NECTAR - Technical Report

Yehonatan Cohen, Danny Hendler, Amir Rubin
Computer Science Department, Ben-Gurion University of the Negev, Be'er-Sheva, Israel
{yehonatc,hendlerd,amirrub}@cs.bgu.ac.il

This report contains technical details and a full competative analysis of the NECTAR algorithm.

In order to estimate the performance of our method, we conducted several experiments. Each of the graphs below emphasizes an aspect of our algorithm. This report contains a comparison of results on real world networks with ground truth, as well as on synthetic networks with different characteristics. A run-time comparison is included over two sorts of networks, as well as a comparison to the AGM-fit method, in the methodology taken by AGM-fit algorithm developers.

I. Real world networks

The DBLP and Amazon networks contain information about their nodes such as a node's category (Amazon) or the journal in which it was published (DBLP). This gives us a functional definition of ground-truth communities in those networks. In [1] these networks, and many more, along with their ground-truth communities, are examined. In particular, they discuss the quality of those communities using several metrics, e.g. modularity and conductance, and propose a ranking method for these ground-truth communities. In our experiments, after applying the community detection algorithms, we used the metrics discussed above to estimate the quality of the cover we got. In the case of both Amazon and DBLP, which has ground-truth communities, we used the top 5000 communities from [2] (without duplications) for our comparison.

The fact that we reduce the amount of communities in the ground-truth, may harm the estimating method - Average-F1, NMI and Omega-Index, as will be explained.

Omega-Index counts the amount of shared communities for any couple of nodes. A drawback of Omega-index is that if in one partitioning we have much less communities than in the other, the score will be low, as the chances for a couples of nodes to share the exact same amount of communities is low.

On the other hand, both NMI and Average F1 suffer from the following issue:

If C_1 includes much more communities than C_2, we will easily reach 0.5, a high score as a starting point. As presented in [3], the NMI as we use it will easily get a score around 0.5.

For the case of average-F1 score, we experience the same result. When we iterate over C_2 - the small collection, with high probability we could find a community in C_1 - the bigger collection, which is similar to it. Again reaching around 1, and in total 0.5, as we take the average.

Putting these together (the fact that we want to use a subset of all ground-truth communities, and the damage this may cause to our metrics), it is desired to find a more appropriate method to choose a subset from the given partitioning (which we have as an output from an algorithm) in order to compare it to the ground-truth.

For each ground-truth community, we take from our collection the community which fits it the most. Using the logic of the Average-F1 score, we take for each community c_1 in the ground-truth $argmax\{F_1(c_1, c_i) : c_i \in C\}$). So in Amazon for instance, if we had 1517 communities, we chose for each algorithm output at most 1517 communities which fits best the communities in the given ground-truth.

In this framework, the amount of the resulting communities of an algorithm may be a crucial factor in our estimation of a partitioning quality. The number of communities each algorithm has returned is summarised in table I. In both cases our method returned a reasonable amount of communities in comparison to other algorithms.

Algorithm	Amazon	DBLP
NECTAR	$23,863$	$44,523$
GCE	$25,962$	$22,657$
C-Finder	$28,402$	$27,075$
COPRA	$76,170$	$52,132$
SLPA	$23,049$	$24,405$
Info-Map	$21,613$	$21,472$
OSLOM	$17,007$	$17,526$

TABLE I: amount of resulting communities.

As can be seen in figures 1 and 2, in Amazon, our method yields the best results over the three metrics used (NMI, Average-F1 and Omega-Index). In DBLP C-Finder wins by a whisker, thanks to a higher NMI value. Both Average-F1 and Omega-Index are almost the same. The rest of the method are significantly inferior on this network.

