Core Decomposition in Directed Networks: Kernelization and Strong Connectivity

Vincent Levorato

Abstract. In this paper, we propose a method allowing decomposition of directed networks into cores, which final objective is the detection of communities. We based our approach on the fact that a community should be composed of elements having communication in both directions. Therefore, we propose a method based on digraph kernelization and strongly p-connected components. By identifying cores, one can use based-centers clustering methods to generate full communities. Some experiments have been made on three real-world networks, and have been evaluated using the V-Measure, having a more precise analysis through its two sub-measures: homogeneity and completeness. Our work proposes different directions about the use of kernelization into structure analysis, and strong connectivity concept as an alternative to modularity optimization.

1 Introduction

Complex networks appear in many applications, including social networks analysis on the Web, which is a topical research subject. These networks carry non-trivial topological properties that characterize their connectivity, and affect the dynamics of their behavior. The analysis of complex networks often leads to the analysis of the roles of elements, or groups of elements, composing a network. Communities detection belongs to this research field, and can be very useful to better understand how networks are structured. In this article, we focus on the problem of finding communities in networks, and more specifically finding cores in *directed networks*. Dealing with methods of community detection for directed networks is a difficult task, and few methods exist compared to methods used in the undirected

Vincent Levorato
CESI, 959 rue de la Bergeresse, 45160 OLIVET, FR
e-mail: vlevorato@cesi.fr

Université d'Orléans, LIFO, Bat. IIIA rue Léonard de Vinci, 45067 ORLEANS, FR
e-mail: vincent.levorato@univ-orleans.fr

P. Contucci et al. (eds.), *Complex Networks V*,
Studies in Computational Intelligence 549,
DOI: 10.1007/978-3-319-05401-8_13,

case. Here are some of the most known works in the literature [9, 17, 13] dedicated to directed networks, or which can be adapted to work with directiveness: *Clauset et al.* method [5], *CFinder* based on the Clique Percolation Method (CPM) [21], *Louvain* method [3], *InfoMap* [23], *Simulated Annealing* for modularity [11], *Wu-Huberman* method [31], *MarkovCluster* algorithm [29], *Multistep Greedy* algorithm [24] and *EM* method (Expectation-Maximization) [18]. More recently, others methods concerning directed networks have been proposed, with more or less good results [32, 15, 14].

Generally, these methods are most of the time designed for undirected networks, and adjusted to work in the directed case: they are *not initially dedicated to the directed case*. There is also a significant amount of methods using modularity optimization. However, this kind of approach has its limits, and can *"miss important substructures of a network"* [10]. Some recent work discuss about *reciprocated interaction* [4], that two people should communicate in both directions, the first person expecting messages from the second person, and vice versa. Our approach is based on this simple idea: in a directed network, *a community should be composed of nodes which can communicate with every nodes in the community, in both directions*. Interesting results in our previous exploratory work [19] encourage us to continue in this direction. Usually represented by graphs in undirected networks, this kind of representation can be modeled by the *connected component* concept, and more restrictively by the *clique* concept. Except that for the directed case, it can be represented by the concept of *strongly connected component*. Finding these components should be equivalent to find *cores* to which other elements of the network will be assigned. This work gives the key concepts of this approach, focusing on the cores finding.

Our paper is structured as such: the first section gives the definitions of graph theory and formal concepts needed to understand our method. Then, the second part exposes the different steps of our approach, followed by some experimental results on real networks. The paper ends by a conclusion which opens discussion on future work directions.

2 Graph Theory Notions

2.1 Graph Definitions

In this article, we consider only *directed graphs* (also noted *digraph*). We give here a short reminder of graph theory notions. Formally, a digraph $G = (V,A)$ is the pair composed of [2]:

- a set $V = \{x_1, x_2, ..., x_n\}$ named *vertices* or *nodes*.
- a family $A = (a_1, a_2, ..., a_n)$ of elements of the Cartesian product $V \times V = \{(x,y)/x \in V, y \in V\}$ named *arcs*.

