Arc-Community Detection via Triangular Random Walks

Paolo Boldi Marco Rosa
Dipartimento di Informatica, Università degli Studi di Milano, Italy

***Abstract*—Community detection in social networks is a topic of central importance in modern graph mining, and the existence of overlapping communities has recently given rise to new interest in arc clustering. In this paper, we propose the notion of triangular random walk as a way to unveil arc-community structure in social graphs: a triangular walk is a random process that insists differently on arcs that close a triangle. We prove that triangular walks can be used effectively, by translating them into a standard weighted random walk on the line graph; our experiments show that the weights so defined are in fact very helpful in determining the similarity between arcs and yield high-quality clustering. Even if our technique gives a weighting scheme on the line graph and can be combined with any node-clustering method in the final phase, to make our approach more scalable we also propose an algorithm (ALP) that produces the clustering directly without the need to build the weighted line graph explicitly. Our experiments show that ALP, besides providing the largest accuracy, it is also the fastest and most scalable among all arc-clustering algorithms we are aware of.**

I. Introduction

Complex networks and, especially, social networks often exhibit a finer internal structure where individuals interact in small subgroups (called communities or modules), based on the individuals' common interests, geographic location, political opinions etc. Understanding how such subgroups are structured and evolve in time is essential for applications like targeted advertising, viral marketing, friend suggestion etc. Social-network mining traditionally identifies a community as a densely connected set of nodes that is in turn only loosely attached to the rest of the network [9]; in this view, community detection translates into finding a partition of the nodes that optimizes some quality function. Most of the literature on this topic focused on the discussion of the mutual merits of various quality functions and on the comparison of algorithms that try to optimize (in an exact or approximate way) some of those functions. It is worth noticing that we are here thinking of the clustering problem in a situation where the only available information is the (directed or undirected) graph underlying the social network, possibly with some weights on its arcs denoting the strength of that bound[1].

The main limit of the approach discussed above is that rarely a node is part of a single community: more often than not, communities overlap giving rise to a complex intertwining that can hardly be reflected into a node partition. For this reason, recent research (see, for example, [2], [17]) has turned its attention to the problem of finding overlapping communities, where each node can be a member of more than one module.

This idea is well motivated and neat for those (frequent) situations in which membership to multiple communities is an exception more than a rule, and most nodes belong clearly to one single communities, with a number of borderline individuals for whom membership is less straightforward. In a large number of scenarioes, however, belonging to more groups is a rule more than an exception, and actually the notion of node community hardly makes sense: like a point in the Cartesian plane belongs to infinitely many lines, an individual in a social network plays potentially infinitely many roles. In those cases, it is often more sensible and interesting to individuate *communities of arcs* rather than *communities of nodes*: this shift of interest (witnessed in the most recent literature [27]) can be thought of as trying to find the reasons behind relations rather than trying to find the reason behind individuals. Or, going on with our metaphor, it is like determining the line to which *two* given points belong—a single point lies on infinitely many lines, but there is only a single line passing through two given points.

This idea is clear if one thinks of social networks such as Facebook: every Facebook user has probably many interests and belongs to a multiplicity of communities; however, every friendship is probably due to one main reason (working together, being relatives, having the same hobby etc.). This thought is so natural that Google+ has explicitly introduced the notion of "circle", later adopted also by Facebook.

In this work, we propose to continue along this line of research trying to exploit the following simple observation: if xy and yz are two relations that have the same motivation (e.g., working together), then probably xz will also be present: in other words, triangles tend to live inside communities. Based on this intuition, we propose the notion of triangular random walk, a stochastic process that treats differently triangular and non-triangular arcs; although this process is not memoryless, we can reduce it to a standard Markov chain on the line graph (using a tool similar to [8], but in a different way). With our approach, we obtain a weighted version of the line graph (a graph whose nodes correspond to the arcs of the original network). The weighted line graph can in turn be clustered using standard tools, hence employing state-of-the-art algorithms for the actual clustering phase: the main

Partially supported by a Yahoo! faculty grant and by by the EU-FET grant NADINE (GA 288956).

