BIOINFORMATICS ORIGINAL PAPER

Vol. 30 no. 4 2014, pages 559–565
doi:10.1093/bioinformatics/btt717

Data and text mining

Advance Access publication December 11, 2013

A combinatorial approach to graphlet counting

Tomaž Hočevar* and Janez Demšar

Faculty of Computer and Information Science, University of Ljubljana, SI-1000 Ljubljana, Slovenia

Associate Editor: Igor Jurisica

ABSTRACT

Motivation: Small-induced subgraphs called graphlets are emerging as a possible tool for exploration of global and local structure of networks and for analysis of roles of individual nodes. One of the obstacles to their wider use is the computational complexity of algorithms for their discovery and counting.

Results: We propose a new combinatorial method for counting graphlets and orbit signatures of network nodes. The algorithm builds a system of equations that connect counts of orbits from graphlets with up to five nodes, which allows to compute all orbit counts by enumerating just a single one. This reduces its practical time complexity in sparse graphs by an order of magnitude as compared with the existing pure enumeration-based algorithms.

Availability and implementation: Source code is available freely at http://www.biolab.si/supp/orca/orca.html.

Contact: tomaz.hocevar@fri.uni-lj.si

Supplementary information: Supplementary data are available at *Bioinformatics* online.

Received on August 30, 2013; revised on November 29, 2013; accepted on December 6, 2013

1 INTRODUCTION

Following the advent of high-throughput methods more than a decade ago, analysis of complex network data has assumed the central role among computational methods in bioinformatics. The huge size of such networks on one hand and the computational intractability of the related methods on the other have spawned a number of innovative analytic approaches.

Pržulj *et al.* (2004) described an approach focused on small induced subgraphs called graphlets. Owing to combinatorial explosion, such analysis is usually limited to the 30 graphlets with 2–5 nodes (Fig. 1). The number of appearances of graphlets in the network provides a description of the network's structural properties. On a local level, counting how many times a particular node participates in each kind of graphlet induced in the network gives a topological signature of the node's neighbourhood represented as a 30-dimensional vector.

For a finer description, the nodes of every graphlet are partitioned into a set of automorphism groups called orbits (Pržulj, 2007). Two nodes belong to the same orbit if they map to each other in some isomorphic projection of the graphlet onto itself. Nodes of graphlets on 2–5 points are grouped into 73 orbits shown by numbers and node colors in Figure 1. For instance, the five nodes from G_{14} belong to three different orbits, marked with different colors and numbers; the black (as well as the grey) nodes have symmetric positions in the graphlet and thus belong to the same orbit (31 for the black, 32 for the grey), and the white node belongs to the orbit 33. By counting the number of times a node of a graph appears in each orbit, the node can be described by a 73-dimensional vector of orbit counts, which reflects its position with respect to the local structure and gives insight into its role in the network.

Existing methods for counting the graphlets and orbits are based on direct enumeration: to count them, they need to find all their embeddings in the network. We propose a new method, Orbit Counting Algorithm (Orca), which reduces the time complexity by an order of magnitude by computing the orbit counts using the relations between them and directly enumerating only smaller graphlets.

1.1 Motivation

Graphlets are used for different kinds of analyses in bioinformatics. Milenković and Pržulj (2008) designed a method for comparing node neighbourhoods based on graphlets and demonstrated that clusters of nodes in protein–protein interaction (PPI) networks, obtained with their graphlet-based distance measure, share common protein properties. They showed how to use this approach to predict functions of proteins and their memberships in protein complexes, subcellular compartments and tissue expressions. Milenković *et al.* (2010b) studied the relation between cancer genes and their network topology. They examined several clustering methods based on a graphlet similarity measure and found a difference between the PPI network structure around the cancer and non-cancer genes. Around 80% of the predicted cancer gene candidates have been validated in the literature. Similarly, cost functions for network alignment that are based on graphlet degree vectors show superior results in comparison with other state-of-the-art methods. In particular, Milenković *et al.* (2013) showed how alignment between the PPI networks of *Saccharomyces cerevisiae*, *Drosophila melanogaster* and *Caenorhabditis elegans* with the human PPI network can be used for identification of genes related to aging, which are difficult to observe directly for humans due to our long lifespans. Milenković *et al.* (2011) also applied graphlets to estimate node's topological centrality. Their graphlet degree centrality measure is based on graphlet degree vectors and captures density and complexity of a node's extended neighbourhood. They showed that the genes participating in key biological processes also reside in complex and dense parts of networks.

Hayes *et al.* (2013) argue that to understand the biological networks, we need to find the mathematical models describing their structure, even though this may not be of direct predictive use. Pržulj *et al.* (2004) used graphlet distributions to show that

*To whom correspondence should be addressed.