The amount of vertices is noted n (also noted $|V(G)|$) and the amount of arcs is noted m (also noted $|A(G)|$). A *path* P is composed of k arcs such as $P = (a_1, a_2, ..., a_i, ..., a_k)$ where for every arc a_i the terminal end coincides with the initial

end of a_{i+1}. Several equivalent notations can be used: $P = ((x_1,x_2),(x_2,x_3),...) = [x_1,x_2,...,x_k,x_{k+1}] = P[x_1,x_{k+1}]$. A *chain* is, like a path, an alternating sequence of vertices and edges, where an edge is an arc without orientation. A *circuit* is a path such that the first node of the path corresponds to the last. It can be viewed as an oriented cycle.

2.2 Connected Components

Here are the different types of connected components we could have in a directed graph [12]:

- a *weakly connected component WCC* of a digraph is a subgraph where: $\forall x,y \in WCC$, there is a chain between x and y.
- an *unilaterally connected component UCC* of a digraph is a subgraph where: $\forall x,y \in UCC$, there is a path between x and y OR there is a path between y and x.
- a *strongly connected component SCC* of a digraph is a subgraph where: $\forall x,y \in SCC$, there is a path between x and y AND there is a path between y and x.

To discover cores in a network, we use a special case of strongly connected component named *strongly p-connected component* by [30] which is related to *l*-edge-connectivity [7] (we use *p-connected* notation instead of *n-connected* to avoid confusion):

- a *strongly p-connected component* p-SCC of a digraph is a subgraph where: $\forall x,y \in p$-SCC, there is a path of length p or less between x and y, and there is a path of length p or less between y and x, with $p \geq 2$.

3 Core detection

3.1 Related Work

Finding cores in order to find communities is a method that can be related to *pattern* identification [13]. This consists in finding maximal subsets which implies separation between them. Clique finding is one of these methods, but is also very restrictive, because each node must have a direct connection to other nodes. This approach has been relaxed by the n-clique definition where each node is connected to others by at least one path which length is at most n, but that can be outside of the n-clique. The n-clan concept fixes the connectedness issue of the n-clique [20].

Our approach is related to these works, but we don't put strong constraint on the size (triads), and we don't want to avoid circuits (directed k-clique), as it is specifically the configuration we are looking for: strongly p-connected components.

3.2 *Searching for Cores Using p-SCC Concept*

Our community definition refers to a group containing elements that can communicate with all other elements of the group, corresponding in digraphs, to strongly connected components (SCC) concept. Tarjan based his algorithm on the search of circuits [27]. To our knowledge, no work has considered the SCCs in the case of researching communities. By simply applying Tarjan's algorithm to directed graph generated through LFR benchmark [16], some communities can be found, but SCCs are often oversized. To refine the process, our approach proposes to find p-SCCs (fig. 1). The problem is that in a digraph, the number of circuits may be exponential in the number of vertices [26]. Therefore, processing all circuits of a graph is not relevant, especially if the graph has a significant number of nodes like in large real-world networks.

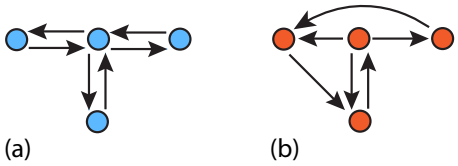


Fig. 1 Examples of p-SCCs: *(a)*: nodes are connected by paths of length at most 2 (2-SCC) *(b)*: nodes are connected by paths of length at most 3 (3-SCC)

Starting from a node s, we look for p-SCC by searching for circuits, but circuits with a given size. Trivially, the length of the path p is bounded by the length of the circuits found into the p-SCC :

$$p \leq 2 \times (c-1)$$

where c is the size of circuits we are searching for.

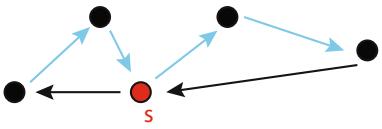


Fig. 2 Searching p-SCC is similar to search circuits from a starting node s. In this case, searching for 4-SCC means searching for circuits of length at most 3. The highlighted path length is 4, the maximal path length which can be found.