II. Amazon best by F1

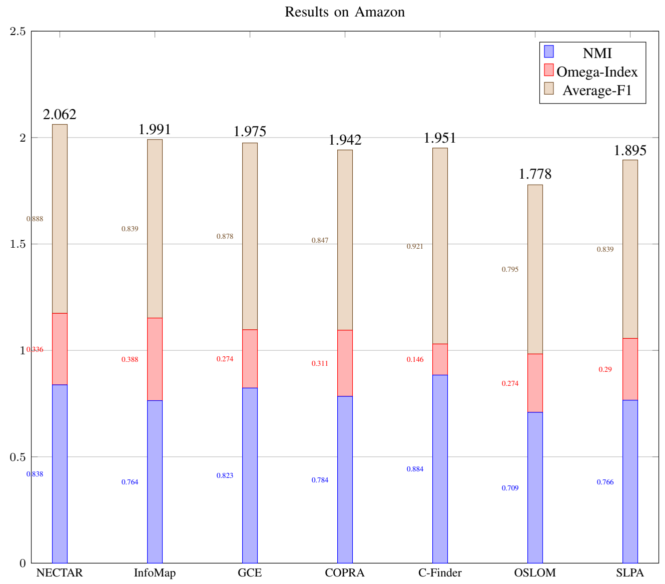

Fig. 1: Amazon F1

III. DBLP - best by F1

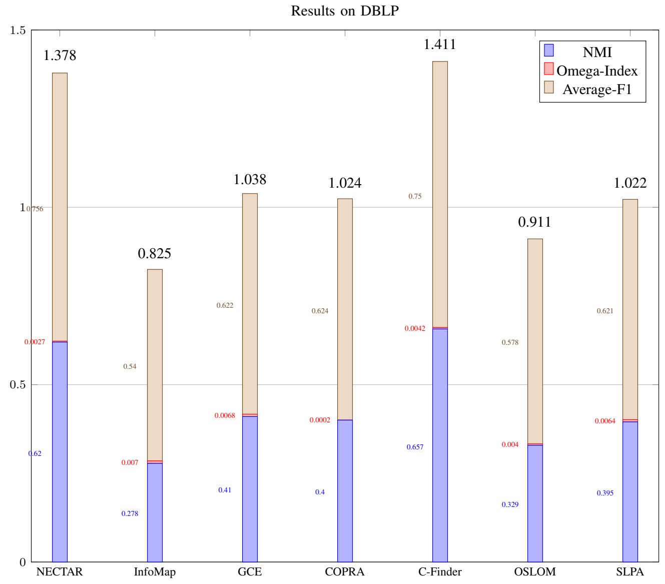

Fig. 2: DBLP F1

IV. Synthetic graphs - overview

The following parameters were used for generating LFR graphs [4]:

Note that our experimental setup is almost identical to the setup used in [5]. We fixed n, number of nodes, on 5000. The average node degree, k, is set to 10 or 40 and O_n (number of overlapping nodes) is 10% or 50% of the nodes, respectively. O_m defines the amount of communities an overlapping node belongs to, was set between 2 and 8. T_1 (minus exponent for the degree sequence) is set to 2, and T_2 (minus exponent for the community size distribution) is 1. Maximum degree is 50. The mixing parameter is the average fraction of neighboring nodes of a node that do not belong to any community that the benchmark node belongs to. This parameter controls the fraction of edges that are between communities, and is set to be 0.3. As for the communities sizes, we have used two settings - big communities, where the sizes various between 20 to 100, and small communities, $10 - 50$. As done in [5], we generated 10 instances for each combination of parameters. We took the average of the results for each algorithm and each metric over the 10 instances.

Parameters for the algorithms:
SLPA - we get by default 11 runs with the parameter r in the range 0.01 to 0.5.
C-Finder - by default, the algorithm works with seeds in sizes from 3 to the size of the biggest clique found in the graph.
OSLOM -runs 10 times, taking the result with the maximum Modularity.
COPRA - we used $v \in \{1, 2, \ldots, 10\}$ repeating each run 10 times and taking the one with the maximum Modularity.
GCE - we used $k \in \{3, \ldots, 8\}$.
NECTAR - we used 20 different values of β in the range 1.01 to 4.5. We set $\alpha = 0.8$.
For real world networks we also used $\{4.6, 4.7, 4.8, 4.9, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 15.0, 20.0\}$.
We set $\alpha = 0.8$.
InfoMap - we use it with the flag indicating overlapping communities.
AGM failed to finish on several graphs after 128 hours, and therefor is not taken into account.

Since all algorithms (except InfoMap) have a user-defined parameter, we took the one that gave the best average result for each instance and each metric, to make the comparison as reliable as possible.

The experiments were conducted on a 24 core Linux machine with 128GB of memory.

Figure 3 presents a comparison of the algorithms' scores on synthetic graphs with big communities. Figure 3 illustrates the corresponding statistics with respect to synthetic graphs with small communities.

SLPA and COPRA reached less than 0.2 in total, and therefor aren't presented.