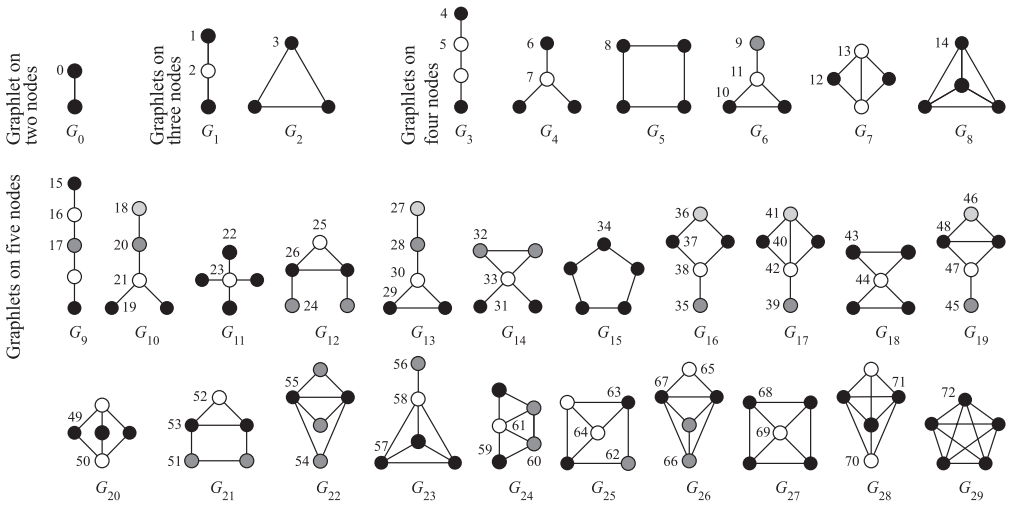

Fig. 1. Graphlets with 2–5 nodes and automorphism orbits. Notation follows (Pržulj, 2007). Colors are chosen arbitrarily; nodes of the same color belong to the same orbit within that graphlet, e.g. both black nodes in G_{14} belong to orbit 31

geometric graphs match the structure of PPI networks better than Erdős–Rényi and scale-free graph models. Using a number of large PPI networks, Hayes *et al.* (2013) further showed that although the network structure may be unstable in regions with low edge-density, high-density regions are suitable for network comparison using graphlet degree distributions.

Graphlets can also assist in other analytic methods, such as global network alignment. GRAph ALigner (GRAAL) (Kuchaiev *et al.*, 2010) is an algorithm for aligning arbitrary networks based solely on their topology, which uses a local topology similarity measure based on graphlet degree vectors. The technique was used to show the large amount of shared network topology between yeast and human PPI networks, which can be used to predict biological functions of aligned proteins or reconstruct phylogenetic trees. H-GRAAL (Milenković *et al.*, 2010a) aligns networks by reducing the problem to a weighted bipartite matching that can be solved with Hungarian algorithm. Finally, MI-GRAAL (Kuchaiev and Pržulj, 2011) integrates multiple sources of node similarity information, including the graphlet degree vectors.

Solava *et al.* (2012) extended the use of graphlets by defining the orbits for graphlet edges and demonstrated their use with a new clustering method that is not limited to locally similar edges and allows some overlap between clusters. As a practical result, they predicted new pathogen-interacting proteins from clusters in the human PPI network that represent drug target candidates.

Therefore, graphlet analysis is a useful tool for bioinformatics, and with the increase of available data there is also a growing need for fast graphlet counting tools.

1.2 Related work

We will denote the explored graph as $G = (V, E)$. Let $= |V|$ and $e = |E|$ be the number of vertices and edges, and let d denote the maximal node degree. Let $N(u)$ denote the set of vertices adjacent to vertex u. In numbering the graphlets and orbits, we follow Pržulj (2007); we refer to the j-th graphlet and i-th orbit by G_j and O_i, respectively.

Counting subgraphs is a computationally intensive task. Common approaches to speed it up include sampling (Kashtan *et al.*, 2004; Pržulj *et al.*, 2006; Wernicke, 2006), exploiting pattern symmetries (Stoica and Prieur, 2009) or using reconfigurable hardware accelerators based on Field-Programmable Gate Array (FPGA) chips (Betkaoui *et al.*, 2011).

The method described in this article is related to the approach developed by Kloks *et al.* (2000), who constructed a system of equations that allows computing the number of occurrences of all six induced four-node subgraphs by knowing the count of any of them. The time complexity of setting up the system equals the time complexity of multiplying two square matrices of size n. We extend this approach to counting how many times each node participates in each orbit. Our method also works on five-node graphlets and scales better on sparse graphs. Kowaluk *et al.* (2011) generalized the result by Kloks *et al.* (2000) to count subgraph patterns of arbitrary size.

