Condensed Graphs: a generic framework for accelerating subgraph census computation

Miguel Martins[1] and Pedro Ribeiro[1]

CRACS & INESC-TEC
DCC-FCUP, Universidade do Porto, Portugal
mlmartins@fc.up.pt, pribeiro@dcc.fc.up.pt

Abstract. Determining subgraph frequencies is at the core of several graph mining methodologies such as discovering network motifs or computing graphlet degree distributions. Current state-of-the-art algorithms for this task either take advantage of common patterns emerging on the networks or target a set of specific subgraphs for which analytical calculations are feasible. Here, we propose a novel network generic framework revolving around a new data-structure, a *Condensed Graph*, that combines both the aforementioned approaches, but generalized to support any subgraph topology and size. Furthermore, our methodology can use as a baseline any enumeration based census algorithm, speeding up its computation. We target simple topologies that allow us to skip several redundant and heavy computational steps using combinatorics. We were are able to achieve substantial improvements, with evidence of exponential speedup for our best cases, where these patterns represent up to 97% of the network, from a broad set of real and synthetic networks.

Keywords: subgraph frequency, subgraph census, condensed graph

1 Introduction

Many complex real world problems can be modelled with networks, from which we need to extract information. Several graph mining methodologies rely on understanding the importance of subgraphs as a very rich topological characterization. Two broadly known examples are are network motifs [12] and graphlet degree distributions [15]. At the core of these approaches lies the subgraph census problem, that is, computing the frequencies of a set of subgraphs. However, this is a fundamentally hard computational task that is related so the *subgraph isomorphism problem*, which is NP-complete [3].

Current algorithms for counting subgraphs typically rely on one of two different conceptual approaches. Several algorithms, such as G-tries [17], QuateXelero [9] or FaSE [13], are based on a subgraph enumeration phase intertwined with isomorphic testing to discover the topological class of each enumerated subgraph occurrence. These algorithms are very general and take advantage of common topologies to speedup the isomorphism computation. Other methods such as ORCA [6] or Escape [14] do not depend on fully enumerating all subgraph occurrences, but at the same time are geared towards more specific and less general sets of subgraphs, taking advantage of some of their analytical properties.

In this paper we propose an hybrid methodology that draws inspiration from both the enumeration and analytical approaches, that we called the **C**ondensation-**D**econdensation **F**ramework (**CFW**). The core motivation is to take advantage of combinatorial effects that create substantial speedups while at the same time being able to keep the general applicability of enumeration based algorithms, not constraining the subgraphs or networks being analyzed. For this we also introduce a new generic data structure, *Condensed Graphs*, that compresses subgraphs in a lossless way, capturing multiple occurrences of the same subtopology. More specifically, and as a proof of concept, we condense star-like peripheral structures, that commonly emerge in real networks. Our framework encapsulates existing enumeration based methods, speeding up their computation by taking advantage of operations enabled through the use of Condensed Graphs. Here we show how it could be coupled with both ESU [19] and FaSE [13] algorithms, but in principle it could be applicable with other enumeration methodologies.

With all of this in place, we achieved very promising results on representative sets of real world and synthetic networks, showcasing the applicability of our methodology. In terms of compression, we depend on the existence of star-like peripheries. Here we show that these are very common on real world networks, and we are able to reach up to 97% of compression. Regarding speedup, our experiments show that the gains can be substantial and that when the networks exhibit this kind of topology we are able to achieve exponential gains. We also try to quantify the desired structural properties that make networks more amenable to our proposed approach.

2 Backgroud

2.1 Notation

A graph $G = (V, E)$ is comprised by a set of edges $E(G)$ connecting vertices $V(G)$. A k-graph is a graph with k vertices. In this paper we will only address simple undirected graphs, with at most a single edge connecting the same pair of vertices. Our data structure could however straightforwardly be extended to support directed graphs, multigraphs or even more complicated graph representations.

