighbor of u and v but not w we can only create one non-induced paw with u in orbit 14 and therefore $nn_{14}(u) = \sum_{\{v,w\} \in T(u)} (d(u) - d(v) - 4)$.

Non-induced Orbit 4 Edge Census. Edge $e = \{u, v\}$ is non-induced in orbit 4 for each path of length 2 starting at u or v that does not contain e. The number of P_3s starting at u equals $\sum_{w \in N(u) \setminus v} d(w) - 1$. However, the node v might be a neighbor of w and therefore, there is a path of length two connecting u and v. Since this creates a 3-node subgraph, more precisely a triangle, and not a quad we have to adjust for this by subtracting twice the number of triangles containing e. Consequently, $en_4(u, v) = \sum_{w \in N(u)} d(w) + \sum_{w \in N(v)} d(w) - 2(d(u) + d(v)) + 2 - 2t(u, v)$.

In the following, we focus on the algorithm calculating all required frequencies to solve the systems of equations.

$$
\begin{bmatrix}
1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
0&1&0&0&1&0&0&1&0&1&1&1&0&1\\
0&0&1&2&1&2&2&2&3&2&2&3&4&4\\
0&0&0&1&0&0&0&0&1&1&0&1&2&2\\
0&0&0&0&1&0&0&2&0&2&2&3&0&4\\
0&0&0&0&0&1&0&0&1&0&1&1&2&2\\
0&0&0&0&0&0&1&1&1&0&0&1&2&2\\
0&0&0&0&0&0&0&1&0&0&0&1&0&2\\
0&0&0&0&0&0&0&0&1&0&0&1&4&4\\
0&0&0&0&0&0&0&0&0&1&0&1&0&2\\
0&0&0&0&0&0&0&0&0&0&1&1&0&2\\
0&0&0&0&0&0&0&0&0&0&0&1&0&4\\
0&0&0&0&0&0&0&0&0&0&0&0&1&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1
\end{bmatrix}
\begin{bmatrix}
ei_0(u,v)\\ ei_1(u,v)\\ ei_2(u,v)\\ ei_3(u,v)\\ ei_4(u,v)\\ ei_5(u,v)\\ ei_6(u,v)\\ ei_7(u,v)\\ ei_8(u,v)\\ ei_9(u,v)\\ ei_{10}(u,v)\\ ei_{11}(u,v)\\ ei_{12}(u,v)\\ ei_{13}(u,v)
\end{bmatrix}
=
\begin{bmatrix}
en_0(u,v) &= \binom{n-2}{2}\\
en_1(u,v) &= m-d(u)-d(v)+1\\
en_2(u,v) &= (d\,(u)+d\,(v)-2)\,(n-3))\\
en_3(u,v) &= t(u,v)(n-3)\\
en_4(u,v) &= \sum_{w\in N(u)} d(w)+\sum_{w\in N(v)} d(w)-2(d(u)+d(v))+2-2t(u,v)\\
en_5(u,v) &= (d(u)-1)(d(v)-1)-t(u,v)\\
en_6(u,v) &= \binom{d(u)-1}{2}+\binom{d(v)-1}{2}\\
en_7(u,v) &= t(u)+t(v)-2t(u,v)\\
en_8(u,v) &= t(u,v)\cdot(d(u)+d(v)-4)\\
en_9(u,v) &= \sum_{w\in T(u,v)} d(w)-2t(u,v)\\
en_{10}(u,v) &= |(N(u)\setminus v\times N(v)\setminus u)\cap E|\\
en_{11}(u,v) &= \sum_{w\in T(u,v)} t(u,w)+t(v,w)-2t(u,v)\\
en_{12}(u,v) &= \binom{t(u,v)}{2}\\
en_{13}(u,v) &= \text{Alg. }\texttt{K4}
\end{bmatrix}
$$

