NECTAR - Technical Report

August 7, 2016

This report contains technical details and a full review of the competitive analysis of the NECTAR algorithm, with respect to 9 state-of-the-art algorithms. This report is organised as follows:
Nectar uses a Random tree classifier in order to dynamically select the objective function to maximise. In Section 1 we present the classifier, and discuss the process of training it. A detailed comparison of results on real world networks with ground-truth, as well as on synthetic networks can be found in Sections 2, 3. In Section 4 a run-time comparison is presented. Section 5 presents a comparison to the AGM-fit method [1], in the methodology taken by AGM-fit algorithm developers. In Section 6 we present WOCC, an extension to the WCC [2] (Weighted Community Clustering) metric. Finally, in Section 7, u can find the full list of tools we used, along with links to their source codes.

1 Dynamic objective function choosing using machine learning

In order to choose dynamically which objective function should be used, we harnessed the power of machine learning.

As a train-set, we used a subset of 1174 combination from the below parameters list. (For 150 of them - the ones used for comparison in the full paper - we generated 2 instances.) $N \in \{1000, 1500, 3000, 5000, 10,000, 20,000\}$, $K \in \{5, 10, 20, 30\}$, $Onp \in \{10, 33, 50\}$ (percentage of nodes which belong to more than one community), $Om \in \{2, 3, 4, 5, 6, 7\}$, $maxK \in \{15, 20, 40, 50, 80\}$, $mu \in \{0.1, 0.2, 0.3\}$, $t1 \in \{1, 2\}$, $t2 \in \{1, 3\}$, The full train-set can be found in `ANONYMISED`.

On each we applied our method with both objective functions. The resulting partitions were then estimated using the three metrics, Omega-Index, NMI and Average-F1 score. For each instance we took the sum of the three metrics, for each objective function. The label of an instance was set to 0 if using Modularity yielded better results than WOCC, and 1 otherwise. (no ties were seen). We added as input to our classifier a weight per instance, which was based on the gap between the two scores. The weight is the absolute value of the gap dived by the largest of the two scores. Notice that when the scores are relatively close to each other, we dont care too much for the classification. We were able to reach better results using this approach than directly trying to predict the gap.

As for the features, for each graph we extracted 5 features: the average nodes degree, number of unique triangles divided by the number of nodes, the ration of nodes participating in a triangle out of all nodes,GCC - the global clustering coefficient and ACC - the average clustering coefficient.

We didn't use any features with absolute values (e.g. - number of nodes or edges) as our training set did not contained graphs larger than $20,000$ nodes, and we wanted our classifier to be able to classify also much larger networks.

As for the classification task, a random-tree classifier gave the best result. The tree depth is 4, its size is 27 and it contains 14 buckets. We used WEKA's,[3] RandomTree with the following values: $Kvalue$ - the number of randomly chosen attributes was set to 0, $allowUnclassifiedInstances$ is false, $minNum$

- the minimum total weight of the instances in a leaf was set to 1.0, without backfitting ($numFolds = 0$), $maxDepth$ - the maximum depth of the tree was set to 4 to prevent over fitting, $numDecimalPlaces$ - the number of decimal places to be used for the output of numbers in the model is 2, $batchSize$ was set to 100 and $breakTiesRandomly$ - break ties randomly when several attributes look equally good was false.

Using 10-fold cross-validation we got a weighted ROC of 0.993 and 0.978 weighted accuracy. TP rate, precision and recall are 0.979 and FP rate is 0.033. Also, the classifier is correct on DBLP, where the weight is significant - 0.41 (in Amazon, however, the weight is 0.03, and the wrong classification is not troubling).

Other then WEKA's RandomTree, we also experimented with XGBOOST (python) [4], with linear-regression on the gap between the two scores, as well as on the weight as described above. We experimented with the three parameters (lambda, alpha, lambda-rate] values in $0 - 10$ each, and number of iterations is in $2, 5, 10, 15, 20, 40, 100$. Using 10-fold cross-validation, the highest weighted accuracy gained was 0.935 (gap) and 0.861 (weight). Also, on DBLP, all regression results were wrong, therefor we preferred to use the RandomTree classifier.