For two graphs G and S such that $V(S) \subseteq V(G)$, then it is said that S is a *subgraph* of G. Moreover, if $\forall u, v \in V(S) : (u, v) \in E(G) \iff (u, v) \in E(S)$, then S is an *induced subgraph* of G. Two graphs, G and G', are said to be *isomorphic* if and only if there is a bijection $f : V(G) \longrightarrow V(G')$, such that $(u, v) \in E(G) \iff (f(u), f(v)) \in E(G')$. An *automorphism* is an isomorphism from a graph on to itself, and the automorphisms of a graph G form a group called $Aut(G)$. Consider a vertex $u \in V(G)$, then the *automorphism orbit* of u is:

$$Orb(u) = \{v \in V(G) | v = g(u), g \in Aut(G)\} \quad (1)$$

Simply put, if u and v are in the same orbit, they are topologically equivalent, which means one could swap their labels without altering the graph topology.

2.2 Problem Definition

In this paper we tackle the following computational problem:

Definition 1. ***(Subgraph Census Problem)*** *Given some positive integer k and a graph G, count the exact number of distinct occurrences of each of all possible connected induced k-subgraphs of G. Two occurences are distinct if there is at least one vertex that they do not share.*

2.3 Related Work

Subgraph census computation has been studied for more then 15 years. In 2002 Milo et al. [12] coined the term *network motifs* as frequent overrepresented induced subgraph patterns and offered the `mfinder` subgraph enumeration algorithm as a first practical approach for computing subgraph frequencies. The first major breakthrough was introduced by Wernicke [19] with the `ESU` algorithm, which avoided graph symmetries and enumerated each subgraph only once. Isomorphism tests for each discovered subgraph occurrence are made trough the third party package `nauty` [11], a highly efficient isomorphism algorithm. In order to reduce the number of needed isomorphism tests, approaches such as `QuateXelero` [9] or `FaSE` [13] encapsulate the topology of the current subgraph match, grouping several occurrences as belonging to the same isomorphic class. If we know beforehand the set of subgraphs that we are interested on (which can possibly be smaller than the entire set of all possible k-subgraphs), the `g-tries` data structure [17] could be used, allowing for further improvements.

All the aforementioned approaches are general (i.e, are applicable to any subgraph size and also allow direction) and rely on doing a full subgraph enumeration. However, for more specific sets of subgraphs there has been an increasing number of more analytical algorithms that take into account the subgraphs topology and its combinatorial effects. For example, `ORCA` [6], which counts orbits and not directly subgraph occurrences, can tackle up to size 5 undirected subgraphs and relies on a derived set of linear equations that relate the orbit counts. This was also generalized for other small undirected orbits [7]. `PGD` [1] (up to size 4) and `Escape` [14] (size 5) are other examples of state-of-the-art analytical algorithms specialized on counting undirected subgraphs.

Our approach differs from these two conceptual approaches, as it tries to combine the general applicability of the enumeration algorithms with combinatorial improvements. However, instead on focusing on the topology of the subgraphs we are looking for, we focus on how to compress the network we are analyzing, targeting specific substructures than can be resumed as a combinatorial object.

All the aforementioned algorithms perform exact computations, but it should be said that there are also methodologies that can trade accuracy for speed, providing approximate results. Furthermore, some algorithms exploit parallelism. For the purposes of this paper we pursue exact sequential census computation, as to improve the baseline algorithm, but our approach could be further extended on the future towards other directions. For a more detailed survey of the state of the art on subgraph counting we refer the reader to [16].

3 A Novel Framework for Subgraph Census

3.1 Peripheral Stars

Our methodology revolves around peripheral areas of a network, which are topologically self contained. This allow us to perform combinatorial calculations to quickly identify a larger number of occurrences of the same subgraph topology, avoiding the need to explicitly pass trough each single occurrence. As an initial proof of concept, here we will focus on *star subgraphs* on undirected networks, but we envision many other potential extensions to more complex peripheries.

Definition 2. ***(Peripheral Star Subgraphs)*** *An induced subgraph S of a graph G is said to be a* ***peripheral star*** *of size m if it is comprised by m vertices of degree one, called the* ***peripheral vertices****, that are connected only to the same vertex s, called the* ***seed vertex****.*

The terms peripheral star and star will henceforth be used interchangeably. An induced star within a peripheral star will be called a *substar*. Furthermore, we will use $P(G)$ to denote the set of all peripheral vertices on a graph G.