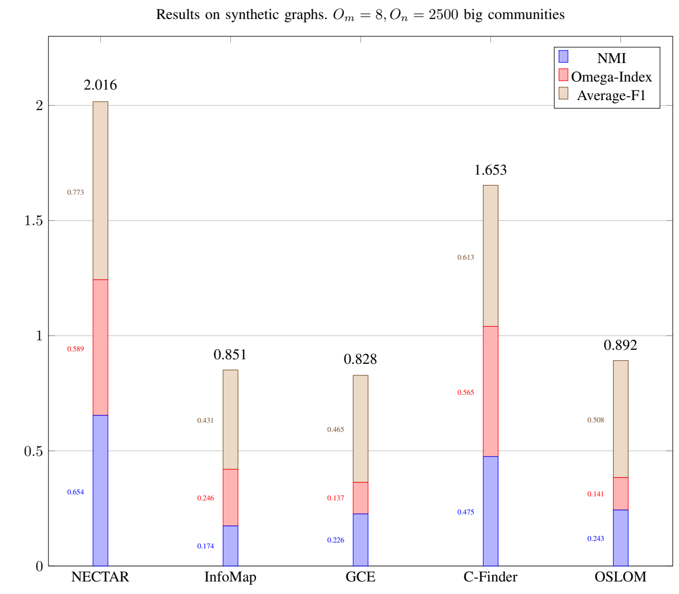

Fig. 3: Benchmarks summery. Algorithms results on graphs with big communities, $O_m = 8$ and $O_n = 2500$.

We choose to examine the graphs with the highest overlap level and the highest amount of communities per node as we believe this is an indication for the algorithm's performance in the most extreme conditions. When dealing with small communities our method is second after C-Finder, again, with a small gap - this time only due to the Omega-Index metric. Notice that NECTAR leads in both NMI and Average-F1.

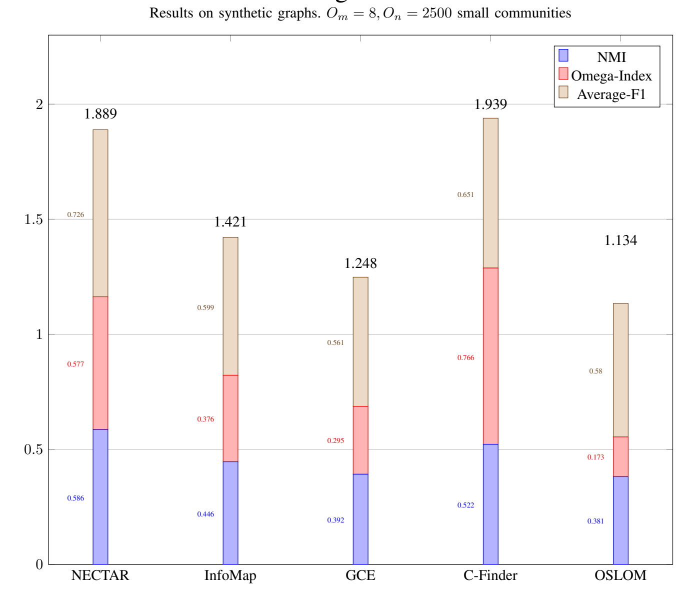

Fig. 4: Benchmarks summery. Algorithms results on graphs with small communities, $O_m = 8$ and $O_n = 2500$.

V. Synthetic Graphs - Full Comparison

In the following three sections we give the full comparison done on synthetic networks using three metrics: NMI, Omega-Index, and Average-F1.

In all three sections, the bottom two graphs are of networks with low overlapping rate (10% of the nodes), meaning $O_n = 500$. In all of these NECTAR maximizes Modularity as the triangles rate is lower than 5.

We want to focus the discussion on the top two graphs, where the overlap rate is 50% ($O_n = 2500$), and therefor conditions are harder for community detection. In these $WOCC$ is maximized. In the left side we have networks with big communities ($20 - 100$ nodes in a community), where our method outperform all others in NMI and Average-F1, and is similar to C-Finder in Omega-Index.

On the right we have networks with small communities ($10 - 50$ nodes in a community). Here C-Finder takes the lead in terms of Omega-Index, and is also leading in most cases in NMI and Average-F1. However, when focusing on the performance of the algorithms on networks with greater overlapping, e.g. higher O_m, the gap closes, and our method takes the lead in NMI and Average-F1 when $O_m = 8$. We are encouraged by that fact to believe that our method can give good results on complicated networks with high overlap rate.

VI. NMI - Synthetic graphs

A comparison between the NMI scores reached using the algorithms is summarised in figure 5 for graphs with average degree 40, and in figure 6 for graphs with average degree 10.