There are several programs for graphlet counting and motif detection that are used in bioinformatics. Fast Network Motif Detection (FANMOD) (Wernicke and Rasche, 2006) is a network motif detection tool based on sampling random subgraphs and comparing their counts with those from random network models. Besides implementing a novel sampling algorithm (Wernicke, 2006), it also provides a full enumeration procedure for graphlets on 2–8 nodes. Whelan and Sönmez (2012) developed GraphletCounter, which works as a Cytoscape plugin and merges graphlet analysis with visual inspection of the network.

GraphCrunch (Milenković *et al.*, 2008) is a tool for large network analysis. It includes a function for computing orbit signatures of every graph node for graphlets of up to five nodes using an enumeration procedure with correction for over-counting some of the graphlets. A well-organized enumeration method imposes constraints that eliminate the need for isomorphism testing except for distinguishing between a few different graphlets; this is further accelerated by comparing the number of edges and individual node degrees. GraphCrunch has been extended with a new method for topological network alignment and with

comparison of the networks with some additional mathematical models (Kuchaiev *et al.*, 2011). The graphlet counting procedure in the new version remained essentially the same.

Rapid graphlet enumerator (RAGE) (Marcus and Shavitt, 2012) takes a different approach to counting four-node graphlets. Instead of counting the induced subgraphs directly, it reconstructs them from counts of non-induced subgraphs. For computing the latter, it uses specifically crafted methods for each of the six possible subgraphs (G_3 to G_8 in Fig. 1). The time complexity of counting non-induced cycles and complete graphs is $O(e \cdot d + e^2)$, whereas counting other subgraphs requires $O(e \cdot d)$. Another bound, which is also more suitable for comparison with our method, is $O(e \cdot d^2) = O(n \cdot d^3)$. Unlike FANMOD and GraphCrunch, RAGE works only for up to four-node graphlets.

2 METHODS

Let x represent a certain node of interest in graph G. Our task is to compute the number of times, o_i, that x appears in each orbit O_i across all graphlets induced in G. We will present an approach based on a system of linear equations that relate the orbit counts o_i. The rank of the system is smaller than the number of orbits by one, so we can compute all values of o_i from directly enumerating only a single one. The algorithm allows to compute the orbits for all points x in a graph in time that is smaller than the existing direct enumeration approaches by an order of magnitude.

We will first show how to construct a system of equations for four-node graphlets. As for the single orbit that must be enumerated, we chose O_{14}, which represents nodes of the complete graph, K_4 (or G_8); we show an efficient way to enumerate it. The approach used for four-node graphlets is less suitable for larger graphlets, so we present a different technique for five-node graphlets.

2.1 Orbits in four-node graphlets

Right sides of equations we are about to construct contain terms that are computed from the graph G. Let $c(u, v) = |N(u) \cap N(v)|$ denote the number of common neighbours of nodes u and v. Let $p(u, v)$ denote the number of paths on three nodes that start at node u, continue with v and end with some node t, which is not connected to u. We can compute $p(u, v)$ as $p(u, v) = deg(v) - 1 - c(u, v)$.

If some node x participates in a k-node graphlet G_i, it also participates in some $(k-1)$-node graphlet G_j. This can be seen by removing one of the graphlet's nodes that are the farthest away from x. The subgraph induced by the remaining nodes is connected (any disconnected node would have to be farther from x than the removed node), so it is isomorphic to some $(k-1)$-node graphlet G_j.

We will use this observation in reverse: every four-node graphlet can be constructed by adding a node to some three-node graphlets. To find the relations between counts of orbits in four-node graphlets for a certain node x, we enumerate all three-node graphlets touching the node and count their possible extensions with the fourth node.

An example is shown in Figure 2. Nodes x, y and z induce graphlet G_1, a path on three nodes; we will observe its extensions to four-node graphlets with the fourth node, w, connected to y and z (dashed lines). The number of such nodes w is $c(y, z)$. In our example, there are $c(y, z) = 3$ such nodes, which we marked by w_1, w_2 and w_3 (Fig. 2a). The edge (x, w) might exist in the graph G (as in the case of w_3, the dotted line) or not (as for w_1 and w_2). With no edge, nodes x, y, z and w form a paw (G_6) with x in orbit O_9 (Fig. 2b). With an edge between x and w, they form a diamond (G_7) with x in orbit O_{12} (Fig. 2c). Because all $c(y, z)$ nodes in $N(y) \cap N(z)$ must participate either in G_6 or G_7, which puts x in O_9 or O_{12}, this gives $o_9 + o_{12} = c(y, z)$ for the particular triplet x, y and z.