A peripheral star only has two orbits: the seed vertex orbit, and the peripheral orbit, This simple topology lies at the core of our speedup. Suppose you have a star of size m. Then, for all $i \in [1, m]$, we know that the number of i-substars is precisely $C_i^m = \binom{m}{i}$, all of them with the exact same isomorphic class. A visual example is given in Figure 1.

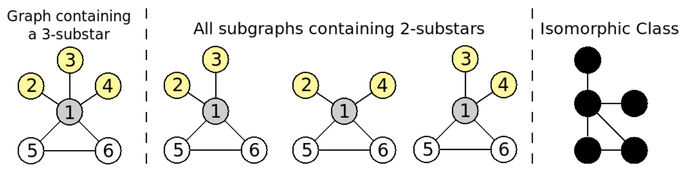
Graph containing a 3-substar
All subgraphs containing 2-substars
Isomorphic Class

Fig. 1. (Left) A graph containing a peripheral star (seed vertex: gray, peripheral vertices yellow, regular vertices: white). (Middle) All possible 2-substars (Right) The corresponding isomorphic class (the same for all subgraphs in the middle)

As the size m of the star, and the size k of the subgraph increase, the number of combinations $\binom{m}{i}$ increases exponentially, a property that we will exploit.

3.2 Condensed Graphs

The first step on our methodology is to compress the original graph, such that all peripheral star subgraphs are discovered and reduced to identifying its size. This process is exemplified in Figure 2. Let SQ_u be the number of peripheral vertices connected to a vertex u (which correspond to the numbers inside parenthesis in the figure). Condensing a graph can be thought as the process of eliminating all peripheral vertices and adding extra information to all other nodes in the form of SQ_u for all vertices u of the condensed graph.

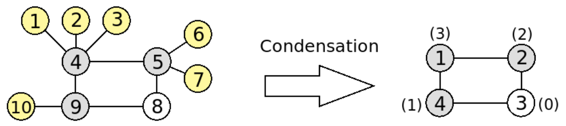

Fig. 2. (Left) A graph. (Right) the resulting Condensed Graph (seed vertices in gray).

We can trivially condense any graph in $\mathcal{O}(|V|+|E|)$ time by iterating through all vertices, reassigning labels to non-peripheral vertices based on the order they were visited. This is in fact a *lossless* compression scheme, since we still maintain all the original topological properties and we can easily decompress back to a graph isomorphic to the original one in time $\mathcal{O}(|V|+|E|)$, an operation we describe as *decondensation*.

3.3 Taking Advantage of Condensation

Classical enumeration algorithms try to explicitly traverse all subgraph occurrences, effectively increasing the frequency by one each time. The key point of our work is precisely to account for multiple occurrences at the same time, taking advantage of the combinatorial effects of self contained peripheries, avoiding the costly explicit traversal of all topologically equivalent substars.

Our framework is general and can be applied to any enumeration algorithm that builds subgraphs by adding one vertex at a time. Consider that we are performing a k-subgraph census and that we already have a partially enumerated vertex set $V_{subgraph}$ of size $d < k$, that we want to extend up to size k. When we add a seed vertex, we can consider all the possible substars that this new vertex may induce. Figure 3 exemplifies this concept. Condensed graphs' properties allow to proceed with the extension and simultaneously tracking multiple occurrences.

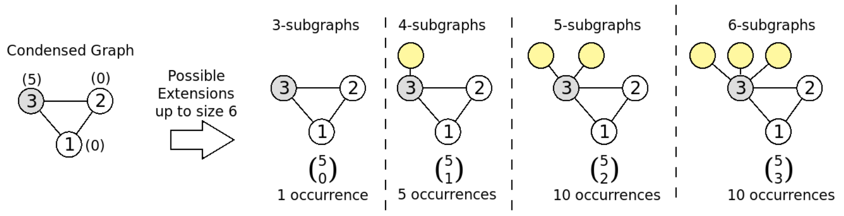

Fig. 3. Extending a condensed graph to subgraphs up to size 6.

With all of these concepts in place, we are now ready to explain our ***C**ondensation-**D**econdensation **F**ramework (**CFW**)*, that is able to improve an existing baseline subgraph census algorithm. An overview of our approach is given in Algorithm 1, which describes, in a k-census of graph G, how to extend any partially enumerated set of nodes $V_{subgraph}$, whose current frequency is given by $cur_{frequency}$.