Fig. 3. System of equations for orbit aware quad census on a edge level. *ei* refers to the induced and *en* non-induced counts

$$
\begin{bmatrix}
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
0&1&0&1&2&1&0&2&0&2&1&3&1&3&2&1&2&2&3&3\\
0&0&1&1&2&1&0&2&0&2&1&3&1&1&2&3&2&3&2&3\\
0&0&0&1&0&0&0&0&0&1&1&0&0&1&1&1&2&2&2&3\\
0&0&0&0&1&0&0&1&0&1&0&3&0&3&1&0&1&1&3&3\\
0&0&0&0&0&1&0&2&0&1&1&0&2&2&3&2&2&4&4&6\\
0&0&0&0&0&0&1&0&3&0&1&0&1&0&1&3&1&3&1&3\\
0&0&0&0&0&0&0&1&0&0&0&0&0&1&1&0&0&1&2&3\\
0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&1&0&1\\
0&0&0&0&0&0&0&0&0&1&0&0&0&2&1&0&2&2&4&6\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&2&2&4&2&6\\
0&0&0&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&1&1\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&1&1&0&2&1&3\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&2&3\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&2&2&6\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&2&0&3\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&3\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&3\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&3\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1
\end{bmatrix}
\begin{bmatrix}
ni_0(u)\\ ni_1(u)\\ ni_2(u)\\ ni_3(u)\\ ni_4(u)\\ ni_5(u)\\ ni_6(u)\\ ni_7(u)\\ ni_8(u)\\ ni_9(u)\\ ni_{10}(u)\\ ni_{11}(u)\\ ni_{12}(u)\\ ni_{13}(u)\\ ni_{14}(u)\\ ni_{15}(u)\\ ni_{16}(u)\\ ni_{17}(u)\\ ni_{18}(u)\\ ni_{19}(u)
\end{bmatrix}
=
\begin{bmatrix}
nn_0(u) &= \binom{n-1}{3}\\
nn_1(u) &= \binom{n-2}{2}d(u)\\
nn_2(u) &= (m-d(u))\cdot(n-3)\\
nn_3(u) &= (\sum_{v\in N(u)} m-d(u))-d(u)\cdot(d(u)-1)\\
nn_4(u) &= \binom{d(u)}{2}(n-3)\\
nn_5(u) &= ((\sum_{v\in N(u)} d(v))-d(u))\cdot(n-3)\\
nn_6(u) &= \sum_{v\in V}\binom{d(v)}{2}-((\sum_{v\in N(u)} d(v))-d(u))-\binom{d(u)}{2}\\
nn_7(u) &= t(u)\cdot(n-3)\\
nn_8(u) &= \frac{1}{3}\sum_{v\in V} t(v)-t(u)\\
nn_9(u) &= \sum_{v\in N(u)}(d(u)-1)\cdot(d(v)-1)-t(u,v)\\
nn_{10}(u) &= (\sum_{v\in N(u)}(\sum_{w\in N(v)} d(w))-d(v))-d(u)\cdot(d(u)-1)\\
nn_{11}(u) &= \binom{d(u)}{3}\\
nn_{12}(u) &= \sum_{v\in N(u)}\binom{d(v)-1}{2}\\
nn_{13}(u) &= t(u)\cdot(d(u)-2)\\
nn_{14}(u) &= \sum_{\{v,w\}\in T(u)}(d(v)+d(w)-4)\\
nn_{15}(u) &= \sum_{v\in N(u)} t(v)-t(u,v)\\
nn_{16}(u) &= \sum_{\{v,w\}\in\binom{N(u)}{2}}|N(v)\cap N(w)|-1\\
nn_{17}(u) &= \sum_{\{v,w\}\in T(u)} t(v,w)-t(u)\\
nn_{18}(u) &= \sum_{v\in N(u)}\binom{t(u,v)}{2}\\
nn_{19}(u) &= \text{Alg. }\texttt{K4}
\end{bmatrix}
$$

Fig. 4. System of equations for orbit aware quad census on a node level. *ni* refers to the induced and *nn* non-induced counts