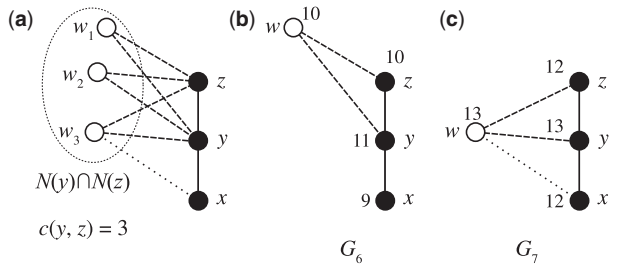

Fig. 2. Relation between orbits O_9 and O_{12}. Solid lines are edges in the three-node graphlet being extended. Dashed lines exist by definition: w (or w_i) are the common neighbours of y and z. Dotted lines are optional edges that make the resulting four-node graphlet on x, y, z and w_i isomorphic to G_6 or G_7

We sum this over all possible three-node paths starting at x. Summation must account for symmetries: each graphlet G_6 appearing in the graph is counted twice with roles of z and w reversed, and G_7 is counted twice with reversed roles of y and w. Accounting for this, we get

$$2o_9 + 2o_{12} = \sum_{\substack{y,\, z:\, x,\, z \in N(y) \\ G[\{x,y,z\}] \cong G_1}} c(y, z)$$

where $\cong$ denotes graph isomorphism (e.g. $G[\{x, y, z\}]$, a subgraph on nodes x, y and z is isomorphic to G_1, a path with three nodes).

For a different example, we will relate orbits O_6 and O_9. We will extend a path on nodes x, y and z with another path that starts with nodes x and y; we denoted the number of such paths by $p(x, y)$ (Fig. 3a). Depending on whether the new node is adjacent to z, the extended graphlet is either a claw (G_4, Fig. 3b) or a paw (G_6, Fig. 3c). After accounting for symmetries and subtracting 1, as $p(x, y)$ also covers the case when $w = z$, we get

$$2o_6 + 2o_9 = \sum_{\substack{y,\, z:\, x,\, z \in N(y) \\ G[\{x,y,z\}] \cong G_1}} (p(x, y) - 1)$$

There are only two three-node graphlets and relatively few possible extensions. Investigating all possibilities in a similar manner yields 10 linearly independent equations with 11 variables that correspond to counts of 11 orbits in four-node graphlets (see the Supplementary Material).

Right sides depend on the graph G and need to be computed for each point x. To accelerate their computation, we precompute values of $c(u, v)$ and $p(u, v)$. In all equations, except for the last one, $c(u, v)$ is computed on pairs of nodes (u, v) that are connected; in $p(u, v)$, they are connected by the definition of p. Therefore, it suffices to precompute $c(u, v)$ and $p(u, v)$ only for all pairs of connected nodes u and v, which requires $O(e)$ space. The last equation, in which the new node closes a cycle, is treated separately. Nodes x and z are not adjacent but we can precompute the number of paths of length 2 that start at node x and end at node y. This requires $O(n)$ space for each point; because we compute orbits for one point at a time, this memory can be recycled. Altogether, all lookups in the sums on the right sides can be done in constant time by sacrificing the memory of size $O(e + n)$ for precomputed values $c(x, y)$ and $p(x, y)$.

The total time complexity for computing all orbits for all nodes is $O(e \cdot d + T_4)$, where $O(T_4)$ is the time needed to enumerate complete graphlets on four nodes. Later in the text, we describe an algorithm that does this in $O(n \cdot d^3)$, yet the actual importance of this term depends on the structure and density of the graph.

2.2 Counting complete graphlets

For every node, we still have to determine the count of one of the 11 orbits. Because graphs are usually sparse, a good candidate is the rare orbit 14, which represents the nodes of the complete graphlet on four

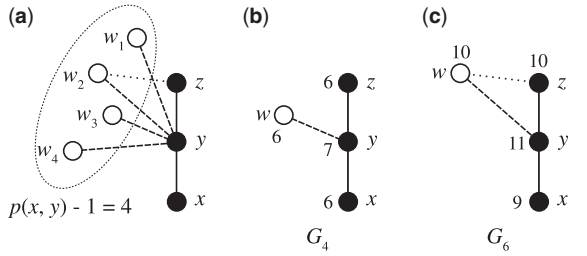

Fig. 3. Relation between orbits O_6 and O_9. Edges are marked like in Figure 2

nodes G_8. Because of few occurrences of this graphlet and its symmetricity, we can efficiently restrict the enumeration.

A straightforward way to count the complete graphlets of size four that touch a given node x_1 is to start with that node and in every step add a neighbour x_i of the last added node x_{i-1}, while checking that the new node is also connected to all nodes before x_i, $x_{j<i-1}$. In this way, when we add x_4 as a neighbour of x_3 we have to check whether it is connected to x_1 and x_2 (dotted lines in Fig. 4a), which is unlikely, especially in sparse graphs.

A better strategy is to find the common neighbours of x_1 and x_2, $N(x_1) \cap N(x_2)$, which can be done in $O(d)$. We then choose pairs (x_3, x_4) from this set and check whether they are connected (Fig. 4b). Candidates generated in this way have to satisfy only one additional condition, as opposed to two in the straightforward approach.