To start the process, we should start by calling $extend\_subgraph(G, k, u, 1)$ for all nodes $u \in V(G)$, that is, we try to create a subgraph starting from every node.

Algorithm 1 Condensation-Decondensation Framework

```
1: procedure EXTEND_SUBGRAPH($G, k, V_{subgraph}, cur_{frequency}$)
2:     if $|V_{subgraph}| = k$ then
3:         $Frequency[V_{subgraph}]$ += $cur_{frequency}$
4:     else
5:         for all vertex $u$ extending $V_{subgraph}$ do        ▷ Using baseline enumeration algorithm
6:             for all $i \in \{0 \dots min(k - |V_{subgraph}| - 1, SQ_u)\}$ do
7:                 $V_{extended} \leftarrow V_{subgraph} \cup \{u\} \cup \{i$ peripheral nodes attached to $u\}$
8:                 extend_subgraph($G, k, V_{extended}, cur_{frequency} \times \binom{SQ_u}{i}$)
```

Now, for each node we add (line 5), we take into account all possible substars that extend up to size k (lines 6 and 7), and we are able to directly identify how many isomorphic occurrences of that particular substar can be obtained (line 8), as previously explained. Notice how we multiply by the current frequency, which allows to consider subgraphs that incorporate multiple substars from different seed vertices. The process stops when we reach the desired subgraph size (line 2), when we can safely increment the frequency by a value reflecting how many multiple isomorphic occurrences we are considering (line 3), as opposed to simply incrementing by one in the baseline enumeration algorithm.

3.4 Baseline Enumeration Algorithms

For the purposes of this paper we will be adapting two well known subgraph counting algorithms that fit into our framework: they explicitly enumerate all occurrences and they work by extending subgraphs one vertex at a time. Given the space constraints of this paper, we will only give a very high level description of the two algorithms, and we refer the reader to the respective original papers for more in-depth detail.

The first of these algorithms is `ESU` [19], which uses carefully chosen restrictions on the way it extends subgraphs, to guarantee that each set of k connected nodes is only enumerated once. Each of these occurrences is then run trough `nauty` [11], a very efficient third-party isomorphism algorithm, so that we identify the topological class for which we need to increment the frequency.

The second algorithm we adapted was `FaSE` [13], which improves the previous approach by avoiding the need of doing one isomorphic test per occurrence. In order to do that, while still using the same baseline enumeration procedure as ESU, it uses the *G-Trie* data structure [17], which can be briefly described as a trie of graphs. In this way, node sets that induce the exact same adjacency matrix will give origin to the same path in the g-trie, allowing us to group many occurrences as belonging to the same topological class. Due to naturally occurring symmetries in the subgraphs, several different paths on the g-trie may still correspond to the same topology, and we still need to identify this. However, we only need one isomorphism test per group of occurrences (a path in the g-trie), which allows for a substantial speedup when compared to the classical `ESU` algorithm.

4 Experimental Results

Our main goal is to compare the baseline algorithms `ESU` and `FaSE`, to the adaptations using our framework, respectively called `Co-ESU` and `Co-FaSE`. All experiments were done on a machine with a 2.4 GHz Intel i5 CPU, 8 GB 1600 MHz DDR3 RAM running macOS High Sierra 10.13.6. All algorithms were implemented in C++ using clang-902 as the compiler.

4.1 Real World Networks

We will first test the algorithms on a broad set of real networks from different backgrounds, which are described in Table 1. For the purposes of this paper we ignored both weights and direction. Thus, we transformed `econpoli` (directed) to an undirected network and ignored the weights in `rtobama`.

Name	**Type**	**Description**	$\|VG)\|$	$\|E(G)\|$	$\|P(G)\|$	CR	μ_{star}	$\max_{star}$	Source
`facebook`	Social	Friendships	2888	2981	2790	97%	279.0	756	[18]
`rtobama`	Social	Retweets	9631	9772	9104	93%	69.2	7413	[18]
`reality`	Social	Phone calls	6809	7680	6284	92%	77.6	233	[18]
`mvcortex`	Brain	Fiber tracts	194	214	160	82%	14.6	23	[18]
`econpoli`	Economic	Transactions	15575	17468	12187	74%	10.7	490	[18]
`genefusion`	Biological	Gene Interact.	291	279	203	56%	3.3	29	[10]
`gridworm`	Biological	Gene Interact.	3518	6531	1887	45%	6.6	323	[18]

Table 1. The set of used real networks, in decreasing order of compression ratio (CR).