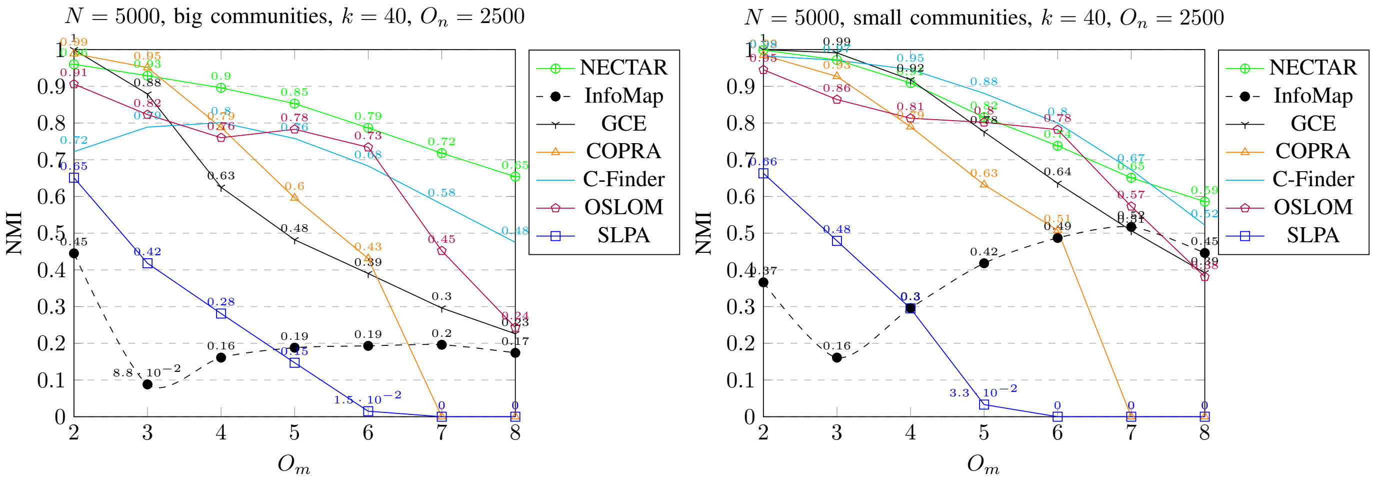

Fig. 5: NMI

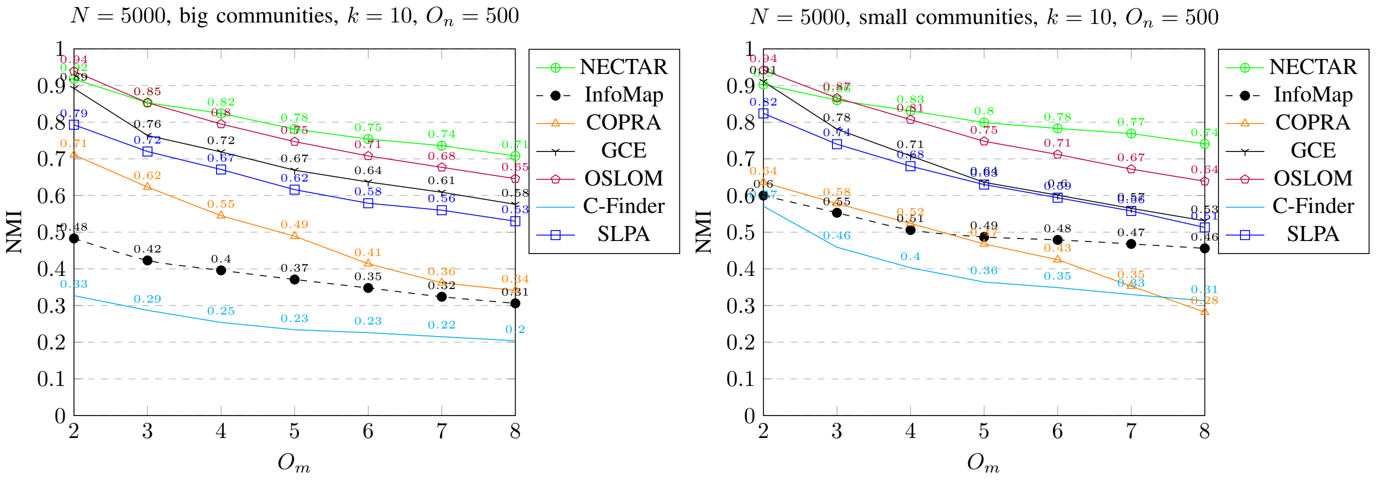

Fig. 6: NMI-10

VII. Omega-Index - Synthetic graphs

A comparison between the Omega-Index scores reached using the algorithms is summarised in figure 7 for graphs with average degree 40, and in figure 8 for graphs with average degree 10.