To avoid counting the same graphlet multiple times, we request that $x_2 < x_3 < x_4$ under some fixed arbitrary ordering of nodes. Although the theoretical time complexity for finding all G_8 that touch x using this algorithm is the same for both approaches, $O(d^3)$, the latter is much faster on sparse graphs.

This method can be generalized for efficient counting of larger complete graphlets in sparse graphs. In every step, we maintain a list of candidate nodes C_i for x_i that are adjacent to all previously added nodes. We select one of these candidates and form a new candidate set C_{i+1} consisting only of nodes in C_i that are adjacent to the selected node, $C_{i+1} = C_i \cap N(x_i)$ and $C_1 = V$. The time complexity of finding all complete k-node graphlets that touch x using this algorithm is $O(d^{k-1})$. Later in the text, we use such procedure to enumerate complete subgraphs on five nodes.

2.3 Orbits on five-node graphlets

For counting four-node graphlets, we constructed a list of equations by adding nodes to three-node graphlets and observing the resulting four-node graphlets. Extending the four-node graphlets to five-node graphlets would yield a huge number of equations that are not linearly independent. We will use a different approach: for each orbit, we choose some node y from the corresponding graphlet and observe the graphlets and orbits in which the node of interest, x, appears if we add edges between y and other nodes in the graphlet.

Let x be the node of interest, let y be the node whose edges we observe and let x_1, x_2 and x_3 be the other three nodes in that graphlet.

Figure 5 illustrates counting of appearances of x in O_{59}, which belongs to G_{24} (Fig. 5a). We will focus on the node marked by y, which is connected to the nodes marked by x_1 and x_3. Removing y reduces G_{24} into a diamond, G_7, with x in orbit O_{12}.

Now assume that we are computing orbits for a certain node x and discover some induced subgraph $H \cong G_7$ with x in O_{12}. We assign labels x_1, x_2 and x_3 to the remaining nodes as shown in the figure. Altogether, the graph G contains $c(x_1, x_3)$ common neighbours of x_1 and x_3 (similar to nodes marked with w in Fig. 3a). Although all these nodes are—by definition of $c(x_1, x_3)$—connected to x_1 and x_3, some are also connected

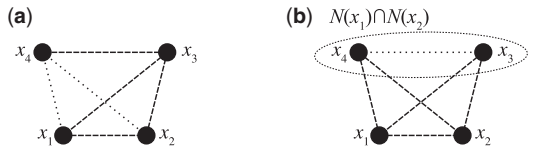

Fig. 4. Enumerating G_8 by adding one neighbour at a time or by checking pairs of neighbours. Dashed edges are added by iterating through neighbours, and dotted edges are checked in the last step

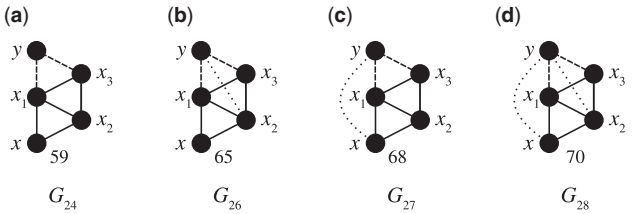

Fig. 5. Computing orbit count o_{59}; figures show graphlets for different edges between y and other nodes and the orbits of x

to x_2 or x, or both. Figure 5 shows all four possibilities, which give graphlets G_{24}, G_{26}, G_{27} and G_{28} with x in orbits 59, 65, 68 and 70, respectively. Therefore, $o'_{59} + o'_{65} + o'_{68} + o'_{70} = c(x_1, x_3) - 1$, where o'_i denote orbits of x with respect to H.

For the relation between o_{59}, o_{65}, o_{68} and o_{70} for the entire graph, we sum this over all possible induced G_7 with x in O_{12}. After considering the symmetries that cause counting the same graphlet multiple times with different assignments of y, x_1, x_2 and x_3, we get

$$o_{59} + 4o_{65} + 2o_{68} + 6o_{70} = \sum_{\substack{x_1, x_2, x_3: \\ x_1 < x_2 \wedge x_3 \notin N(x), \\ G[\{x, x_1, x_2, x_3\}] \cong G_7}} c(x_1, x_3) + c(x_2, x_3) - 2$$

Condition $x_1 < x_2$ (under some arbitrary ordering of nodes) is needed to consider each graphlet G_7 just once. The other two conditions put x in O_{12}. The second term in the sum, $c(x_2, x_3)$, accounts for the case in which the roles of x_1 and x_2 are exchanged.