For each network, the previous table reports the number of nodes ($|V(G)|$) and edges ($|E(G)|$), as well as the number of peripheral nodes ($|P(G)|$). To indicate the potential for speedups using our framework, we give an idea of much we are compressing the graph in the form of a compression ratio $CR = \frac{|P(G) \setminus P_1(G)|}{|V(G)|}$, where $P_1(G)$ are stars of size 1, which we disregard given that they are not combinatorially exploitable. Furthermore, we report the average size of stars larger than size 1 (μ_{star}) and the size of the largest star ($\max_{star}$).

Table 2 summarizes the experiments done with real networks, reporting the execution time (in seconds) of all algorithms, as well as the speedup of our adaptations when compared with the respective baseline algorithm. In each network, we show the results obtained for different subgraph sizes k.

The first major insight is that our adaptations are always quicker than their original counterparts for all non-trivial cases ($> 1s$), confirming we are indeed improving the baseline algorithms. Furthermore, our speedup tends to increase superlinearly with the size k in both algorithms, with more gains on the cases where the computation time is already higher. This is due to the fact that larger larger subgraphs will naturally correspond to an (exponentially) larger number of occurrences that we can combinatorially exploit. We also note that our speedup

Network	k	k-census execution time (s)		Speedup	k-census execution time (s)		Speedup
		ESU	Co-ESU		FaSE	Co-FaSE	
`facebook`	3	0.23	0.01	23.3x	0.04	0.01	3.8x
	4	44.91	0.07	641.6x	4.72	0.01	471.5x
	5	9720.98	1.08	9000.91x	1006.70	0.05	20133.9x
	6	$>$ 5 hours	13.11	N/A	$>$ 5 hours	0.31	N/A
`rtobama`	3	19.96	0.19	105.07x	2.058	0.1	20.6x
	4	$>$ 5 hours	2.98	N/A	8439.36	0.23	36692.9x
	5	$>$ 5 hours	254.79	N/A	$>$ 5 hours	7.62	N/A
`reality`	3	0.14	0.09	1.54x	0.02	0.05	0.39x
	4	7.60	0.58	13.09x	1.00	0.09	11.11x
	5	451.61	10.08	44.80x	43.72	0.53	82.5x
	6	$>$ 5 hours	244.19	N/A	2307.81	12.48	184.9x
`mvcortex`	7	1.69	0.1	16.94x	0.50	0.01	49.6x
	8	16.63	0.79	21.06x	6.96	0.05	139.2x
	9	120.36	3.67	32.79x	65.80	0.22	299.1x
	10	933.74	18.27	51.11x	662.87	0.83	798.6x
	11	6340.48	51.91	122.14x	4067.36	3.47	1172.2x
`econpoli`	3	0.07	0.47	0.16x	0.04	0.34	0.1x
	4	2.62	3.2	0.82x	2.23	0.53	4.2x
	5	165.15	66.58	2.48x	130.04	2.61	49.8x
`genefusion`	8	2.67	0.45	5.92x	1.19	0.04	29.7x
	9	21.45	1.93	11.11x	6.85	0.16	42.8x
	10	81.85	7.72	10.60x	38.60	0.67	57.6x
	11	209.43	29.18	7.18x	211.60	1.73	122.3x
`gridworm`	4	28.63	7.51	3.81x	1.25	0.38	3.3x
	5	5307.49	391.71	13.55x	125.64	16.94	7.4x
	6	$>$ 5 hours	$>$ 5 hours	N/A	$>$ 5 hours	16726.42	N/A

Table 2. Speedup of our adaptations vs baseline algorithms on real networks.

is typically higher with `FaSE`, which is already substantially faster than the `ESU` algorithm. We suspect this might be caused due to synergies between our condensation-decondensation operation and the way `FaSE` operates, which might result in smaller g-tries and less isomorphic tests needed.