For instance, searching for circuits of length 3 starting from a node means searching for at most 4-SCCs (fig. 2). We propose an algorithm which returns a p-SCC starting from a given node (alg. 1). As the algorithm is searching for circuits, and considering that p parameter sets the circuit length, we can only find p-SCCs with p being an even number. It is written in a non-recursive way, but time complexity should be approximatively the same as Tarjan's algorithm, which is $O(n+m)$, with two differences: we don't always need to pass through every arc (depends on path length), but we should pass through nodes several times.

```
Input: G: digraph, s:starting node, p:path length (even integer)
Data: astack: stack of arcs, vpath: stack of nodes, c: integer (circuit size)
Remark: Aout(k) represents set of outing arcs of the node k.
source(a) represents the source node of the arc a, dest(a) represents the destination node of the arc a.
Result: C:set of nodes (p-SCC)
C ← ∅; C ← C ∪ {s};
vpath.push(s); c ← (p+2)/2;
foreach a in Aout(s) do
    astack.push(a);
end
while astack ≠ ∅ do
    a ← astack.pop();
    w ← vpath.peek();
    if source(a) ≠ w then
        while source(a) ≠ w do
            w ← vpath.pop();
        end
        vpath.push(w);
    end
    z ← dest(a);
    if z = s then
        C ← C ∪ vpath;
    end
    else
        if |vpath| < c then
            foreach b in Aout(z) do
                astack.push(b);
            end
            vpath.push(z);
        end
    end
end
return C;
```

Algorithm 1. Algorithm for extracting the p-SCC.

3.3 Digraph Kernelization

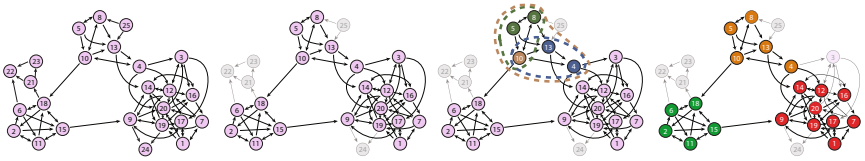
Before describing the whole method based on core detection, we show how we can optimize the search for p-SCCs by "cleaning" the digraph, meaning excluding nodes and arcs which should never belong to a circuit: this brings us to the notion of *graph kernelization*. We are interested in the kernelization which is used in the FVS problem (Feedback Vertex Set) [28]. As we work in the directed case, we used the kernelization technique applied to the directed FVS [8], following the four first rules of the method (it removes self-loop, multiple arcs, isolated nodes, and recursively chained nodes with only outgoing or incoming arcs).

4 Digraph Cores Decomposition Method

This section describes our method for core detection in directed networks. Let use the following notations: G is the input digraph (network), and $\mathscr{K}$ is the set of output cores. The method follows these steps, considering a given p:

1. Kernelize G.
2. For each node of G as the starting node, process p-SCC.
3. Sort p-SCCs by size. Starting from the biggest one, put them one by one in $\mathscr{K}$ if it doesn't intersect existing cores already inserted into $\mathscr{K}$. In case of cores having the same size, take the most connected one (biggest amount of arcs).
4. (optionnal) Remove p-SCCs with size inferior to a given threshold K_{min}.

Illustration: The figure 3 gives an illustration of our method, step by step, with $p = 4$ (meaning we search for circuits of length at most 3). Let take a digraph (a), and apply the first step which is kernelization (b). Some nodes are ignored, and won't be considered. The second step processes 4-SCCs, node by node: in the example (c), only five iterations are represented (nodes with labels 4 ,5 ,8 ,10 ,13), and for each node, a 4-SCC is computed, which can be the same for several nodes (nodes 5 and 8 produce the same 4-SCC, same thing for nodes 4 and 13). The last step (d) extracts the biggest and non-intersecting 4-SCCs giving the final result with 3 cores.