Using a similar construction for other orbits, except for O_{72}, gives 57 linear equations for 58 orbits (see the Supplementary Material). Like for four-node graphlets, we directly enumerate the orbit O_{72}, which belongs to the complete graphlet. Equations are linearly independent due to the way in which they were constructed: each equation is set up with one orbit in mind (e.g. O_{59} in the aforementioned example), and the other orbits that appear in the equation belong to graphlets with a larger number of edges (the additional edges between y and the other nodes, like the dotted edges in Fig. 5b–d). Additional nice consequence besides independence is that the system is easy to solve, as orbit counts can be computed from those belonging to graphlets with more edges towards those with less.

When constructing the equations, we choose y that allows for efficient computation of the right sides: we will ensure that the right sides contain only the node degrees and the numbers of common neighbours of pairs and of *connected* triplets $[c(u, v), c(u, v, t)]$. This will allow us to precompute and store the values of $c(u, v)$ and $c(u, v, t)$ for all pairs and connected triplets in G before computing the orbit counts for individual nodes.

First, we choose the node y so that the remaining nodes constitute a four-node graphlet, i.e. removing y does not break the graphlet into disconnected components, which would require enumeration of disconnected subgraphs. Second, the node y has to have at most three connections to avoid the need to compute the number of common

neighbours of four points, $c(u, v, w, t)$. Besides, when y has three neighbours, they need to be connected.

A node y that fulfils these criteria exists for all orbits except O_{72}. Precomputing the values $c(u, v, t)$ for all connected triplets takes $O(e \cdot d^2)$ time, and storing them in a hash table takes $O(e \cdot d)$ space. Computation of the right sides also requires enumerating all the four-nodes graphlets, which again has a complexity of $O(e \cdot d^2)$.

The total time required to compute all orbit counts for all $x \in V$ is then $O(e \cdot d^2 + T_5)$ with $O(e \cdot d)$ space, where $O(T_5)$ is the time required to enumerate all complete five-node graphlets (G_{29}). The algorithm thus has the same upper bound complexity as the existing algorithms, $O(n \cdot d^4)$. However, experiments show that the bound is not tight: the contribution of the $O(T_5)$ is negligible over the range of sensible graph densities, and the actual running times are smaller by an order of magnitude.

We could use the same technique to construct systems of equations for larger graphlets. However, we reduced the running times by imposing some conditions to the selection of the node y. We have not researched whether such nodes also exist for larger graphlets; although theoretically interesting, this may be of little practical use in the context of bioinformatics.

3 RESULTS AND DISCUSSION

We compared the speed of Orca with RAGE, GraphCrunch and FANMOD. We ran all experiments on a modest desktop computer (Intel Core 2, 2.67 GHz). We have not experimented with parallel execution; all four algorithms allow for trivial distribution of work on multiple cores, so the benefits of parallelization should be the same for all.

We compared the performance of methods on the three largest species-specific PPI networks from the July 2013 update of the Database of Interacting Proteins (Salwinski *et al*., 2004) and the human PPI network from the BioGRID (Chatr-Aryamontri *et al*., 2013) 3.2.104 release. The sizes of individual datasets are presented in Table 1.

All algorithms except the significantly slower FANMOD counted orbits for four-node graphlets in the smaller graphs in a few seconds (Table 2). Five-node graphlets present a more difficult task: running GraphCrunch on the *S.cerevisiae* PPI network took >9 min (as compared with 4.4 s for four-node graphlets). FANMOD was almost 10 times slower, whereas Orca finished the same task 80 times faster, in 6.6 s. RAGE is limited to four-node graphlets. We got similar results for the other two networks.

In the larger human network, Orca counted the four-node graphlets 100 and 1800 times faster than Rage and GraphCrunch, respectively; we aborted FANMOD after 24 h. Orca was also the only algorithm capable of counting five-node graphlets in a human PPI network in less than a day.

For comparison with RAGE, we included a test network of Internet autonomous systems (http://www.netdimes.org/PublicData/csv/ASEdges4_2012.csv.gz) that was used as the benchmark for RAGE (Marcus and Shavitt, 2012). FANMOD required >9 h, GraphCrunch finished in 37 min, RAGE in 3 min and Orca in 2.5 s. Orca finished the computation for five-node graphlets in 49 min, whereas the other two algorithms were stopped after 24 h.

The time that Orca needs for counting orbits in five-node graphlets is comparable with those that GraphCrunch needs for four-node graphlets. This is consistent with the way the two algorithms are constructed: GraphCrunch enumerates four-node graphlets to count them, whereas Orca enumerates them to count five-node graphlets. As expected, the time needed for enumeration of complete five-node graphlets is negligible at these network densities.

For more insight into time complexities of the compared algorithms, we tested them on synthetic data using three different random network models—Erdős–Rényi, geometric and Barabási–Albert, random graphs. Erdős–Rényi graphs are constructed by randomly connecting e pairs of nodes. We generated geometric graphs by randomly placing nodes in a 3D unit cube and connecting the e closest pairs; geometric graphs show largest resemblance to protein interaction networks (Pržulj *et al*., 2004). Barabási–Albert preferential attachment model generates scale-free networks that exhibit hubs and individual highly connected nodes.