2 Results on 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 [5] 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 [6] (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 [7], 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 1. In both cases our method returned a reasonable amount of communities in com-

parison to other algorithms.

Algorithm	Amazon	DBLP
NECTAR	23,863	44,523
GCE	25,962	22,657
Cfinder	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
DEMON	28,300	58,951
Big-Clam	15,000	15,000
LC	107,745	130,755

Table 1: 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 Cfinder 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.

2.1 Amazon

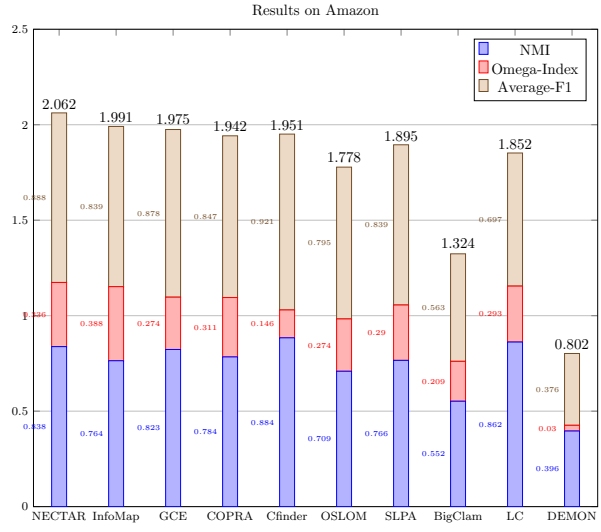

Figure 1: Amazon F1

2.2 DBLP

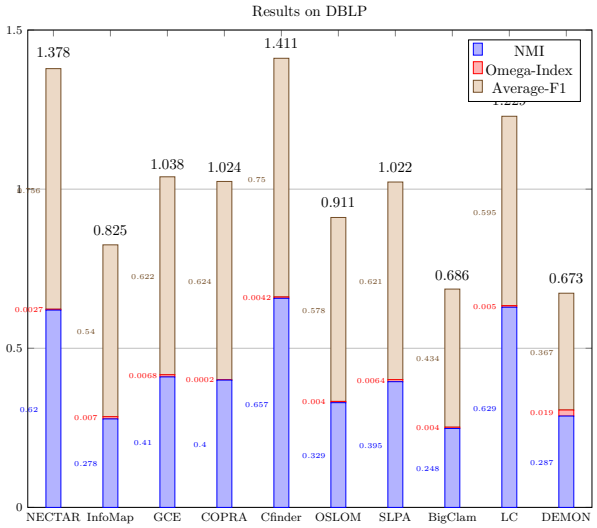

Figure 2: DBLP F1

3 Results on Synthetic networks

The following parameters were used for generating LFR graphs [8]:

Note that our experimental setup is almost identical to the setup used in [9]. 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 [9], we generated 10 instances for each combination of parameters. We took the aver-

age 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.
Cfinder - 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.
Big-Clam - for K (amount of communities) we used 20 different values in the range $100 - 350$ when the average degree was 10 and $120 - 600$ when it was 40. For real world networks, we used 10 different values in the range $5,000 - 15,000$.
LC - the python version was used.
DEMON - we used e in $0.25, 0.5, 0.8$.

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.

3.1 Synthetic networks - Full Comparison

In the following three sections we present a comprehensive comparison, conducted 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 these settings NECTAR maximizes modularity as the triangles rate is lower than 5.

We focus on the top two graphs, where the overlap rate is 50% ($O_n = 2500$). Therefor, as mentioned earlier, the conditions are harder for community detection. In these settings WOCC is maximized. Networks with big communities ($20 - 100$ nodes in a community) are presented in the left-hand side, in which our method outperforms the other algorithms in terms of NMI and Average-F1, and is equivalent to Cfinder in Omega-Index.

Small communities ($10 - 50$ nodes in a community) are presented in the right-hand side. Here Cfinder 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 $O_m = 8$. This fact encouraged us to believe that our method can give good results on complicated networks with high overlap rate.

3.2 NMI - Synthetic networks

Figure 3 illustrates a comparison of the NMI scores of the various algorithms, for graphs with average degree 40. A corresponding comparison for graphs with average degree 10 can be seen in figure 4.