For the two top performing networks `facebook` and `rtobama`, we focus on `FaSE` and `Co-FaSE`, since `ESU` did not perform fast enough in our time constraints to draw significant conclusions. Although `facebook` has higher CR, both show evidence of exponential speedup. However, in `rtobama` speedup seems to grow faster, reaching 4 orders of magnitude for $k = 2$ while `facebook` only matches this results for $k = 3$. Going into further detail, even the precise values of speedup favor `rtobama`, (36692.87x *versus* 20133.9x). Note that the $\max_{star}$ in `facebook` is 756, while in `rtobama` is 7413, accounting for $\approx 26\%$ and $\approx 77\%$ of the total networks' sizes respectively. Moreover, $\binom{n}{k}$ scales exponentially with n

with regards to k. Although μ_{star} is higher for `facebook` the difference in size of $\max_{star}$ completely overshadows the impact of the former metric.

The next pair of networks analysed `reality` and `mvcortex`, that have a CR disparity of 10% between them. Regarding the former, in both comparisons, speedup seems to increase in a linear fashion, with 1 order of magnitude of speedup improvement measured for `Co-ESU` for $k = 5$, and 2 orders for `Co-FaSE` for $k = 4$. Addressing the latter network, speedup in both cases grows in a linear fashion with k, but in different orders of magnitude. In the case of `Co-ESU`, we measured up to 2 order of magnitude. In `Co-FaSE`, the results are more dramatic, reaching 3 orders of magnitude. We suspect that, `mvcortex` showed considerably better results than `reality`, even with less CR, because we were able to measure values of k that were closer to the optimum value of $\binom{\max_{star}}{k}$ and by extension, took full advantage for smaller stars.

Focusing on `econpoli` and `genefusion` they differ 18% in CR. For the former network, due to the size discrepancy among them and our hardware and time limitations, this led to a smaller number of observations. The consequences in speedup remain very similar for both adaptations, with a spike for $k = 7$, that we once attribute to the order of growth of maxima of $\binom{n}{k}$. Addressing `genefusion`, we were able to draw measure performance up to $k = 11$. Keep in mind that $\operatorname{argmax}_k \binom{\max_{star}}{k} = 15$ and, as theory predicts concerning `Co-FaSE`, speedup grows linearly steady up to $k = 10$, but for $k = 11$ it almost doubles, since it is a point of ramp-up for the gradient of $\binom{n}{k}$. Surprisingly, this was not observed for `ESU`, and we do not yet have any credible theory regarding this phenomenon.

Finally, our last and worst performing network, `gridworm`, that has 11% less condensation ratio than `genefusion`. Its $\max_{star}$ accounts for $\approx 11\%$ of the overall network. However, due to its size and complexity, we were only been able to measure speedup for a small range of k. Surprisingly, it is one of the few examples (along with `econpoli`, but much more drastic), that benefits `ESU` the most which is improved by one order of magnitude $k = 5$. For the same k, `Co-FaSE` follows behind closely measuring $7.4x$ speedup.

4.2 Synthetic Networks

To gain more insight into the specific properties that benefit our approach, we follow the same experimentation workflow, but for synthetic networks, generated using the `NetworkX` package [5]. We considered the following network models:

Barabási-Albert [2] This model generates scale-free networks, who emerge in a plethora of phenomena in the real world, using *preferential attachment* as its connection mechanism. We will refer to it as `BA`(n, m), with n being the number of nodes and m the number of initial edges on each newly added vertex

Holmes-Kim [8] This model extends the `BA` model to produce networks with an higher clustering coefficient: after an edge is created between the newly added vertex v and another vertex w, a random neighbour of w is selected and an edge between it and v is created with probability p, thus forming a triangle between these three vertices. The alias for this model will be `HK`(n, m, p).

Random Power-Law Trees [4] This model generates trees with a power law degree distribution. The model is too intricate to summarize, but essentially `NetworkX`'s implementation takes three parameters, n the size of graph, γ the exponent of the power-law and $tries$, the number of tries necessary to ensure the degree sequence forms a tree. The alias for this model will be $\texttt{PLTrees}(n, \gamma, tries)$.