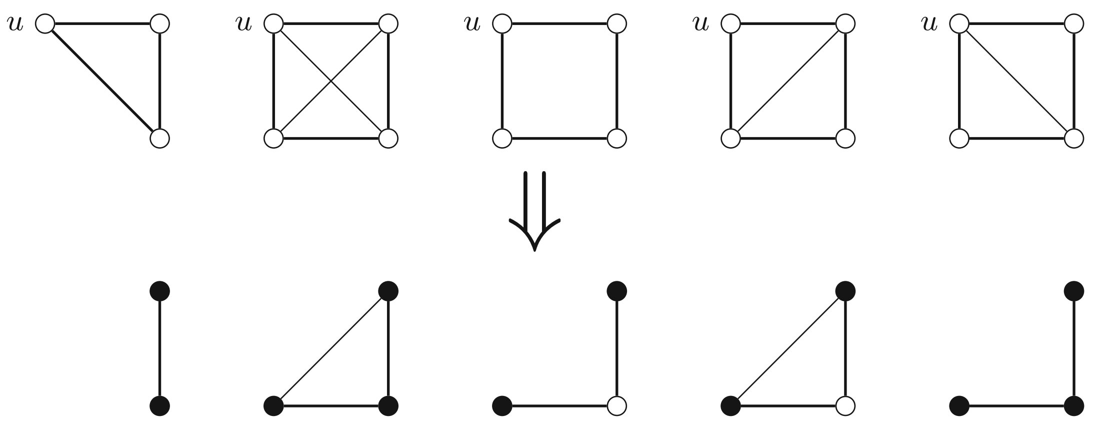


Fig. 5. Top: Configurations that have to be found by our algorithm. Bottom: Resulting patterns to be detected when processing node u. Filled nodes are marked as neighbors of u

3.3 Listing Complete Quads

In order to be able to solve the systems of equations we need to compute the non-induced quad counts as well as any of the induced frequencies. This requires an algorithm that is capable of solving the following tasks on a node and edge level:

1. Counting and listing all K_3.
2. Calculating non-induced C_4 frequencies.
3. Determine induced counts of any quad.

We chose to calculate the induced counts for K_4 to fulfill requirement 3. The reasons are (a) to our knowledge there are no algorithms calculating induced counts on a node and edge level for any other quad more efficiently than the algorithm we are presenting here; (b) a K_4 has the property that all nodes and edges lie in the same orbit; (c) all non-induced C_4 can be counted during the execution of our algorithm. Since listing, also known as enumerating, all K_4 has to solve the subproblem of listing all K_3 we will start explaining our algorithm by presenting how K_3s can be listed efficiently. Note that this algorithm satisfies requirement 1.

Listing all triangles in a graph is a well studied topic [18]. We show in our previous work [18] that one of the oldest triangle listing algorithms, namely `K3` by Chiba and Nishizeki [3] is in practice the fastest. This algorithm is based on neighborhood intersection computations. To achieve the running time of $\mathcal{O}(a(G)m)$ Chiba and Nishizeki process the graph in a way, such that for each intersection only the neighborhood of the smaller degree node has to be scanned. This is done by processing the nodes sequentially in decreasing order of their degree. The currently processed node marks all its neighbors and is removed from the graph. Then the number of marked neighbors of a marked node is calculated.

Let us think of this algorithm differently. When we process node u and remove it from the graph then every triangle that contains u is an edge where both endpoints are marked, cf. Fig. 5. This perception of the algorithm directly points

Algorithm 1. K3 / C4 / COMPLETE (Chiba and Nishizeki 1985 [3])

```
1  initialize $mark$ with 0;
2  order nodes by successively removing the node of min. degree from the graph;
3  orient $G$ and sort adjacencies according node ordering;
4  calculate $t(u)$ and $t(e)$ using K3; // line 5-15
5  for $u = v_2, \dots, v_n$ do
6      for $v \in N^-(u)$ do  $mark(v) \leftarrow mark(v) + 1$ ;
7      for $v \in N^-(u)$ do
8          $mark(v) \leftarrow mark(v) - 1$;
9          for $w \in \{w \in N(v) : w < u\}$ do
10             $visited(w) \leftarrow visited(w) + 1$;
11             $processed(w) \leftarrow processed(w) + 1$;
12         for $w \in \{w \in N^+(v) : w < u\}$ do  $mark(w) \leftarrow mark(w) + 2$ ;
13         for $w \in \{w \in N^+(v) : w < u\}$ do
14             $mark(w) \leftarrow mark(w) - 2$;
15             if $mark(w) \neq 0$ then
16                 increment $K_3$ related non-induced counts;
17                 for $x \in \{x \in N^+(w) : x < u\}$ do
18                     if $mark(x) = 3$ then
19                         increment induced $K_4$ count;
20     for $v \in N^-(u)$ do
21         for $w \in \{w \in N(v) : w < u\}$ do
22             $processed(w) \leftarrow processed(w) - 1$;
23             if $processed(w) = 0$ then
24                 increment non-induced $C_4$ of $u$ and $w$ by $\binom{visited(w)}{2}$;
25                 $visited(w) \leftarrow 0$;
26             increment non-induced $C_4$ of $\{u, v\}, \{v, w\}$ and $v$ by $visited(w) - 1$;
27 solve system of equations;
```