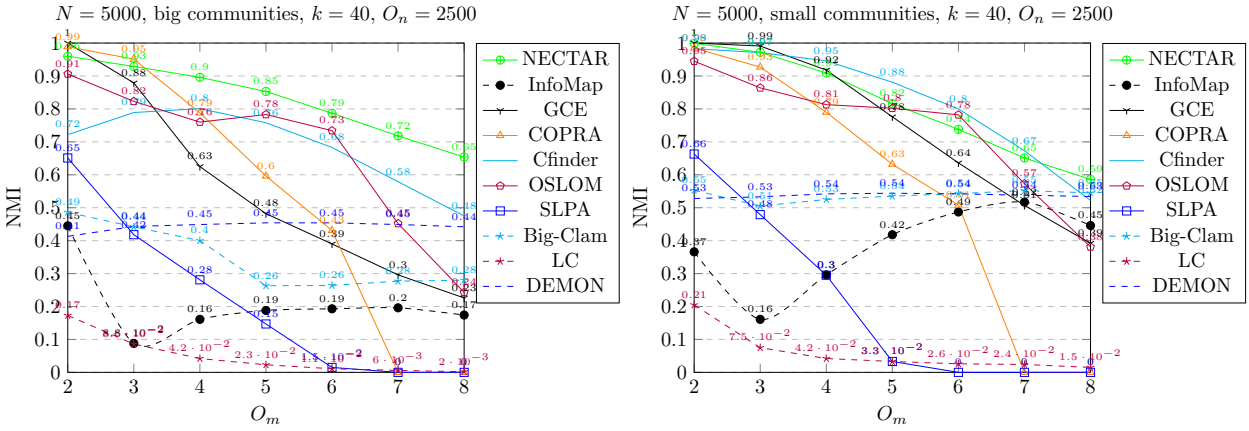

Figure 3: NMI

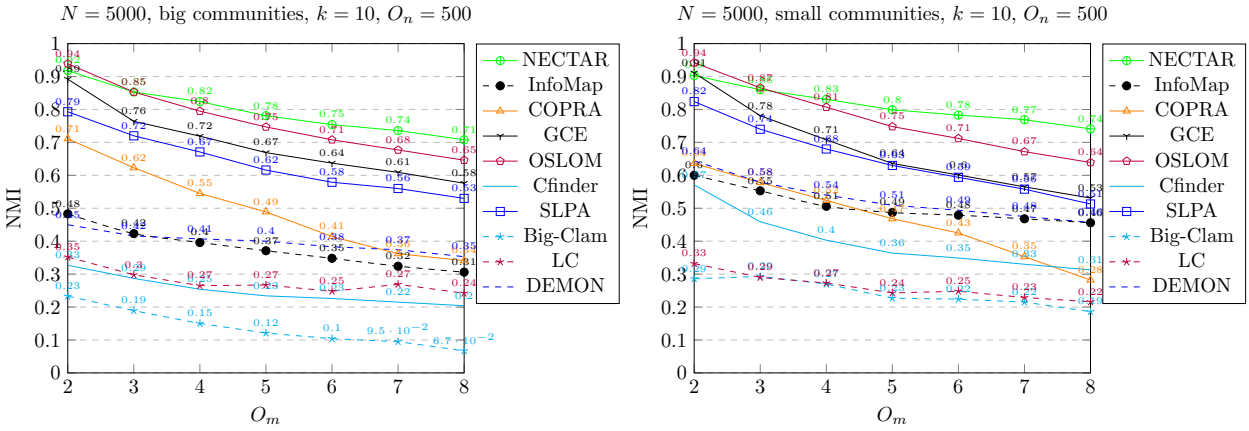

Figure 4: NMI-10

3.3 Omega-Index - Synthetic networks

Figure 5 illustrates a comparison of the Omega-Index scores of the various algorithms, for graphs with average degree 40. A corresponding comparison for graphs with average degree 10 can be seen in figure 6.