[1]Even if other information about vertices and edges may be available, it is usually computationally unfeasible to leverage it to detect communities.

978-0-7695-4839-5/12 $26.00 © 2012 IEEE
DOI 10.1109/LA-WEB.2012.19

limit of this approach is that the line graph is itself some orders of magnitudes larger than the original graph, so even its construction can become a computational burden (let alone the time and resources that the clustering algorithm will then require). For this reason, we develop an *ad hoc* version, called ALP, of a well-known clustering technique that carries out the clustering on the weighted line graph without having to compute it explicitly. Experiments on real-world networks of different sizes and types show that triangular walks can be extremely helpful in finding meaningful communities, outperforming significantly all other approaches; moreover, ALP turns out to be very efficient and can be used on large networks for which all other approaches would be prohibitive.

To summarize, the main contributions of this paper are: a) the definition of two weighting schemes (called w_T and v_T in this paper) for the arcs of the line graph that allow one to individuate arc-communities in the underlying graph; b) a clustering algorithm (ALP) that is able to use such schemes without the need to compute the line graph explicitly; c) a series of experiments proving that the weighting schemes proposed produce a significant improvement over all known techniques (in terms of quality, independently of the clustering algorithm adopted), and that ALP in itself can obtain the same results much more efficiently; in fact, it is the fastest and most scalable among all arc-clustering algorithms we are aware of.

II. Triangular random walks

Given a (directed) graph $G = (V_G, A_G)$ with no self-loops, we let $n_G = |V_G|$ and $m_G = |A_G|$ be the number of nodes and arcs of G, respectively; for every node x we let $N_G(x)$ be the set of *successors* of x and $d_G(x) = |N_G(x)|$ (the *(out)degree* of x). If G is symmetric (i.e., undirected), we use the term *edge* to refer to an unordered pair of nodes that are connected by an arc. We sometimes write xy to denote the arc (x, y) (or the edge $\{(x, y), (y, x)\}$, if the graph is undirected).

A *random walk* on a directed graph G is a stochastic process $X_0, X_1, \ldots$ where $X_0, \cdots \in V$, and for each $x, y \in V$, $P[X_0 = x] = 1/n$ and $P[X_{t+1} = y \mid X_t = x]$ is $1/d(x)$ if $y \in N(x)$, 0 otherwise[2]; this definition can be easily extended to positively weighted graphs (making $P[X_{t+1} = y \mid X_t = x]$ proportional to the weight of (x, y)). Intuitively, a random walk describes the behavior of a surfer wandering in the graph, who starts from a random node and at each step chooses uniformly at random (or proportionally to the weights) among the successors of the current node (jumping to a random node if the current one has no successors).

The random walk is a Markov chain and if G is undirected, connected and not bipartite, then the random walk has a unique stationary distribution $\mathbf{v}$ with $v_x = d(x)/2m$ [22]. For a general graph, however, the random walk is not ergodic, hence the stationary distribution may not be unique; to circumvent this problem, one can introduce [4], [13], [24] the notion of restart.

[2]For the sake of completeness, when $d(x) = 0$ we let $P[X_{t+1} = y \mid X_t = x] = 1/n$ for all y.

For a fixed $\alpha \in [0, 1]$, a *random walk with restart with damping factor* α on G is a stochastic process $X_0, X_1, \ldots$ as before, but where the surfer chooses the next node as follows: with probability α she picks a node uniformly at random among the successors of the current node; with probability $1 - \alpha$, instead, she jumps to a random node in the graph[3]. The latter event is called *teleportation* or "restart". It can be shown [4] that for all $\alpha < 1$ the random walk with restart has a unique stationary distribution (actually, the PageRank of G with damping factor α); when $\alpha = 1$ we get back to the standard random walks, instead.

One suggestive way to think of this random process is the following: a random surfer is trying to collect some knowledge and every node represents an expert that may provide some piece of information. After the surfer has finished visiting expert x she receives a list of other possible people that x trusts; the surfer may decide (with probability α) to accept x's suggestion and to visit one of them, or may rather decide to do it her way and to teleport to a random expert instead.