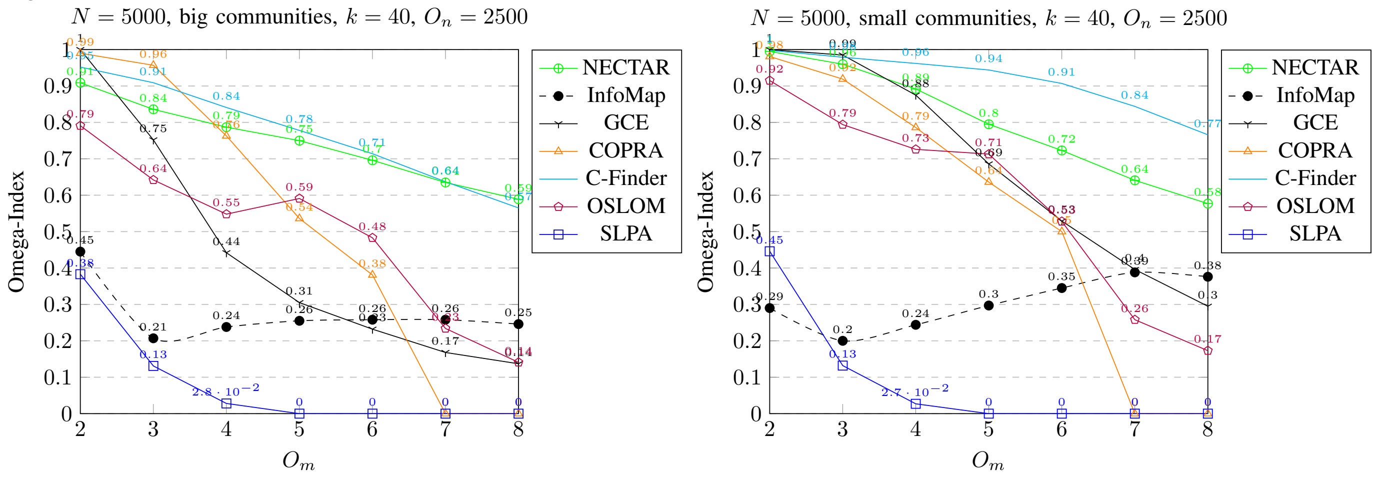

Fig. 7: Omega

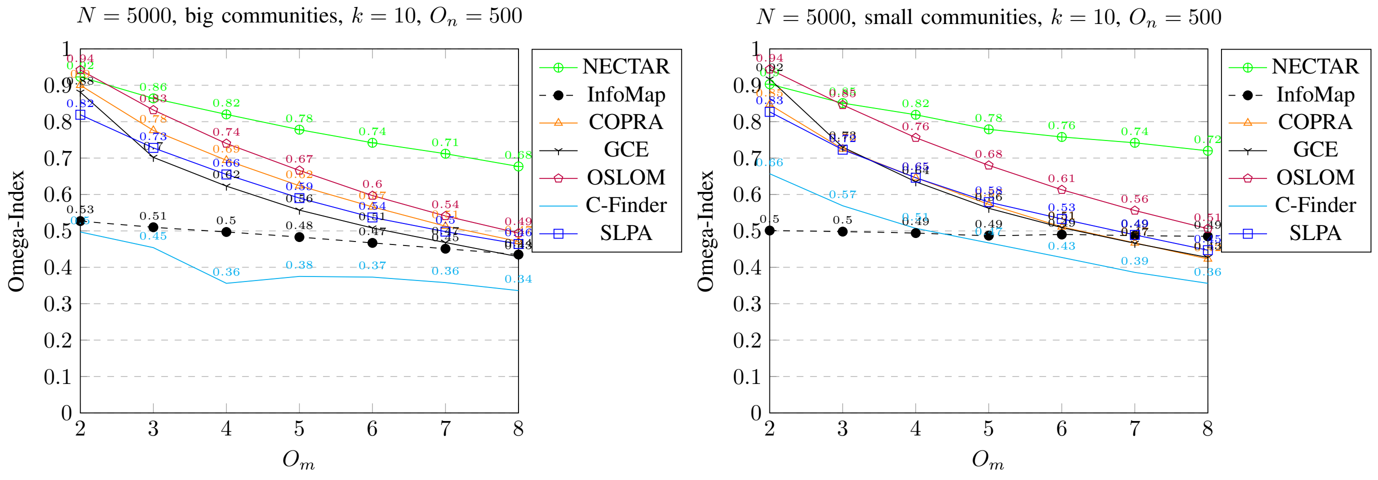

Fig. 8: Omega-10

VIII. Average-F1 - Synthetic graphs

A comparison between the Average-F1 scores reached using the algorithms is summarised in figure 9 for graphs with average
degree 40, and in figure 10 for graphs with average degree 10.