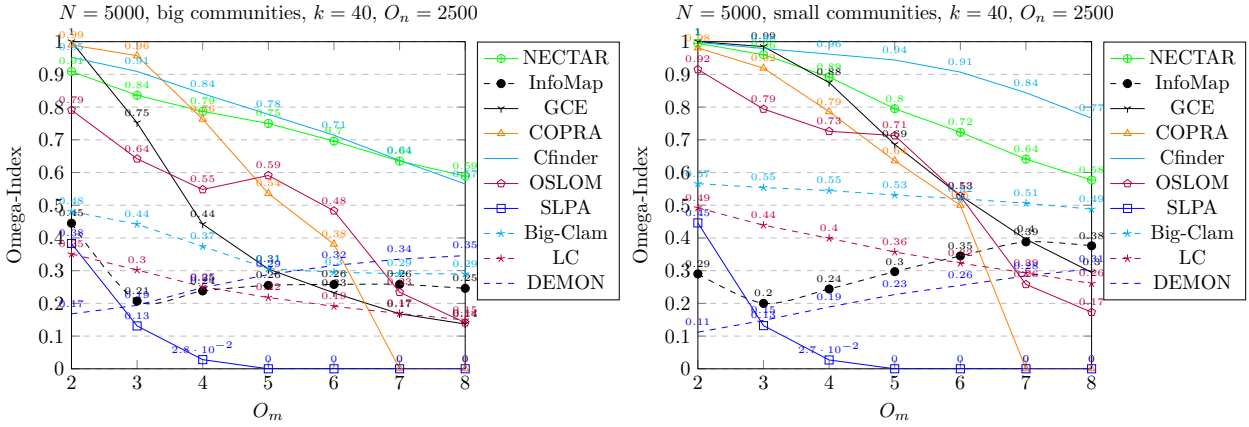

Figure 5: Omega

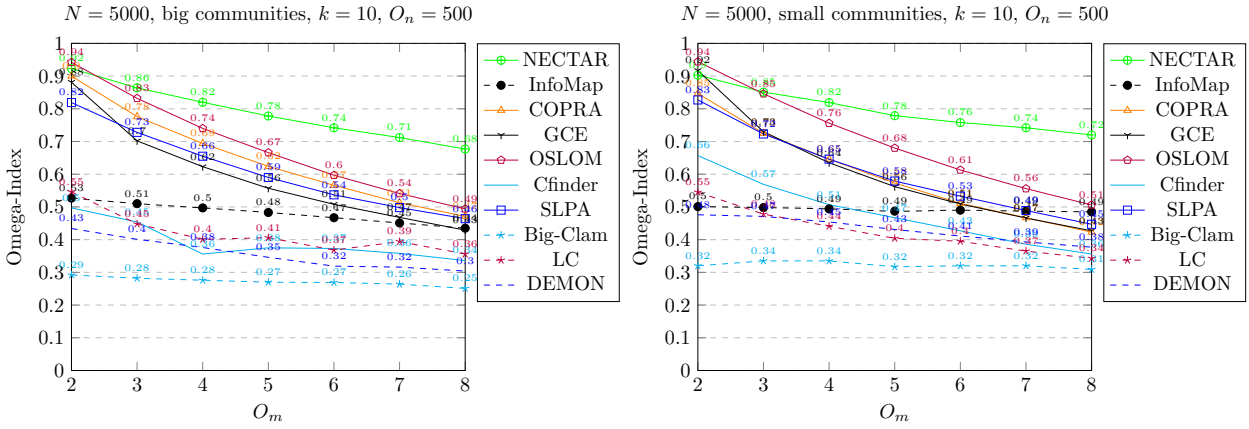

Figure 6: Omega-10

3.4 Average-F1 - Synthetic networks

Figure 7 illustrates a comparison of the Average-F1 scores of the various algorithms, for graphs with average degree 40. A corresponding comparison for graphs with average degree 10 can be seen in figure 8.