(a) Input digraph (b) Kernelization (c) Processing the 4−SCCs (d) Cores result

Fig. 3 Illustration of the decomposition method of a digraph into cores

This method returns a set of cores which can be used to cluster the remaining nodes of the network to have a complete clustering. As some clustering methods like k-means algorithm, the number of communities is set by the number of cores. In this article, as we don't focus on clustering methods, our experiments use a simple aggregative method like center-based clustering methods, and assign each node to the community having the nearest core, in term of *chain* length. The K_{min} value can be useful to avoid too small cores that shouldn't be considered.

5 Experiments

Validating communities structures corresponds to validate a clustering method. The difficulty is to find an objective measure of quality of clusters. For our experiments,

we use *V-Measure* which is an alternative to F-Measure, and *Normalized Mutual Information* from the information theory field to compare the clustering obtained by our method to the reference classes. We based our experiments on real data networks, as some experiments have already been done on generated networks (LFR Benchmarks [16]) in our previous work with good results [19]. Moreover, there is an issue with the LFR Benchmark, as *it produces graphs already kernelized*, which puts a strong constraint on generated graphs. On the contrary, in the experiment part, we observe that the real-world networks we used are strongly kernelizable.

5.1 Clustering Evaluation

The entropy notion is used to express the used measures, and is noted as follow, with X and Y two discrete random variables: $H(X)$ and $H(Y)$ for the marginal entropies, $H(X|Y)$ and $H(Y|X)$ for the conditional entropies, $H(X,Y)$ for the joint entropy. Measures give results between 0 (worst matching) and 1 (best matching). Here are the two evaluation measures definitions:

- **V-Measure** [22] is an entropy based-evaluation measure, composed of two concepts: *completeness* and *homogeneity*. With C a set of classes (reference), K a set of clusters (unsupervised method), the homogeneity is defined as:

$$h = \begin{cases} 1 & if\ H(C,K) = 0 \\ 1 - \frac{H(C|K)}{H(C)} & else \end{cases}$$

  and the completeness is defined as:

$$c = \begin{cases} 1 & if\ H(K,C) = 0 \\ 1 - \frac{H(K|C)}{H(K)} & else \end{cases}$$

  A clustering result satisfies homogeneity if all of its clusters contain only elements which are members of a single class, and a clustering result satisfies completeness if all the elements that are members of a given class are elements of the same cluster. The V-Measure is based on homogeneity and completeness scores such as:

$$V_\beta = \frac{(1+\beta) \cdot h \cdot c}{(\beta \cdot h) + c}$$

  with a β parameter which can be used to weight homogeneity and completeness scores. In our experiments, we set $\beta = 1$ (balanced weights).

- **Normalized Mutual Information** (NMI) [6] Mutual information is a measure used in information theory domain, giving the amount of information that one random variable contains about another. This measure is defined between the cluster assignments K and a pre-existing labeling set of classes C normalized by:

$$NMI(K,C) = \frac{I(K,C)}{\sqrt{H(K)H(C)}}$$

with $I(K,C)$ the mutual information of K and C such that $I(K,C) = H(K) - H(K|C)$.

5.2 Results

In order to test our approach, we used directed network datasets already known in the literature [1, 25]. Three networks have been used (see tab. 1). Several characteristics of the network are given like density, degree information, communities maximum and minimum sizes, but also the mixing parameter μ [17] and the directed modularity Q_d [18].

Table 1 Network datasets

Directed Network	*Political Blog*	*Cora*	*Citeseer*
$\|V\|$	1,222	2,485	2,120
$\|A\|$	19,024	5,209	3,768
Classes	2	7	6
Density	1.27%	0.08%	0.08%
Degrees	$k_{mean} = 31$ $k_{min} = 1$ $k_{max} = 467$	$k_{mean} = 4$ $k_{min} = 1$ $k_{max} = 169$	$k_{mean} = 4$ $k_{min} = 1$ $k_{max} = 100$
Communities size	$\|C\|_{min} = 588$ $\|C\|_{max} = 636$	$\|C\|_{min} = 131$ $\|C\|_{max} = 726$	$\|C\|_{min} = 115$ $\|C\|_{max} = 532$
μ	0.09	0.18	0.28
Q_d	0.41	0.63	0.51