It is interesting to observe that one may also actually consider the stationary distribution *on the arcs of* G: the probability $P[X_t = x, X_{t+1} = y]$ that the random surfer goes along the arc (x, y) is $P[X_{t+1} = y \mid X_t = x]P[X_t = x] = v_x(\alpha w(x, y) + (1 - \alpha)/n)$, where $\mathbf{v}$ is the stationary distribution on the nodes and $w(x, y)$ is the weight on the arc (x, y) (that is, $1/d(x)$ in the unweighted case). We will refer to this distribution as the *arc-stationary distribution*.

The main idea of this paper is that we want to introduce a bias in the behavior of the random surfer, by allowing her some amount of short-term memory; in particular, the choice of the next node will not depend only on the current node but *also on the previous one*. The bias is finalized to privilege (or punish) triangles, i.e., suggestions of the current node that were also suggested by the previous node. Whether we decide to privilege triangles or to punish them depends on our interpretation of triangles: if we think that the double suggestion reinforces the idea that the suggested node is reliable, we will privilege triangles; if otherwise we suspect that the double suggestion is rather a form of lobbying, we will tend to avoid triangles.

Thus, we will define a triangular random walk $X_0, X_1, \ldots$ on an *unweighted*[4] graph using two parameters, $\alpha, \beta \in [0, 1]$: α is a damping factor and will have the same meaning as before (it is used to decide whether to follow a link or to teleport); β will instead be used to determine whether triangles or non-triangles should be privileged.

Two subtly different definitions of triangular random walks can be given, depending on the specific meaning of β: we will call them mass-triangular and ratio-triangular, respectively. In a triangular random walk with parameters α and β, the next node (x_{t+1}) is chosen depending on the current node x_t and

[3]As before, if the current node has no successors then the next node is chosen at random among all nodes in the graph.

[4]As before, extending this notion to weighted graphs is trivial, but for the sake of readability in this paper we prefer to limit ourselves to the unweighted case.

on the previous node x_{t-1}, as follows: (i) with probability $1-\alpha$, we teleport: x_{t+1} is a randomly chosen node; (ii) otherwise, we choose among the successors $N(x_t)$ of the current node, but treating differently the *triangular successors* (the set $N(x_t) \cap N(x_{t-1})$) and the *non-triangular successors* (the set $N(x_t) \setminus N(x_{t-1})$)[5]; here, the two definitions differ: in the *(mass-)triangular random walk*, we first decide whether we shall select a non-triangular successor (with probability β) or a triangular one (with probability $1-\beta$); then, the specific non-triangular or triangular successor is chosen uniformly at random; in the *ratio-triangular random walk*, all triangular successors are selected with the same probability, say p, and all non-triangular successors with probability βp (p should be chosen so that the sum of such probabilities is 1).

The names we adopted for the two kinds of random walks should be evocative of the meaning of β: in the mass-triangular random walk, β is the overall amount of probability of choosing a non-triangular successor; in the ratio-triangular random walk, it is the ratio between the probability of choosing a(ny) non-triangular successor over the probability of choosing a(ny) triangular one.

The two kinds of processes coincide when $\beta = 0$ (in that case, they both only choose triangular successors, except when teleporting). Moreover, ratio-triangular random walks reduce to standard random walks with restart when $\beta = 1$ (because, in that case, the probability of choosing triangles and non-triangles is the same), whereas there is no choice of β that makes a mass-triangular random walk the same as a standard random walk. The latter observation may suggest that ratio-triangular random walks should be preferred, but the mathematical treatment of mass-triangular walks is simpler, and for this reason we shall actually treat the latter as our "default" type of triangular walk (and omit "mass" in the following). Triangular walks can have a number of potential applications; for example, they may be used fruitfully in bibliometrics to moderate the problem of nepotistic citations in scientific works (in this case, triangles should be punished rather than promoted). In this paper, however, we wish to speculate on the possible usage of triangular walks to single out arc-communities in social networks, where triangles are used as a form of reinforcement.