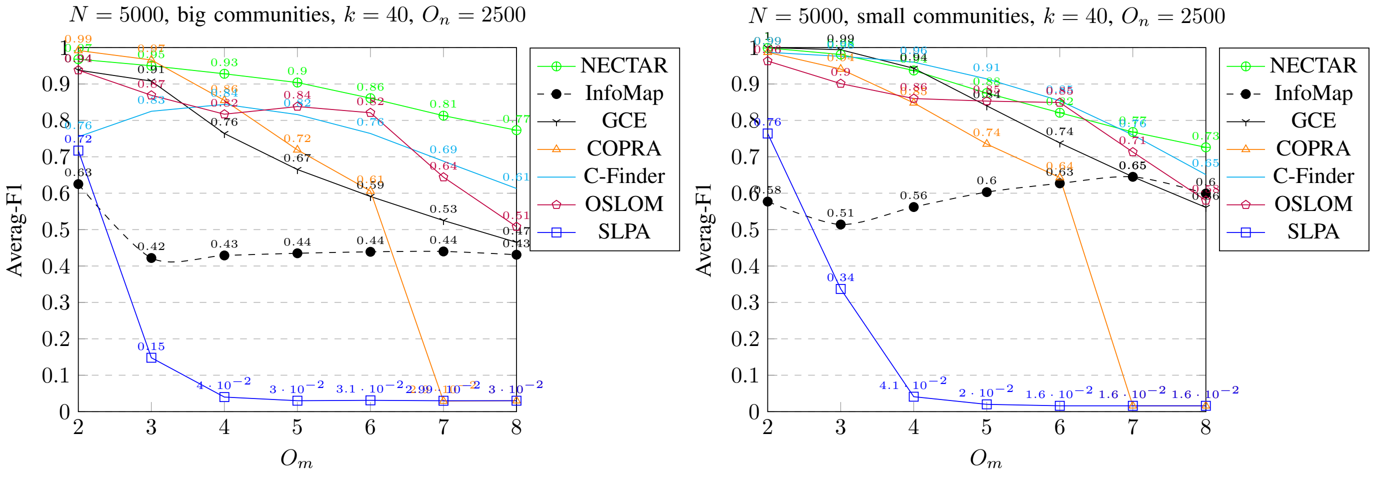

Fig. 9: F1

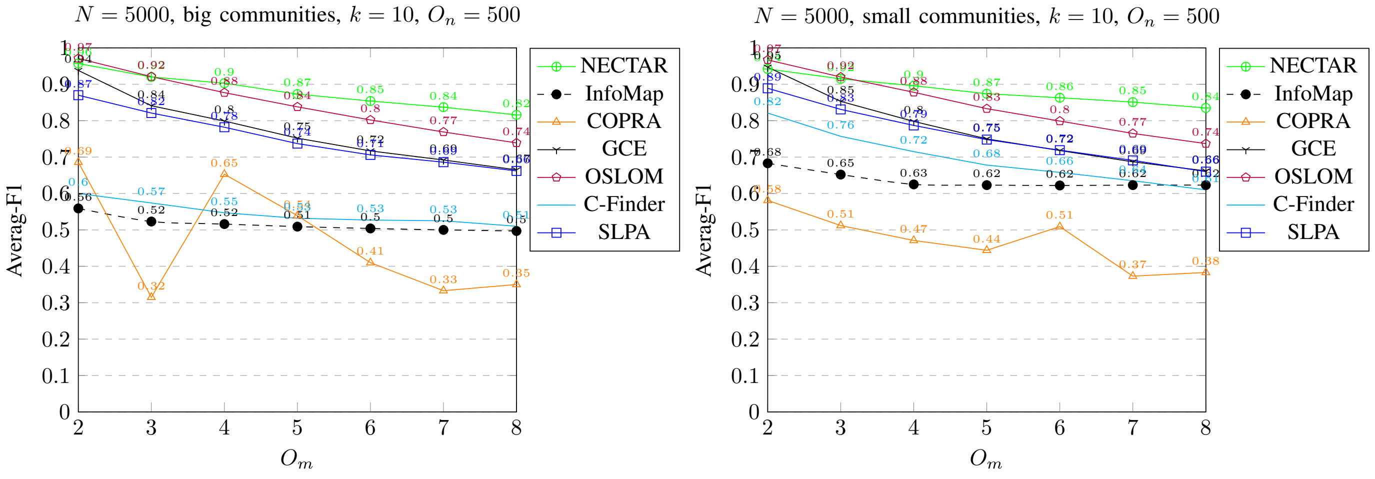

Fig. 10: F1-10

IX. Sub-networks community detection challenge

In AGM-fit [6] communities are modeled in a network using a Community-Affiliation Graph Model(AGM). AGM is used to generate graphs with overlapping communities. The community detection procedure on a given network is achieved by fitting the AGM - i.e. finding the AGM which has the highest probability of generating the given network. Communities on the networks can be extract from the resulting AGM.