5.2.1 Core Decomposition

In tab. 3, we compare the obtained cores with the reference classes, using the p parameter which corresponds to the path length of a p-SCC, and the K_{min} parameter which is the minimum core size (only relevant results are shown). In most cases, cores have a good completeness score, meaning that we succeed in having nodes which belong to a single class in only one core. On the other hand, the homogeneity score tends to be better when the threshold of the minimum core size is increased (Political Blog and Cora networks), having nodes of a same core belonging to a single class. The interpretation that can be made from these results is that the more the graph is compressed, the less the K_{min} gets an high value. When too many nodes are available to build cores, the K_{min} threshold has to be high to remove some eventual noise, giving less nodes usable in the core creation. In the results of tab. 3, we first consider completeness value, and then the homogeneity value. We give more importance to the completeness score, as it gives better results in the final process communities detection.

Table 2 Network kernel sizes

Directed Network	*Political Blog*	*Cora*	*Citeseer*
$\|V\|_K$	811	2,485	2,120
$\|A\|_K$	15,833	5,209	3,768
Compression rate (nodes)	33%	84%	97%

Table 3 Cores detection on real-world networks

(a) Political Blog

		V-Measure				
p	**K min size**	**h**	**c**	**V**	**Nb Nodes**	**Nb Cores**
2	2	0.35316	0.95295	0.51534	329	20
2	3	0.40471	0.94876	0.56739	308	13
2	4	0.44344	0.94601	0.60383	296	10
2	5	0.52053	0.96137	0.67538	281	7
2	6	0.59364	0.97468	0.73787	269	5
2	7	0.7381	**1.0**	0.84932	255	3
2	**16**	**1.0**	**1.0**	1.0	239	2
4	2	0.60926	0.98967	0.75421	401	12
4	3	0.79398	0.98896	0.88081	380	5
4	4	0.90979	**1.0**	0.95276	372	3
4	**5**	**1.0**	**1.0**	1.0	367	2

(b) Cora

		V-Measure				
p	**K min size**	**h**	**c**	**V**	**Nb Nodes**	**Nb Cores**
4	2	**0.41019**	**0.9412**	0.57137	98	28
4	3	0.57836	0.9352	0.71471	47	11
6	2	0.41481	0.93729	0.5751	103	28
6	3	0.58141	0.92778	0.71485	52	11
6	4	0.70183	0.92268	0.79724	32	6

(c) Citeseer

		V-Measure				
p	**K min size**	**h**	**c**	**V**	**Nb Nodes**	**Nb Cores**
2	1	0.49039	0.93415	0.64315	54	26
2	2	0.19087	0.29364	0.23136	6	2
4	**1**	**0.48994**	**0.93454**	0.64285	58	26
4	2	0.5907	0.77251	0.66948	12	3

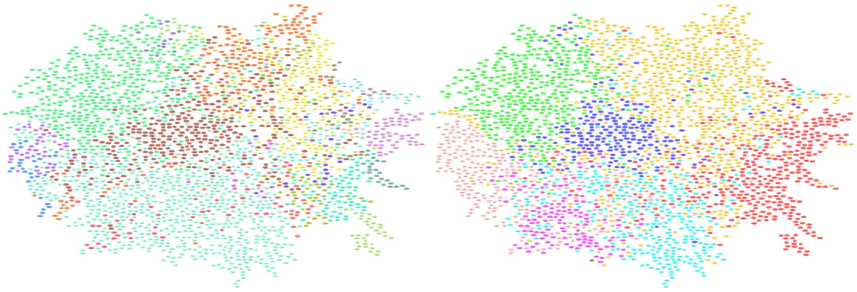
5.2.2 Communities Detection

Using an aggregative method, the cores first absorb nodes which are in the kernel but not in the cores, giving pre-built communities. Then, the nodes outside the kernel are absorbed by the pre-built communities to give a final clustering of communities. Clusters are not strongly connected, but unilaterally connected. The results in tab. 4 show that even with a naive method of clustering, the communities structures remain ”acceptable“. The cores used in the final clustering process are the cores having the best completeness scores in the core decomposition operation. Observing the results, we can make the assumption that the compression of graphs impacts the quality of cores, and therefore the detection of communities. With a small amount of nodes in the kernel, the choice to make between the nodes to build the cores is important, as it determines the final process of communities detection. Also, having only big cores means setting a too high K_{min} threshold value, which can have a negative impact on the cores detection, and some communities cannot be found in the process. For