us to a solution for the second and third requirement. As shown in Fig. 5, when node u is removed from the graph, every K_4 that contains u becomes a K_3 where all nodes are marked, implying that K3 can be easily adapted to list all K_4s. Chiba and Nishizeki call this extension COMPLETE. Furthermore, only nodes that are connected to a neighbor of u can create a non-induced C_4 and each C_4 contains at least two marked nodes. Since all these nodes are processed already during the execution of algorithm K3 counting non-induced C_4 on a node and edge level can be also done in $\mathcal{O}(a(G)m)$ time. The corresponding algorithm is called C4 in [3] and the combination of these different algorithms is presented in Algorithm 1. It runs in $\mathcal{O}(a(G)^2m)$ [3], and its novelty is that it follows the idea of directing the graph acyclic as we already proposed in the context of triangle listing [18]. Furthermore, this acyclic orientation allows omitting node removals, and given the proper node ordering, it has the property that the maximum outdegree is bounded by $\mathcal{O}(a(G))$. Therefore, unlike for algorithm `COMPLETE` and `C4` [3], no amortized running time analysis is of need to prove that the running time is in $\mathcal{O}(a(G)^2 m)$ and $\mathcal{O}(a(G)m)$, respectively, as we will show next.

Runtime. We will first show that the running time bound of our variant implementation of algorithm `C4` is in $\mathcal{O}(a(G)m)$, therefore we ignore lines 4, 6, 8, 12 – 19 and 27 of Algorithm 1 for now.

The running time of the remaining algorithm is given by the following equation:

$$\begin{aligned} t(\texttt{C4}) &\leq \sum_{u\in V} d^-(u) + 2 \sum_{v\in N^-(u)} d^-(v) + d^+(v) \\ &= m + 2\sum_{v\in V} d^+(v)(d^-(v) + d^+(v)) \\ &\leq m + 4m\Delta^+(G) \end{aligned}$$

As we order the nodes by successively removing the node of minimum degree from the graph, which can be computed in $\mathcal{O}(m)$ using a slightly modified version of the algorithm presented in [2], it holds that $\Delta^+(G) < 2a(G)$ [24]. The time required to initialize all marks is in $\mathcal{O}(n)$, orienting the graph is in $\mathcal{O}(n+m)$, and consequently the total running time is in $\mathcal{O}(a(G)m)$.

Let us now focus on the time required for calculating all K_4s and therefore ignore lines 9 – 11 and 20–27 of Algorithm 1 that is given by the following equation:

$$\begin{aligned} t(\texttt{COMPLETE}) &\leq \sum_{u\in V} d^-(u) + \sum_{v\in N^-(u)} 2d^+(v) + \sum_{w\in N^+(v)} d^+(w) \\ &\leq m + \Delta^+(G)\sum_{v\in V} 2d^+(v) + \sum_{w\in N^+(v)} d^+(w) \\ &\leq m + 2m\Delta^+(G) + \Delta^+(G)\sum_{v\in V} d^-(v)\Delta^+(G) \\ &= m + 2m\Delta^+(G) + m\Delta^+(G)^2 \end{aligned}$$

By the same arguments it follows that our variant implementation of `COMPLETE` runs in $\mathcal{O}(a(G)^2 m)$. Since, line 4 is in $\mathcal{O}(a(G)m)$ [18] and solving the system of equations requires $\mathcal{O}(n+m)$ time, the overall complexity of Algorithm 1 is in $\mathcal{O}(a(G)^2 m)$.

4 Runtime Experiments

We provide experimental evidence that our approach is not only asymptotically faster but also more efficient in practice than the currently fastest orbit-aware quad census algorithm. Comparison is restricted to the *orca software* implementing the approach of Hočevar and Demšar [6], as the authors show that it is superior to other software tools in the context of quad census computation.

Additionally, it is the only software we are aware of that can compute the orbit-aware quad census on an edge level, even if only for connected quads. To the best of our knowledge, except in the orca code, there is no other documentation of their approach.

4.1 Setup and Data

We implemented our approach in C++ using the *Standard Template Library* and compiled the code with the g++ compiler version 4.9.1 set to the highest optimization level. The *orca software* is freely available as an R package. To avoid measuring error due to the R and C++ interface communication we extracted the C++ code and cleaned it from all R dependencies.