Table 1. Statistics of benchmark real-world networks

Network	Nodes	Edges	Maximum degree
S.cerevisiae	5097	22 282	289
Escherichia coli	2984	11 626	178
D.melanogaster	7618	22 864	178
Human	18 170	1 37 775	9716
Internet autonomous systems	25 368	75 004	3781

Table 2. Comparison of algorithms on real-world networks

Network	Four-node graphlets				Five-node graphlets		
	FANMOD	GraphCrunch	RAGE	Orca	FANMOD	GraphCrunch	Orca
S.cerevisiae	62 s	4.4 s	1.7 s	<0.1 s	87 min	9.5 min	6.6 s
E.coli	34 s	1.8 s	1.0 s	<0.1 s	38 min	4.1 min	4.8 s
D.melanogaster	21 s	3.1 s	1.6 s	<0.1 s	18 min	2.8 min	2.3 s
Human	/	183 min	11.8 min	6.1 s	/	/	269 min
Internet autonomous systems	574 min	37 min	3.0 min	2.5 s	/	/	49 min

Note: We aborted the algorithms that took more than a day and marked the corresponding results with /.

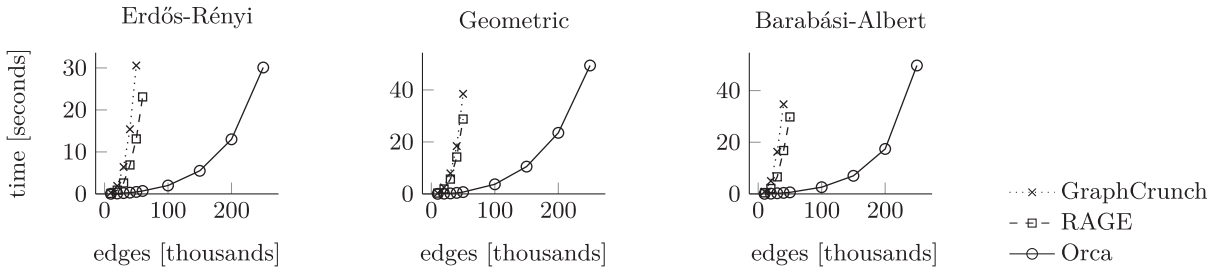

Fig. 6. Comparison of times needed for counting orbits in four-node graphlets in random networks. Graphs are cut off at one minute; results of experiments in which the methods were allowed to run for up to 1 h are available in the Supplementary Material

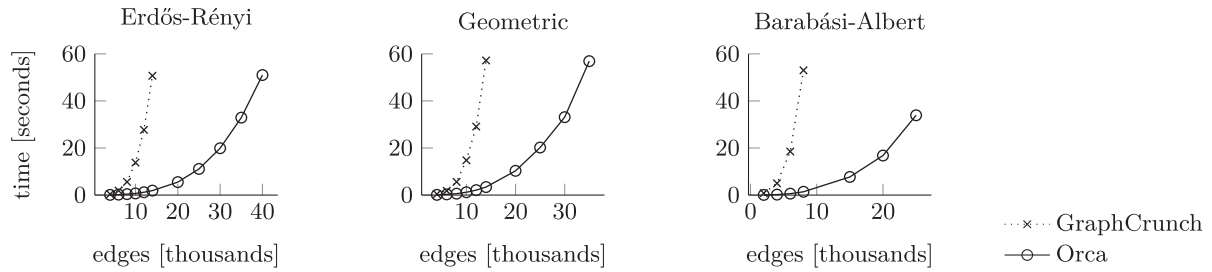

Fig. 7. Comparison of times needed for counting orbits in five-node graphlets in random networks

We explored the performance of GraphCrunch, RAGE and Orca at different network densities. FANMOD was not included as it consistently finished previous tests far behind GraphCrunch. All graphs had 1000 nodes; for each method, we increased the graph density until the method needed more than a minute to complete the test. The corresponding graphs were relatively dense, containing up to 40% of all possible edges for test with four-node graphlets and ~10% for five-node graphlets.

RAGE counted the four-node graphlets slightly faster than GraphCrunch, but they were both significantly outperformed by Orca (Fig. 6 and Supplementary Tables S1–S6). We observed similar results when counting five-node graphlets (Fig. 7). Orca achieved the highest gain in comparison with other methods on Barabási–Albert models, in which hubs present a large obstacle for GraphCrunch and RAGE. This makes Orca more suitable for real-world networks, which often display the small-world property and contain hubs.