To start playing with our idea, let us consider Zachary's famous karate club network [28]: this is an undirected graph whose nodes represent the members of a karate club and with an edge between two individuals if they happened to have seen each other outside of the club for some reason; the club ended up splitting in two (in our drawings, the nodes are depicted differently according to the group they will end up in), and one can hope to find information about how the members will decide to group based solely on their friendship relations. We first tried a standard random walk on this dataset to see how frequently each edge was run through in either direction (Figure 1): no pattern is evident. But if we do the same with a triangular walk some edges get more emphasis, witnessing that some bounds are stronger than others (Figure 2, with $\beta = 0.2$): those edges are usually between members that will end up in the same group (with an exception concerning node 9 that indeed seems to be more strictly bound to the group of circles than to the group of squares). If we decrease β to 0.01, some clans would become almost grotesquely evident.

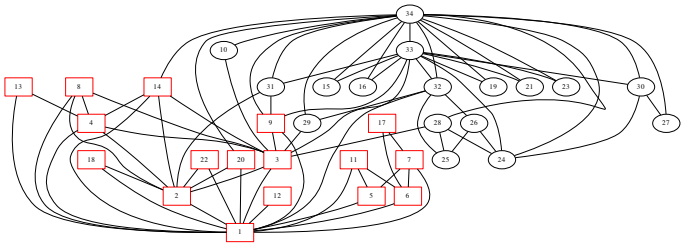

Fig. 1. Standard random walk on the karate club dataset; edge width is proportional to the frequency with which that edge was run through in either direction.

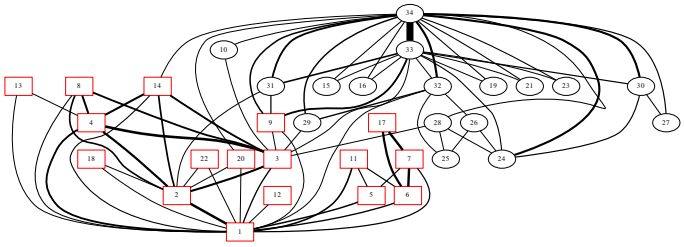

Fig. 2. Triangular random walk on the karate club dataset, with $\beta = 0.2$ (see also Figure 1).

A. Triangular walks and line graphs

A triangular random walk is a Markov chain of order 2 [22], because the next state depends on the current state *and* on the previous one. To study the long-term behavior of higher order chains, it is customary to change the state space and reduce the stochastic process to an equivalent one that is memoryless; this is easily solved by using the notion of *line graph*.

Given a graph G, its line graph $L = L(G)$ has the arcs of G as vertices (i.e., $V_L = A_G$), and arcs of the form (xy, yz) (where xy and yz are two arcs of G). Note that even when G is symmetric, $L(G)$ is not; for example, if G is the undirected graph in Figure 3, its corresponding line graph $L(G)$ is represented in Figure 3 (for the time being, ignore the colors on its arcs). The idea of using line graphs to study the behavior of an arc-aware random surfer was already proposed in [8], but they adopt a subtly different notion of line graph that is undirected; for our purposes, instead, the directed definition is much more well-suited (also because it adapts readily to the case when the original graph is itself directed).

[5]If either set is empty (or if $t = 1$) we choose uniformly in $N(x_t)$ (or in V, if the latter is empty), as in a standard random walk. The rationale behind this choice is that, based on the knowledge that we have (the current node and the previous one), all the outgoing arcs are equivalent.

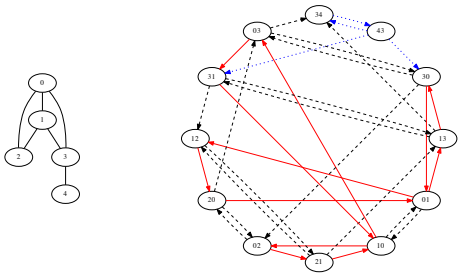

Fig. 3. A small undirected graph G (left) and the corresponding line graph $L(G)$. Continuous (red) arcs correspond to choosing triangular successors; dashed (black) arcs correspond to the choice of non-triangular successors; dotted (blue) arcs are used for the cases where either set is empty.