To make comparisons fair, we generated all networks with 1000 vertices. Table 3 gives an overview of the used synthetic networks, including the model parameters and the topological characteristics of the generated networks.

Model	$\|V G)\|$	$\|E(G)\|$	$\|P(G)\|$	CR	μ_{star}	$\max_{star}$
$\texttt{BA}(1000, 1)$	1000	999	686	53%	4.27	49
$\texttt{HK}(1000, 1, 0.9)$	1000	999	677	53%	4.05	44
$\texttt{PLTrees}(1000, 3, 100000)$	1000	1052	526	45%	3.17	54

Table 3. The set of used synthetic networks generated using `NetworkX` package.

We purposefully chose an high p parameter in `HK`, to see how well our framework would work on a scale-free network with high average clustering coefficient. Note that for for both `BA` and `HK` the m parameter is 1, since its a necessary condition for emergence of peripheries. Regarding `PLTrees` , the γ is set to 3 by default to result in a scale-free network.

To avoid visual clutter, we will not include the model parameters in Table 4, and they will be referred simply by `BF`, `HK` and `PLTrees`

Network	k	k-census execution time (s)		Speedup	k-census execution time (s)		Speedup
		ESU	Co-ESU		FaSE	Co-FaSE	
`BA`	5	2.14	0.36	5.96x	0.17	0.03	5.6x
	6	53.68	5.98	8.98x	3.6	0.21	17.2x
	7	1084.62	120.31	9.02x	118.87	2.88	41.3x
	8	11756.72	1607.72	7.31x	4621.29	31.03	150.0x
`HK`	5	3.47	0.39	8.91x	0.28	0.02	14.2x
	6	44.86	7.76	5.78x	44.86	0.23	195.0x
	7	1027.72	232.21	4.43x	106.52	2.86	37.2x
	8	8413.58	2821.62	2.98x	2406.00	42.5	56.6x
`PLTrees`	6	3.25	0.02	162.70x	0.20	< 0.01	N/A
	7	29.85	0.05	596.98x	2.40	0.01	240.4x
	8	264.26	0.16	1651.63x	19.90	0.02	995.2x
	9	1569.30	0.51	3077.07x	213.22	0.05	4264.5x
	10	11843.88	2.4	4934.95x	2666.02	0.25	10664.1x

Table 4. Speedup of our adaptations vs baseline algorithms on synthetic networks.

Concerning `BA`, `FaSE` clearly benefits from our framework, showing evidence of superlinear speedup, up to 2 orders of magnitude of improval. In the case of

ESU, a slight increase from $k = 5$ up to $k = 7$ was measured. In $k = 8$ the trend shifts in the opposite direction. We suspect that, for larger values of k, a similar pattern would occur, with an average of 1 order of magnitude of improvement with slight shifts in the trend of speedup. Note that has $k \rightarrow \left(\lceil \frac{\max_{star}}{2} \rceil = 500\right)$ the results can change drastically.

Addressing HK, the measurements are relatively similar to the ones observed in BA. Once again, FaSE benefits the most from our implementation. In this case, the trend does not appear to be strictly monotone. We suspect it will vary between 1 and 2 orders of magnitude of speedup as k grows, and then an upwards shift improvement when when the gradient ramps up as k approaches 500. Unfortunately, we are limited once again by our hardware and time constraints to make an concrete comparison.

PLTrees display our best results on synthetic data. ESU shows evidence of a non-linear relationship regarding speedup, reaching 3 orders of magnitude of improvement. Although speedup seems to grow more slowly, between $k = 6$ and 8. Concerning FaSe, it is once again favoured, and shows evidence of superlinear speedup, reaching 4 orders of magnitude of performance improvement.

5 Conclusion

The goal of this paper was to build a generic framework adaptable to current subgraph census algorithms, from which we selected and effectively adapted ESU and FaSE. We have experimentally shown that our adaptations are significantly faster for a diverse set of networks extracted from different contexts. The framework enhanced significantly both algorithms on our experimen , up to 4 orders of magnitude speedup for both Co-ESU and Co-Fase, with indications of exponential speedup for our best cases. Note also that Co-ESU does not uses g-tries, and it still outperformed FaSE in all networks except gridworm, which further outlines the potential of our approach.