4 CONCLUSION

Graphlet-based network analysis is useful for various tasks in bioinformatics, such as alignment of PPI networks and prediction of protein functions based on topological similarities. Past studies used these approaches to, for instance, identify genes related to cancer (Milenković *et al*., 2010b) and aging (Milenković *et al*., 2013).

We presented a new algorithm for counting graphlet orbits that is based on derived relations between orbit counts. To count the orbits for k-node graphlets, it enumerates $(k-1)$-node graphlets and a single k-node graphlet. Empirical results confirm that this decreases the time complexity by an order of magnitude in comparison with other known methods. In practical terms, the algorithm counts orbits in large PPI networks 50–100 times faster than other state-of-the-art algorithms.

Funding: Slovenian Research Agency (P2-0209, J2-5480).

Conflict of Interest: none declared.

REFERENCES

Betkaoui,B. *et al.* (2011) A framework for FPGA acceleration of large graph problems: graphlet counting case study. In: *2011 International Conference on Field-Programmable Technology*. IEEE, pp. 1–8.

Chatr-Aryamontri,A. *et al.* (2013) The BioGRID interaction database: 2013 update. *Nucleic Acids Res.*, **41**, D816–D823.

Hayes,W. *et al.* (2013) Graphlet-based measures are suitable for biological network comparison. *Bioinformatics (Oxford, England)*, **29**, 483–491.

Kashtan,N. *et al.* (2004) Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. *Bioinformatics*, **20**, 1746–1758.

Kloks,T. *et al.* (2000) Finding and counting small induced subgraphs efficiently. *Inf. Process. Lett.*, **74**, 115–121.

Kowaluk,M. *et al.* (2011) Counting and detecting small subgraphs via equations and matrix multiplication. In: *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*. pp. 1468–1476.

Kuchaiev,O. and Pržulj,N. (2011) Integrative network alignment reveals large regions of global network similarity in yeast and human. *Bioinformatics (Oxford, England)*, **27**, 1390–1396.

Kuchaiev,O. *et al.* (2010) Topological network alignment uncovers biological function and phylogeny. *J. R. Soc. Interface*, **7**, 1341–1354.

Kuchaiev,O. *et al.* (2011) GraphCrunch 2: software tool for network modeling, alignment and clustering. *BMC Bioinformatics*, **12**, 24.

Marcus,D. and Shavitt,Y. (2012) RAGE - a rapid graphlet enumerator for large networks. *Comput. Netw.*, **56**, 810–819.

Milenković,T. and Pržulj,N. (2008) Uncovering biological network function via graphlet degree signatures. *Cancer Inform.*, **6**, 257–273.

Milenković,T. *et al.* (2008) GraphCrunch: a tool for large network analyses. *BMC Bioinformatics*, **9**, 70.

Milenković,T. *et al.* (2010a) Optimal network alignment with graphlet degree vectors. *Cancer Inform.*, **9**, 121–137.

Milenković,T. *et al.* (2010b) Systems-level cancer gene identification from protein interaction network topology applied to melanogenesis-related functional genomics data. *J. R. Soc.*, **7**, 423–437.
Milenković,T. *et al.* (2011) Dominating biological networks. *PLoS One*, **6**, e23016.
Milenković,T. *et al.* (2013) Global network alignment in the context of aging. In: *Proceedings of the International Conference on Bioinformatics, Computational Biology and Biomedical Informatics*. pp. 23–32.
Pržulj,N. (2007) Biological network comparison using graphlet degree distribution. *Bioinformatics (Oxford, England)*, **23**, e177–e183.
Pržulj,N. *et al.* (2004) Modeling interactome: scale-free or geometric? *Bioinformatics (Oxford, England)*, **20**, 3508–3515.
Pržulj,N. *et al.* (2006) Efficient estimation of graphlet frequency distributions in protein-protein interaction networks. *Bioinformatics (Oxford, England)*, **22**, 974–980.
Salwinski,L. *et al.* (2004) The database of interacting proteins: 2004 update. *Nucleic Acids Res.*, **32**, D449–D451.
Solava,R.W. *et al.* (2012) Graphlet-based edge clustering reveals pathogen-interacting proteins. *Bioinformatics (Oxford, England)*, **28**, i480–i486.
Stoica,A. and Prieur,C. (2009) Structure of neighborhoods in a large social network. In: *2009 International Conference on Computational Science and Engineering*. IEEE, pp. 26–33.
Wernicke,S. (2006) Efficient detection of network motifs. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, **3**, 347–359.
Wernicke,S. and Rasche,F. (2006) FANMOD: a tool for fast network motif detection. *Bioinformatics (Oxford, England)*, **22**, 1152–1153.
Whelan,C. and Sönmez,K. (2012) Computing graphlet signatures of network nodes and motifs in Cytoscape with GraphletCounter. *Bioinformatics (Oxford, England)*, **28**, 290–291.