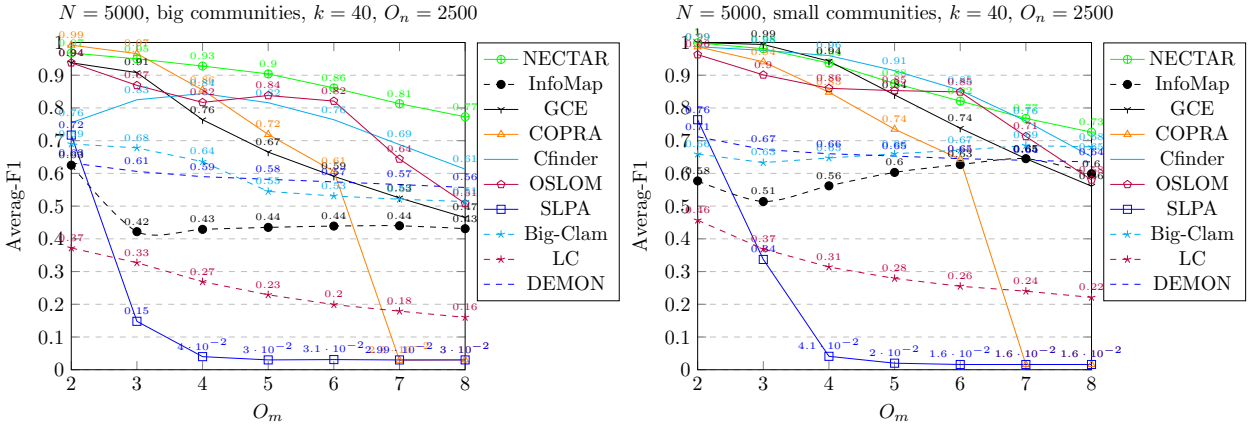

Figure 7: F1

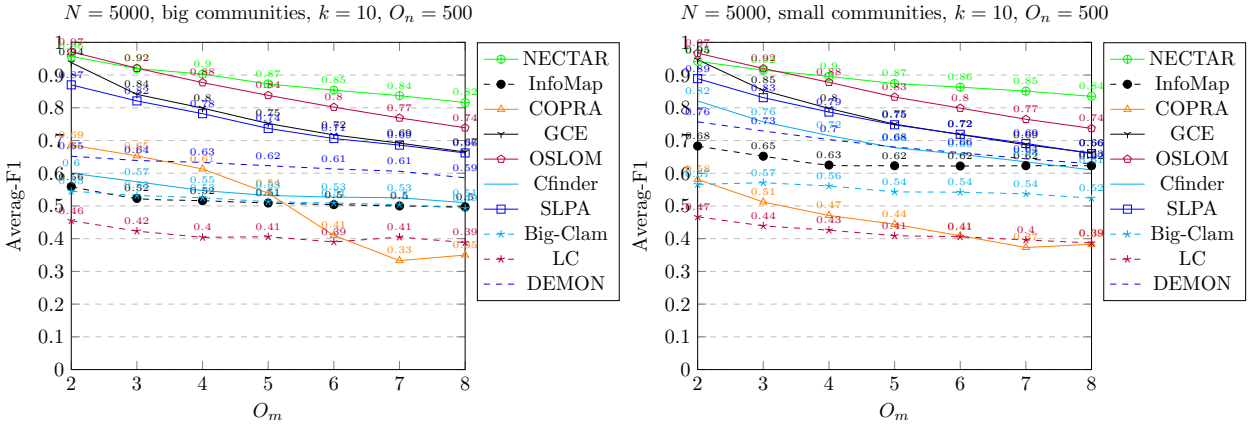

Figure 8: F1-10

4 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. Figure 9 - networks with $1,000 \leq N \leq 300,000$ (number of nodes), and $k = 10$ (average node degree).
a. Figure 10 - networks with $N = 5,000$ (number of nodes), and $10 \leq k \leq 70$ (average node degree).

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 varies between 10 to 50. We generated 2 instances for each combination, and averaged the run time. The experiments were conducted on a 24 core Linux machine with 128GB of memory.

As can be seen in the figures, 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 (9, 10) the algorithms are denoted by NECTAR-M and NECTAR-W respectively.

A note regarding the metric used: in all graphs in figure 9, NECTAR would used modularity (NECTAR-M). On the other hand, in all graphs in figure 10, except for the case where $K = 10$, NECTAR used WOCC (NECTAR-W).

More specifically, when N grows NECTAR-M's performance is equivalent to that of the other algorithms. For 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 implementation of calculating WOCC.

To better understand this we add NECTAR-M−24 and NECTAR-W−24 to the graph, which is the 24-threads run-time of the algorithm. The results are much better, placing our method in the second place.