instance, in fig. 4, the central community has been found by our method using the parameters $p = 4, K_{min} = 2$. If we set $K_{min} = 3$, cores of size 2 are excluded, and this central community is not detected.

Table 4 Real-world networks communities detection results based on core decomposition

Network	Measures				Amount of Communities
	h	c	V	NMI	
Political Blog	0.70385	0.69929	0.70156	0.70116	2
Cora	0.35335	0.46349	0.40099	0.40469	28
Citeseer	0.28162	0.38734	0.32613	0.32742	26



(a) Clustering using an aggregative method based on core decomposition.

(b) Reference classes.

Fig. 4 Communities detection comparison on the Cora citation network ($p = 4, K_{min} = 2$)

6 Conclusion

In this article, we focused on an approach dedicated to directed networks, and we gave a method allowing the decomposition of these networks into cores. These cores can be used by any clustering method based on centers to detect communities. Our various contributions can be presented as follows:

- We provide a simple and efficient algorithm to generate these p-SCCs in a digraph. This approach can be classified in the *pattern identification* category that we can find in some method classification, while being flexible enough.
- The interest of using kernelization process has been highlighted : it reduces the core detection process, and can give some information on the network structure.
- An important thing about these results is that we didn't take into account the modularity concept in our approach. As a large part of the communities detection algorithms are dedicated to modularity optimization [13], we want to stress the point that we can have interesting results in communities detection without this concept.

Several options can be considered for the continuation of this work. As we said, we have to apply our method to others real-world datasets. We should also study how to increase the quality of the core detection, and it could be interesting to have the possibility to automatically fix the K_{min} threshold value. Testing other based-centers clustering methods should be done too. Also, the case of overlapping communities should be considered, as our approach could be quickly adaptable with p-SCCs which naturally overlap each other. In our opinion, our work points out that no clear or unanimous consensus about the definition of communities exists, and provides a new point of view on the detection of communities into directed networks, being omnipresent in the Web nowadays.

Acknowledgements. We would like to thank Pr. Ioan Todinca, Mathieu Liedloff, Anthony Perez and the GAMoC team from the LIFO laboratory at University of Orléans (FR) for their listening and advices. Special thanks to Guillaume Cleuziou from the Constraints and Machine Learning team.

References

1. Adamic, L.A., Glance, N.: The political blogosphere and the 2004 u.s. election: divided they blog. In: Proceedings of the 3rd International Workshop on Link Discovery (LinkKDD 2005), pp. 36–43. ACM, New York (2005)
2. Berge, C.: Graphes et Hypergraphes. Dunod, Paris (1970)
3. Blondel, V.D., Guillaume, J.-L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. Journal of Statistical Mechanics (10) (2008)
4. Cheng, J., Romero, D.M., Meeder, B., Kleinberg, J.M.: Predicting reciprocity in social networks. In: SocialCom/PASSAT, pp. 49–56 (2011)
5. Clauset, A., Newman, M.E.J., Moore, C.: Finding community structure in very large networks. Phys. Rev. E 70(6), 066111 (2004)
6. Danon, L., Díaz-Guilera, A., Duch, J., Arenas, A.: Comparing community structure identification. Journal of Statistical Mechanics: Theory and Experiment (9), P09008–P09008 (2005)
7. Diestel, R.: Graph Theory, 4th revised edn. Springer (July 2010)
8. Fleischer, R., Wu, X., Yuan, L.: Experimental study of fpt algorithms for the directed feedback vertex set problem. In: Fiat, A., Sanders, P. (eds.) ESA 2009. LNCS, vol. 5757, pp. 611–622. Springer, Heidelberg (2009)
9. Fortunato, S.: Community detection in graphs. Physics Reports 486(3-5), 74–174 (2010)
10. Fortunato, S., Barthélemy, M.: Resolution limit in community detection. Proceedings of the National Academy of Science 104(1), 36–41 (2006)
11. Guimerà, R., Nunes Amaral, L.A.: Functional cartography of complex metabolic networks. Nature 433, 895–900 (2005)
12. Harary, F.: Graph Theory. Addison-Wesley, Reading (1969)
13. Labatut, V., Balasque, J.-M.: Detection and interpretation of communities in complex networks: Practical methods and application. In: Computational Social Networks: Tools, Perspectives and Applications, pp. 81–113 (2012)
14. Lai, D., Lu, H., Nardini, C.: Finding communities in directed networks by pagerank random walk induced network embedding. Physica A: Statistical Mechanics and its Applications 389(12), 2443–2454 (2010)