The tests were carried out on a single 64-bit machine with an 3.60 GHz quad-core Intel Core i7-4790 CPU, 32 GB RAM, running Ubuntu 14.10. The times were measured via the `gettimeofday` command with a resolution up to 10^{-6} s. We ran the executable in a single thread and forced it to one single core, which was dedicated only to this process. Times were averaged over 5 repetitions.

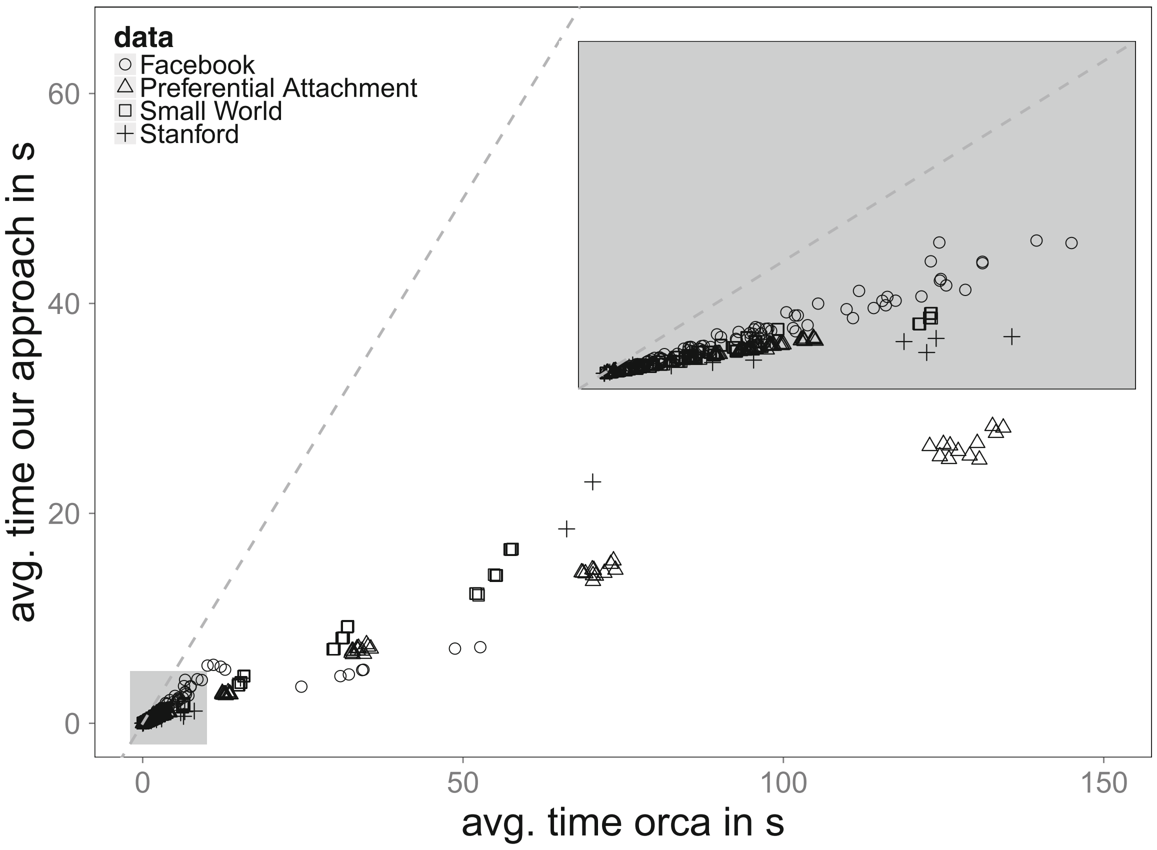
Data. We compared both approaches on a number of real world networks. The *Facebook100 dataset* [22] comprises 100 Facebook friendship networks of higher educational institutes in the US with network sizes of $762 \leq n < 41K$ nodes and $16K < m < 1.6M$ edges. Although these networks are rather sparse they feature a small diameter, thereby implying a high concentration of connected quads. Apart from this we tested the algorithms on a variety of networks from the *Stanford Large Network Set Collection* [11]. The downloaded data were taken from different areas to have realistic examples that encompass diverse network structures.