As AGM failed to finish its run on the full networks we use, we follow the test methodology used in **??** to estimate it, and compare its results to our method. We used the DBLP network in our comparison, where AGM outperformed all other algorithms by the biggest gap, as can be seen in [6].

60 sub-networks of DBLP were used as input, generated in the following way:
we sampled 60 nodes, 10 of them are in exactly 2 communities, 10 are in exactly 3 communities, 10 in exactly 4, 10 in exactly 5 and 10 are at least in 6 communities. Looking at the communities the original node is a part of, all nodes members in those communities, as well as all edges between them, were used to build the sub-network. We limited the amount of nodes in a network to a maximum of 200 times the amount of communities used to generate it. So if the node we started with was in 4 communities, we allowed a maximum of 800 nodes in the sub-network. The purpose of this limitation is to avoid situations where huge communities are chosen.

Another limitation we added is the overlap rate between two communities used, by any of the sub-graphs to a maximum of 50%. This limitation is made to ensure that we have different sub-graphs, as we saw that many of the communities in the ground-truth are almost identical.

Our method outperform A, as shown in figure 11. As a side note - only 9 of the sub-networks had a triangle rate slightly larger than 5, scattered between the different O_m values, so in most cases Modularity was maximized by NECTAR.

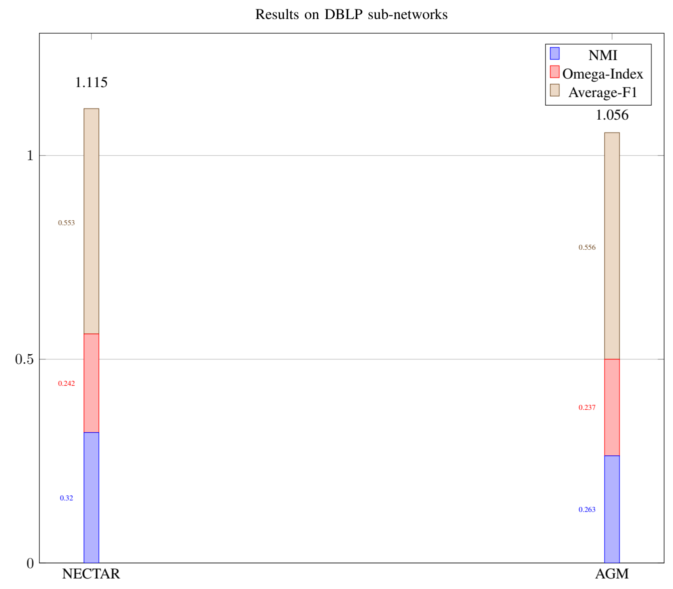

Fig. 11: sub-networks comparison

X. Run time comparison

In order to estimate our method run-time, we conducted a competitive analysis between 8 algorithms, taking into account two kinds of networks:
a. In figure 12 we have networks with N (number of nodes) between $1,000$ and $50,000$, and k (average node degree) is set to 10.
b. In figure 13 we have networks with k between 10 and 80, and N is set to $5,000$.
The other parameters used to generate the networks are set as follows: O_n (number of overlapping nodes) is 10% of the nodes. O_m defines the amount of communities an overlapping node belongs to, was set to 5. T_1 (minus exponent for the degree sequence) is set to 2, and T_2 (minus exponent for the community size distribution) is 1. Maximum node degree is 50, and the mixing parameter, which is the expected percentage of a nodes edges which connects it to non-relevant nodes is 0.3. As for the communities sizes, we used small communities, where the sizes various between 10 to 50. We generated 2 instances for each combination, and took the average run time. The experiments were conducted on a 24 core Linux machine with 128GB of memory.

As can be seen in the figures, overall our method run-time is of the same scale as others.
We estimate it with two objective functions: Q^E and $WOCC$, regardless of the triangles rate. In the figures the algorithms are denoted by NECTAR-M and NECTAR-W respectively. In details, when N grows NECTAR-M has performance as good as the others, while in NECTAR-W it seems that the situation is not as good. However, this is an encouraging hint - the problematic run-time of NECTAR-W is not a result of the node-centric search approach, but may be hidden in the requirements of calculating $WOCC$, or in our implementation. To better understand this we add NECTAR-W* to the graph, which is the run time of the NECTAR-W algorithm without

the initialization step. In the case of large amount of nodes, the most expensive part of the algorithm run is the initialization, which may be partly due to our implementation. In all of these graphs the triangles rate was smaller than 5, therefor $WOCC$ was in used by NECTAR.