15. Lambiotte, R., Sinatra, R., Delvenne, J.-C., Evans, T.S., Barahona, M., Latora, V.: Flow graphs: Interweaving dynamics and structure. Phys. Rev. E 84, 017102 (2011)
16. Lancichinetti, A., Fortunato, S.: Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. Physical Review E 80(1), 1–8 (2009)
17. Lancichinetti, A., Fortunato, S.: Community detection algorithms: A comparative analysis. Phys. Rev. E 80, 056117 (2009)
18. Leicht, E.A., Newman, M.E.J.: Community structure in directed networks. Phys. Rev. Lett. 100(11), 118703 (2008)
19. Levorato, V., Petermann, C.: Detection of communities in directed networks based on strongly p-connected components. In: IEEE International Conference on Computational Aspects of Social Networks (CASoN), pp. 211–216 (October 2011)
20. Mokken, R.J.: Cliques, clubs and clans. Quality & Quantity 13(2), 161–173 (1979)
21. Palla, G., Derényi, I., Farkas, I., Vicsek, T.: Uncovering the overlapping community structure of complex networks in nature and society. Nature 435, 814–818 (2005)
22. Rosenberg, A., Hirschberg, J.: V-measure: A conditional entropy-based external cluster evaluation measure. In: Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), pp. 410–420 (2007)
23. Rosvall, M., Bergstrom, C.T.: Maps of random walks on complex networks reveal community structure. In: Proceedings of the National Academy of Sciences USA, pp. 1118–1123 (2008)
24. Schuetz, P., Caflisch, A.: Efficient modularity optimization by multistep greedy algorithm and vertex mover refinement. Phys. Rev. E 77, 046112 (2008)
25. Sen, P., Namata, G.M., Bilgic, M., Getoor, L., Gallagher, B., Eliassi-Rad, T.: Collective classification in network data. AI Magazine 29(3), 93–106 (2008)
26. Tarjan, R.: Enumeration of the Elementary Circuits of a Directed Graph. SIAM Journal on Computing 2(3), 211–216 (1973)
27. Tarjan, R.E.: Depth-first search and linear graph algorithms. SIAM Journal on Computing 1(2), 146–160 (1972)
28. Thomassé, S.: A quadratic kernel for feedback vertex set. In: Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2009), Philadelphia, PA, USA, pp. 115–119. Society for Industrial and Applied Mathematics (2009)
29. Van Dongen, S.: Graph clustering via a discrete uncoupling process. SIAM. J. Matrix Anal. & Appl. 30(1), 121–141 (2008)
30. Wasserman, S., Faust, K.: Social Network Analysis: Methods and Applications. Cambridge University Press (1994)
31. Wu, F., Huberman, B.A.: Finding communities in linear time: A physics approach. The European Physical Journal B Condensed Matter 38(2), 331–338 (2003)
32. Yang, T., Chi, Y., Zhu, S., Jin, R.: Directed network community detection: A popularity and productivity link model. In: SDM 2010: Proceedings of the 2010 SIAM International Conference on Data Mining (2010)