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 $k = 10$, WOCC was in use by NECTAR.

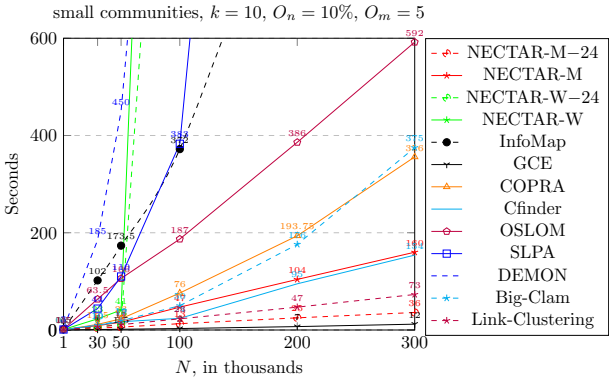

Figure 9: Run time comparison, varying N

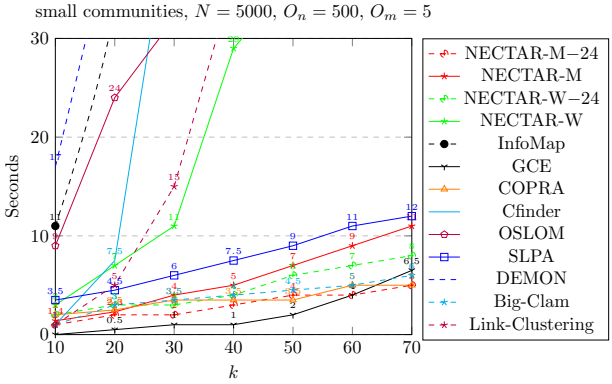

Figure 10: Run time comparison, varying K

5 Sub-networks community detection challenge

AGM-fit [1] models communities 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 net-

works can be then extract from the resulting AGM.

AGM failed to finish on several graphs after 128 hours, and therefor we follow the test methodology used in [1] to estimate it, and compare its results to our method. We used the DBLP network in our comparison, since AGM outperformed all other algorithms by the biggest gap, as can be seen in [1].

60 sub-networks of DBLP were used as input, generated as follows:
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 in at least 6 communities. For each such sampled node, we generated a sub-network in the following way: we took all nodes in the communities the node is a part of, and used all edges between them.

We limited the amount of nodes in a network to a maximum of 200 times the amount of communities used to generate it. For instance, say the node we started with was a member of 4 communities, then 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, since (as mentioned earlier) we observed that many of the communities in the ground-truth are almost identical.

Our method outperform AGM-fit, as shown in figure 11.

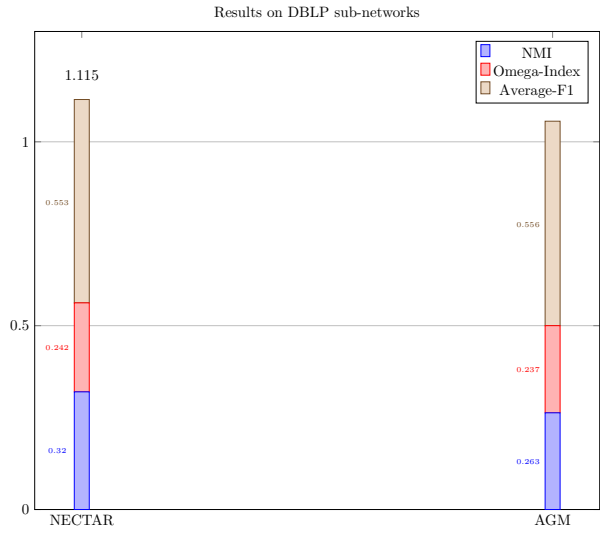

Figure 11: sub-networks comparison

6 WOCC - Weighted Overlapping Community Clustering

Weighted Community Clustering (WCC) [2] measures the quality of a given partitioning of the graph, with respect to the triangles found in the network. It is based on the observation that triangle structures are much more likely to exist within communities than across them. This observation is leveraged for quantifying the quality of graph partitions (that is, non-overlapping communities). It is formally defined as follows. For a set of nodes S and a node v, let $t(v,S)$ denote the number of triangles that v closes with nodes of S. Also, let $vt(v,S)$ denote the number of nodes of S that form at least one triangle with v. $WCC(v,S)$, quantifying the extent by which v should be a member of S, is defined as:

$$WCC(v,S) = \begin{cases} \frac{t(v,S)}{t(v,V)} \cdot \frac{vt(v,V)}{|S \setminus v| + vt(v, V \setminus S)} & t(v,V) > 0 \\ 0 & otherwise \end{cases} \quad (1)$$

where V is the set of graph nodes. The cohesion level of a community S is defined as:

$$WCC(S) = \frac{1}{|S|} \sum_{v \in S} WCC(v,S). \quad (2)$$

Finally, the quality of a partition $\mathcal{C} = \{C_1, \ldots, C_n\}$ is defined as the following weighted average:

$$WCC(\mathcal{C}) = \frac{1}{|V|} \sum_{i=1}^{n} |C_i| \cdot WCC(C_i). \quad (3)$$

We present $WOCC$ - Weighted Overlapping Community Clustering. WOCC is based on the above definition of WCC, adjusted to fit the weighted overlapping community detection problem. Global WOCC is presented in details below. Notice that WOCC reduces to WCC (non-weighted, non-overlapping) when all edges weights are 1 and no overlapping communities are presented as input.

$WOCC(v, S)$ is the improvement gained in WOCC when adding node v to community S. We set it to be 0 if $t(v, V) = 0$, and otherwise

$$\frac{t(v,S)}{t(v,V)} \cdot \frac{vt(v,V)}{(|S \setminus \{v\}| \cdot \hat{w} + vt(v, V \setminus S)}$$

Where:
For a triangle T in G, $weight(T)$ is the minimum weight of the edges in T.
$t(v,S) := \sum_{T \text{ is a triangle in } S \wedge v \in T} weight(T)$.
The sum of weights of all triangles in S which v is a node in.
$vt(v,S) := \sum weight(u,v)$. Where $u \in V$ and v and u share a triangle in S.
The sum of weights of edges to nodes which close at one triangle in S with v.
$\hat{w}$ is the average edge weight.
We use $\hat{w}$ in order to reflect the relationship between the node v and community S - in the original WCC, the second fraction is equal to 1 when node v closes a triangle with all nodes in S. In our case, we must take into account the edges weights, and we therefor use $\hat{w}$ to estimate the strongest possible connection between v and S, with respect to the edges weights in the graph. Alternatively, the average weights of edges connected to v can be used, instead of $\hat{w}$.

Finally, for a given set of sets of nodes $\mathcal{C}$, covering the network, the global WOCC is:

$$WOCC(\mathcal{C}) = \frac{1}{\sum_{S_i \in \mathcal{C}} |S_i|} \cdot \sum_{S_i \in \mathcal{C}} \sum_{v \in S_i} WOCC(v, S_i)$$

.
The main difference between the global WCC and the global WOCC is that we divide by the sum of sizes of all communities, instead of the number of nodes.

7 Tool-box

NECTAR implementation can be found at Anonymised
$LFR(syntheticgraphgenerator)implementation -$

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/`
Cfinder - `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/`
Big-Clam - `https://snap.stanford.edu/snap/description.html`
DEMON - `http://www.michelecoscia.com/?page_id=42`
LC - `http://barabasilab.neu.edu/projects/linkcommunities/`

Sources for the metrics we used:
NMI - `https://github.com/aaronmcdaid/Overlapping-NMI`
Average-F1 and Omega-Index - Anonymised (our implementation)

Machine learning: WEKA - `http://www.cs.waikato.ac.nz/ml/weka/`
XGBOOST -`http://xgboost.readthedocs.io/en/latest/`

References

[1] 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.

[2] A. Prat-Pérez, D. Dominguez-Sal, J. M. Brunat, and J.-L. Larriba-Pey, "Shaping communities out of triangles," in *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, 2012, pp. 1677–1681.

[3] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: An update; sigkdd explorations, volume 11, issue 1, 2009," *Reproduced with permission of the copyright owner. Further reproduction prohibited without permission*, 2013.

[4] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," *arXiv preprint arXiv:1603.02754*, 2016.

[5] 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.

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

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

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

[9] 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.