Now, it is easy to see that a triangular random walk with parameters α, β on the (unweighted) graph G is equivalent to a random walk with damping factor α on the weighted line graph $L(G)$, where

$$w_T(xy, yz) \triangleq \begin{cases} \frac{1-\beta}{|N(y)\cap N(x)|} & \text{if } z \in N(y) \cap N(x) \\ \frac{\beta}{|N(y)\setminus N(x)|} & \text{if } z \in N(y) \setminus N(x). \end{cases} \quad (1)$$

In other words, every arc in $L(G)$ (that is to say, every two-step walk $x \to y \to z$ in the original graph) has a different weight depending on whether it can be closed by a triangle (i.e., if $x \to z$ was also an arc of G) or not. If you look again at Figure 3, continuous (red) arcs correspond to the first case (e.g., $10 \to 03$ is one such arc, because 13 is also an arc of G), whereas dashed (black) arcs correspond to the second case (e.g., $31 \to 12$); note, in particular, that all arcs of the form $xy \to yx$ fall in the second class[6]. Some nodes of $L(G)$ (i.e., arcs of G) require some care, because their outgoing arcs are all non-triangular; those outgoing arcs are hence not weighted using the formula above (it would not make sense since one of the denominators is zero), but they have a constant weight instead (such arcs are drawn as dotted (blue) arrows in Figure 3).

For $\alpha < 1$ the random walk with restart on $L(G)$ weighted by w_T has a stationary distribution $\mathbf{v}_T$: note that, since the nodes of $L(G)$ are arcs of G, $\mathbf{v}_T$ assigns a probability $v_T(xy)$ with each arc xy of the original graph. Note also that, as explained in the previous section, the stationary distribution on the nodes of $L(G)$ induces a stationary distribution on its arcs:

$$v_T(xy, yz) \triangleq v_T(xy)(\alpha w_T(xy, yz) + (1-\alpha)/n_{L(G)}). \quad (2)$$

This is the fraction of time that the random surfer walking on $L(G)$ with weights w_T spends on the path $x \to y \to z$, and can be used as way of weighting the graph $L(G)$ alternative to (1).

Computing the stationary distribution $\mathbf{v}_T$ is a well-understood task (it amounts to a weighted version of PageRank) for which efficient and computationally sound algorithms exist [13], [26]; of course, $L(G)$ is larger than G (it has m_G nodes and $\sum_x d_G(x)^2$ arcs), but not much larger actually because of the sparsity of G and of the way its degrees are distributed. In particular, if G is undirected and has $\approx Ck^{-\alpha}$ nodes of degree k, then $L(G)$ will have $\approx C^2k^{-2\alpha}$ nodes of outdegree k.

[6]Differently from [8], we do not reserve stuttering walks (walks of the form $x \to y \to x$) a special treatment.

III. Arc-clustering via triangular random walks: a) using an off-the-shelf algorithm

As outlined in the previous sections, along the same line as [8], instead of clustering directly the arcs of G (as done, for example, by [12]), we turn to some suitably weighted version of the line graph $L(G)$, where we can make good use of all the paraphernalia for node-clustering of a directed graph. In other words, we can use an off-the-shelf node-clustering algorithm feeding it with the weighted (directed) graph $L(G)$. As weighting function (on the arcs of $L(G)$), we can use either of the weighting schemes defined in (1) and (2). For comparison, we may consider the weights of a standard random walk $w_S(xy, yz) = 1/d(y)$ or the corresponding arc stationary distribution $v_S(xy, yz)$ (as before, $v_S(xy)$ is the stationary distribution of the standard random surfer on the node xy); here, the subscript "S" stands for "standard". Another baseline is to feed the clustering algorithm with the unweighted graph $L(G)$ itself.