The condensation ratio of a network is highly correlated with performance, but does not fulfill a causal relationship. We refer back to to the properties of the binomial the function that, coupled with the size of the largest star, affect speedup drastically. Note that $\max_{star}$ only gives a lower bound insight for potential speedup, since it does not account for the remaining smaller stars. From this, it is easy to see why networks with higher condensation ratio like reality are outperformed by others with less condensation like mvcortex, since we are able to explore values of k close to the maximum of for its $\max_{star}$.

On our set of synthetic networks, we observed that our framework improves performance on scale-free networks, that are very recurrent on a plethora of real world phenomena, with the best case being the PLTrees.

The results are very promising and indicate this is a viable path for improving existing enumeration algorithms without losing generality. For the close future, we intend to tackle other types peripheries and to extend our approach to more complex networks, including aspects such as edge direction, temporal information and multiple layers of connectivity.

Acknowledgements: This work is financed by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia within project : UID/EEA/50014/2019.

References

1. Ahmed, N.K., Neville, J., Rossi, R.A., Duffield, N.: Efficient graphlet counting for large networks. In: Int. Conference on Data Mining. pp. 1–10. IEEE (2015)
2. Barabasi, A.L., Albert, R.: Emergence of scaling in random networks. Science 286(5439), 509–512 (1999)
3. Cook, S.A.: The complexity of theorem-proving procedures. In: ACM symposium on Theory of computing. pp. 151–158. STOC '71, ACM (1971)
4. Gao, Y.: The degree distribution of random k-trees. Theoretical Computer Science 410, 688–695 (2009)
5. Hagberg, A., Schult, D., Swart, P., Conway, D., Séguin-Charbonneau, L., Ellison, C., Edwards, B., Torrents, J.: Networkx. high productivity software for complex networks. Webová strá nka https://networkx. lanl. gov/wiki (2013)
6. Hočevar, T., Demšar, J.: A combinatorial approach to graphlet counting. Bioinformatics 30(4), 559–565 (2014)
7. Hočevar, T., Demšar, J.: Combinatorial algorithm for counting small induced graphs and orbits. PloS one 12(2), e0171428 (2017)
8. Holme, P., Kim, B.J.: Growing scale-free networks with tunable clustering. Physical review E 65(2), 026107 (2002)
9. Khakabimamaghani, S., Sharafuddin, I., Dichter, N., Koch, I., Masoudi-Nejad, A.: Quatexelero: an accelerated exact network motif detection algorithm. PloS one 8(7), e68073 (2013)
10. Kunegis, J.: Konect: the koblenz network collection. In: Proceedings of the 22nd International Conference on World Wide Web. pp. 1343–1350. ACM (2013)
11. McKay, B.D.: nauty user's guide (version 2.2). Tech. rep., Technical Report TR-CS-9002, Australian National University (2003)
12. 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)
13. Paredes, P., Ribeiro, P.: Towards a faster network-centric subgraph census. In: International Conference on Advances in Social Networks Analysis and Mining. pp. 264–271. IEEE (2013)
14. Pinar, A., Seshadhri, C., Vishal, V.: Escape: Efficiently counting all 5-vertex subgraphs. In: International Conference on World Wide Web. pp. 1431–1440. International World Wide Web Conferences Steering Committee (2017)
15. Pržulj, N.: Biological network comparison using graphlet degree distribution. Bioinformatics 23, e177–e183 (Jan 2007)
16. Ribeiro, P., Paredes, P., Silva, M.E., Aparicio, D., Silva, F.: A survey on subgraph counting: Concepts, algorithms and applications to network motifs and graphlets. arXiv preprint arXiv:1910.13011 (2019)
17. Ribeiro, P., Silva, F.: G-tries: a data structure for storing and finding subgraphs. Data Mining and Knowledge Discovery 28, 337–377 (March 2014)
18. Rossi, R.A., Ahmed, N.K.: The network data repository with interactive graph analytics and visualization. In: AAAI (2015), http://networkrepository.com
19. Wernicke, S.: Efficient detection of network motifs. IEEE/ACM Transactions on Computational Biology and Bioinformatics 3(4), 347–359 (2006)