Additionally, we generated synthetic networks from two different models. The one class of generated graphs are small-worlds, which were created by arranging nodes on a ring, connecting each one with its r nearest neighbors, and then switching each dyad with probability p. The other class of graphs are drawn from a preferential attachment like model. Here we added n nodes over time to the initially empty network and each new node connects to r existing nodes, each of which either chosen by preferential attachment or with probability p by random. We generated graphs with fixed $n = 20000$ and varying average degree as well as graphs with $n \in \{50000, 140000, \ldots, 500000\}$ and gradually increasing average degree. Four graphs were generated for each parameter combination.

We refer the reader to [18] for a more detailed description of the utilized graph models, the tested Stanford graphs, the chosen average degree, and parameters r and p.

4.2 Results

In Fig. 6 we present the results of our experiments. In the top subfigure we plotted the avg. time needed by our approach against the avg. running time of *orca* for all but the largest Standford graphs. Each point that lies below the



Graph	$\vert V\vert$	$\vert E\vert$	Alg. 1	orca	speedup
wiki-Talk	2 394 385	4 659 565	1m33s	7m41s	4.95
com-LiveJournal	3 997 962	34 681 189	2m47s	14m00s	5.00
soc-LiveJournal1	4 847 571	42 851 237	5m09s	24m33s	4.77
com-orkut	3 072 441	117 185 083	22m06s	109m52s	4.97

Fig. 6. Top: Avg. running time of *orca* vs. avg. running time of our approach in seconds for all but the largest SNAP graphs. Dots below the main diagonal indicate that the algorithm on the y-axis is faster. Embedded plot displays gray area in higher resolution. Bottom: Time comparison for the largest SNAP graphs

main diagonal indicates that our approach is faster than orca. Consequently, the picture makes it clear that our algorithm is faster than the *orca software* for each tested network, even though we compute the whole node and edge orbit aware quad census. The same findings are obtained for the larger graphs taken from SNAP.

The speed-up we achieve lies between 1.6 and 10 for the tested graphs. In general, however, the speed-up should be in $\Theta(\log \Delta(G))$ for larger graphs. The reason is that, once n exceeds $30K$, the algorithm implemented in the orca software runs in $\mathcal{O}(\Delta(G)^2 m \log \Delta(G))$, instead of $\mathcal{O}(\Delta(G)^2 m)$. The logarithmic factor originates from the time required for adjacency testing. While the orca software uses an adjacency matrix for these queries for graphs with $n \leq 30K$, it takes $\mathcal{O}(\log \Delta(G))$ for larger graphs (binary search) since no adjacency matrix is constructed. In contrast Algorithm 1 requires only $\mathcal{O}(n)$ additional space to perform adjacency tests in constant time. Note that orca's algorithm using the adjacency matrix appears to follow the ideas of Chiba and Nishizeki, though without exploiting the potential of utilizing a proper node ordering. Besides the faster K_4 algorithm another important aspect explaining the at least constant speed-up of our approach is our system of equations. For both the node and edge orbit-aware quad census Hočevar and Demšar do not calculate the exact non-induced counts. This requires that each induced subgraph with 3 nodes is listed several times and more importantly also non-cliques, which is not the case in our approach.

5 Conclusion

We presented two systems of equations that enable us to efficiently determine the orbit-aware quad census of a graph down to the level of nodes and edges by applying an efficient single-subgraph listing algorithm and it's subroutine. It was shown how induced and non-induced frequencies relate to one another and that we can compute the non-induced frequencies in $\mathcal{O}(a(G)m)$ time. This matches the best known running time bound for the more restricted non-induced quad census on the graph level, i.e., oblivious to the specific nodes and edges involved in each quad. With Algorithm 1 we showed a routine that is capable of computing all non-induced frequencies and listing all K_4 while running in $\mathcal{O}(a(G)^2m)$ time, which is the asymptotically best known running time bound for listing any induced quad. This implies that the total running time of our approach matches the best known running time for quad census computation on a graph level in sparse graphs [12]. In experiments we were able to show that the simplicity of our system of equations in combination with this efficient algorithm outperforms the currently best software to calculate the quad census. We point out that Algorithm 1 can be parallelized with little effort and by following the same technique our orbit aware quad census can be extended to directed graphs.