The main limit of the proposed method is that it cannot be directly applied to truly undirected graphs: since it is designed for directed graphs, reciprocal arcs (i.e., parallel arcs in opposite directions) may end up in two different communities. In cases when this fact can be a problem, one has to decide what to do about reciprocal arcs that happened to be clustered differently—one possible solution is to place the corresponding edge in either community, or to use a special community that corresponds to the given pair.

a) Computational issues: Computing the line graph $L(G)$ and its weights w_T is straightforward and can be performed in time $O(m_{L(G)})$ (i.e., linear in the output size), provided that one has direct access to G; moreover, although their size is obviously larger than the original graph (see Table I), line graphs turn out to be easily compressible (about 2 to 3 bits/link in their natural order, much less if suitably permuted [5]). After $L(G)$ has been produced, weighted PageRank can be computed very quickly (using for example the techniques of [7]), and in our experiments always resulted to converge in less than 20 iterations even for $\alpha = 1-10^{-2}$. The final node-clustering phase clearly depends on the algorithm used, but our method of choice [3] turns out to be reasonably fast — actually, the line graph construction is almost as expensive as the clustering itself. In fact, the explicit construction of the line graph is the main limit of this approach, especially for networks that are comparatively denser (such as Hollywood).

	n_G	$m_G = n_{L(G)}$	$m_{L(G)}$
free word assoc.	10 225	71 679	955 552
DBLP	986 324	6 707 236	211 808 396
Hollywood	2 180 759	228 985 632	242 026 293 162

TABLE I
SIZE OF LINE GRAPHS FOR SOME OF THE DATASETS WE SHALL USE IN SECTION VI; OBSERVE THAT HOLLYWOOD IS COMPARATIVELY DENSER THAN THE OTHER GRAPHS (WITH AN AVERAGE DEGREE OF ABOUT 105), WHICH IS WHY THE NUMBER OF ARCS IN $L(G)$ IS SO LARGE (THE AVERAGE DEGREE IS IN THIS CASE 1 057).

IV. ARC-CLUSTERING VIA TRIANGULAR RANDOM WALKS: B) USING ALP

When the graph is comparatively denser having to compute explicitly $L(G)$ can become a serious limitation; nonetheless, there is conceptually no need to do so—the graph $L(G)$ might be handled *implicitly*. If we want to approach the problem this way, however, we need to develop a specially tailored clustering algorithm that mimics what it would do on the (weighted version of) $L(G)$ without having it represented explicitly.

We tackled this idea by writing an implementation of the LP (Label Propagation) algorithm [18] that clusters the arcs of G based on an implicit representation of $L(G)$, weighted as in (1) or (2): we call this implementation ALP (for "Arc Label Propagation"); the reason behind the choice of LP with respect to other clustering algorithms is that it provides a good compromise between quality and speed. Moreover, due to its very diffusive nature, LP is best suited to translate into an algorithm that implicitly propagates information on the line graph. ALP takes G as input and works almost exactly as a standard LP [18] would do if run on $L(G)$, with the following adjustments: (a) LP is natively intended to be run on unweighted graphs, and it is based on a diffusive process where each node (arc, for ALP) decides whether to change its own label based on the majority of the labels in its neighboring nodes (arcs, for ALP); our adaptation to weighted graphs just changes the way majority is computed (summing up weights of neighbors instead of counting them); (b) Since LP is designed for undirected graphs, ALP actually considers the symmetrized version of $L(G)$ when being executed; in other words, an execution of ALP on G is equivalent to an execution of LP on a symmetrized weighted version of $L(G)$.

A final remark is that, if ALP is to be run with the weights v_T of (2), a preliminary computation of weighted PageRank on $L(G)$ should be performed; also this step can be carried out implicitly, without ever having to deal with $L(G)$.

V. RELATED WORK

Although node-clustering is traditionally much more developed and better understood (see [21] for an up-to-date survey), recently many authors advocated the adoption of link-clustering [27], [8], [12] as a way to overcome the problem of overlapping communities in complex networks. The advantage of this approach over the solution of soft or hierarchical node-clustering [15], [11] is that the latter is better suited for situations where the presence of a node in many communities is an exception rather than a rule; on the contrary, using link-clustering allows one to give multiple membership a more understandable meaning in the common situations when every single node is likely to belong to more than one cluster but each node-to-node relation can be explained as co-affiliation to some community (like in the well-known model of affiliation networks [14]). Of course, even in the latter situation co-affiliation can be due to many reasons (co-affiliation to many communities), one reason usually prevails.