When k grows we see again that NECTAR-M performances are good. NECTAR-W in this case is not so far from most algorithms, and has a reasonable growing rate. Except for $k = 10$, in these graphs the triangles rate was larger than 5, therefor $WOCC$ was in used by NECTAR.

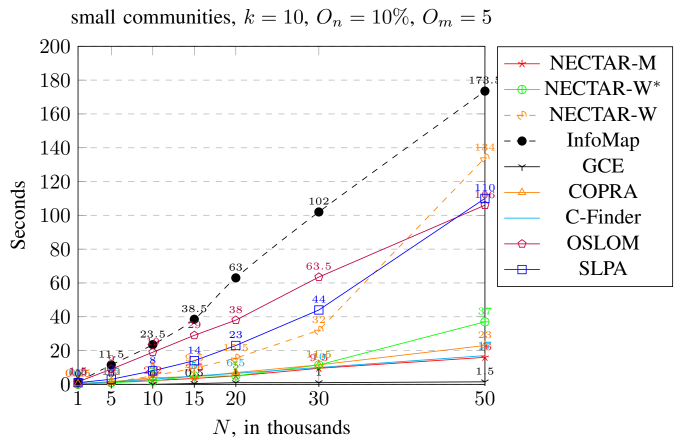

Fig. 12: Run time comparison

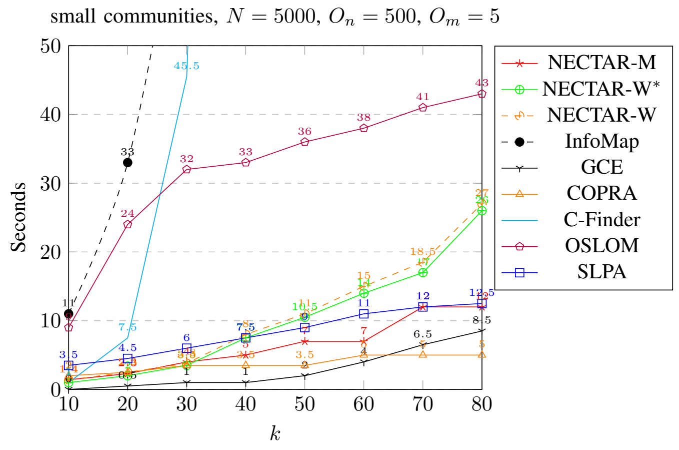

Fig. 13: Run time comparison

XI. Tool-box

NECTAR implementation can be found at https://github.com/amirubin87/NECTAR/tree/master/V2/
LFR (synthetic graph generator) implementation - https://sites.google.com/site/andrealancichinetti/files/

Implementation of the algorithms we used can be found here:
OSLOM - http://oslom.org/software.htm
SLPA - https://sites.google.com/site/communitydetectionslpa/
GCE - https://sites.google.com/site/greedycliqueexpansion/
C-Finder - http://hal.elte.hu/cfinder/wiki/?n=Main.Software
COPRA - http://www.cs.bris.ac.uk/~steve/networks/software/copra.html
Info-map - http://www.mapequation.org/code.html
AGM-fit - http://snap.stanford.edu/agm/

Sources for the metrics we used:
NMI - https://github.com/aaronmcdaid/Overlapping-NMI
Average-F1 and Omega-Index - https://github.com/amirubin87/NECTAR/tree/master/Tools (our implementation)

References

[1] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," *Knowledge and Information Systems*, vol. 42, no. 1, pp. 181–213, 2015.

[2] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection," http://snap.stanford.edu/data, Jun. 2014.

[3] A. F. McDaid, D. Greene, and N. Hurley, "Normalized mutual information to evaluate overlapping community finding algorithms," *arXiv preprint arXiv:1110.2515*, 2011.

[4] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Physical review E*, vol. 78, no. 4, p. 046110, 2008.

[5] J. Xie, S. Kelley, and B. K. Szymanski, "Overlapping community detection in networks: The state-of-the-art and comparative study," *ACM Computing Surveys (csur)*, vol. 45, no. 4, p. 43, 2013.

[6] J. Yang and J. Leskovec, "Community-affiliation graph model for overlapping network community detection," in *Data Mining (ICDM), 2012 IEEE 12th International Conference on*. IEEE, 2012, pp. 1170–1175.