References

1. Auber, D., Chiricota, Y., Jourdan, F., Melançon, G.: Multiscale visualization of small world networks. In: 9th IEEE Symposium on Information Visualization (InfoVis 2003), 20–21 October 2003, Seattle, WA, USA (2003)
2. Batagelj, V., Mrvar, A.: A subquadratic triad census algorithm for large sparse networks with small maximum degree. Soc. Netw. **23**(3), 237–243 (2001)
3. Chiba, N., Nishizeki, T.: Arboricity and subgraph listing algorithms. SIAM J. Comput. **14**(1), 210–223 (1985)
4. Eppstein, D., Goodrich, M.T., Strash, D., Trott, L.: Extended dynamic subgraph statistics using h-index parameterized data structures. Theoret. Comput. Sci. **447**, 44–52 (2012). doi:10.1016/j.tcs.2011.11.034
5. Eppstein, D., Spiro, E.S.: The h-index of a graph and its application to dynamic subgraph statistics. In: Dehne, F., Gavrilova, M., Sack, J.-R., Tóth, C.D. (eds.) WADS 2009. LNCS, vol. 5664, pp. 278–289. Springer, Heidelberg (2009)

6. Hočevar, T., Demšar, J.: A combinatorial approach to graphlet counting. Bioinformatics **30**(4), 559–565 (2014)
7. Holland, P.W., Leinhardt, S.: A method for detecting structure in sociometric data. Am. J. Sociol. **76**(3), 492–513 (1970)
8. Holland, P.W., Leinhardt, S.: Local structure in social networks. Sociol. Methodol. **7**, 1–45 (1976)
9. Kloks, T., Kratsch, D., Müller, H.: Finding and counting small induced subgraphs efficiently. Inf. Process. Lett. **74**(3–4), 115–121 (2000)
10. Kowaluk, M., Lingas, A., Lundell, E.: Counting and detecting small subgraphs via equations and matrix multiplication. In: Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, 23–25 January 2011, pp. 1468–1476 (2011)
11. Leskovec, J., Krevl, A.: SNAP Datasets: stanford large network dataset collection, June 2014. http://snap.stanford.edu/data
12. Lin, M.C., Soulignac, F.J., Szwarcfiter, J.L.: Arboricity, h-index, and dynamic algorithms. Theor. Comput. Sci. **426**, 75–90 (2012)
13. Marcus, D., Shavitt, Y.: RAGE - A rapid graphlet enumerator for large networks. Comput. Netw. **56**(2), 810–819 (2012)
14. Melançon, G., Sallaberry, A.: Edge metrics for visual graph analytics: a comparative study. In: 12th International Conference on Information Visualisation, IV 2008, 8–11 July 2008, London, UK, pp. 610–615 (2008)
15. Milenković, T., Lai, J., Pržulj, N.: GraphCrunch: a tool for large network analyses. BMC Bioinformatics **9**(70), 1–11 (2008)
16. Milenković, T., Pržulj, N.: Uncovering biological network function via graphlet degree signatures. Cancer Inf. **6**, 257–273 (2008)
17. Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., Alon, U.: Network motifs: simple building blocks of complex networks. Science **298**(5594), 824–827 (2002)
18. Ortmann, M., Brandes, U.: Triangle listing algorithms: back from the diversion. In: 2014 Proceedings of the Sixteenth Workshop on Algorithm Engineering and Experiments, ALENEX 2014, Portland, Oregon, USA, 5 January 2014, pp. 1–8 (2014)
19. Pržulj, N., Corneil, D.G., Jurisica, I.: Modeling interactome: scale-free or geometric? Bioinformatics **20**(18), 3508–3515 (2004)
20. Robins, G., Pattison, P., Kalish, Y., Lusher, D.: An introduction to exponential random graph (p^*) models for social networks. Soc. Netw. **29**(2), 173–191 (2007)
21. Solava, R.W., Michaels, R.P., Milenković, T.: Graphlet-based edge clustering reveals pathogen-interacting proteins. Bioinformatics **28**(18), 480–486 (2012)
22. Traud, A.L., Kelsic, E.D., Mucha, P.J., Porter, M.A.: Comparing community structure to characteristics in online collegiate social networks. SIAM Rev. **53**(3), 526–543 (2011)
23. Wernicke, S., Rasche, F.: FANMOD: a tool for fast network motif detection. Bioinformatics **22**(9), 1152–1153 (2006)
24. Zhou, X., Nishizeki, T.: Edge-coloring and f-coloring for various classes of graphs. In: Du, D.Z., Zhang, X.S. (eds.) Algorithms and Computation. LNCS, vol. 834, pp. 199–207. Springer, Heidelberg (1994)