The usage of line graphs to model link-clustering is especially promoted by Evans and Lambiotte [8] (who also take into consideration notions of weighting that deal with the problem of over-representing high-degree nodes), but they exploit the undirected version of line graphs instead of the directed one [10], and they do not distinguish between triangular and non-triangular arcs. It should be noted that the roles of (open and closed) triangles in social networks is well known and studied in the realm of SNA, under the name of *triads* [25].

As explained, our technique relies on some external node-clustering algorithm that uses a weighted version of $L(G)$, with the hope that triangular random walks highlight clear cuts between communities as they should. To test our hypothesis, we obviously need a clustering algorithm that can handle large weighted directed graphs; we tried three different clustering algorithms which satisfy our requirements and are considered the state of the art for massive complex networks: clustering via Potts' model as proposed in [19], the hierarchical Infomap algorithm presented in [20] and the Louvain method [3]. In our tests the latter proved to be the fastest among these candidates and produces also the best results in term of accuracy, so we will adopt it in our experiments. Actually, however, all the tested methods improve their performance on the versions of $L(G)$ that were weighted according to our criterion.

VI. EXPERIMENTS

The experiments that we are going to describe have been run using public datasets and relying heavily on the WebGraph [6] framework (in particular, the line-graph transformation was implemented as a part of it). The remaining tools are available as "Satellite Software" in the http://law.dsi.unimi.it/ website. In most of the experiments, we shall need a way to evaluate the clustering quality. More precisely, we suppose to be given a graph G with a measure of similarity σ between its arcs. The output of an arc-clustering algorithm is going to be a labelling function λ providing a label for every arc xy of the input graph. To evaluate the quality of the given arc-clustering λ with respect to the similarity σ, we shall use a variant of the Probabilistic Rand Index (PRI) [23]:

$$\mathrm{PRI}(\lambda,\sigma) = \sum_{\lambda(xy)=\lambda(x'y')} \sigma(xy,x'y') - \sum_{\lambda(xy)\neq\lambda(x'y')} \sigma(xy,x'y').$$

The cost of evaluating this quantity is prohibitive (quadratic in the number of arcs), thus we shall instead estimate its value by sampling pairs of arcs $\{xy, x'y'\}$ according to the following criteria: (i) the two arcs xy and $x'y'$ are sampled uniformly

at random (PRI_u); (ii) a node $x = x'$ is chosen uniformly at random, and we select two of its successors y and y' again at random (PRI_n); (iii) a node $x = x'$ is chosen at random proportionally to its degree, and we select two of its successors y and y' at random (PRI_d). While the first is an unbiased estimator of the PRI, the latter two aim at providing a more fine-grained understanding of the local quality of the clustering obtained. PRI is a quality measure that indirectly takes the number of clusters into account: an excessive fragmentation, for example, will produce bad PRI values, because similar arcs that are put in different clusters contribute negatively to the score. Nonetheless, we will also discuss the number of clusters obtained in our experiments.

b) Parameter tuning: For this set of experiments, we worked on the DBLP graph[7]; The DBLP graph is a scientific collaboration network where each vertex represents a scientist and two vertices are connected if they have worked together on an article. The current version (July 2011) of the DBLP dataset contains 986 324 authors and 2 684 847 publications, giving rise to 3 353 618 co-authorship edges. This network corresponds to the typical situation in which every author can belong to more than one scientific community (because typically, during their life, scientists work on many different and often scarcely related topics), but collaborations usually correspond to a specific topic. Based on this interpretation, we labelled each edge of DBLP with the concatenation of all titles of the co-authored papers, and the similarity between two edges is computed as the cosine distance between the corresponding term vectors (we normalized the words though a Porter's stemmer and used TF-IDF [1] for term weighting); this measure of similarity σ between edges is our ground truth.

In this experiment we used the weights v_T of (2) computed with different values of α and β to see how they impact on the quality of the clustering obtained with respect to similarity; we used ALP as a clustering algorithm, but in our experiments it seems that parameter can be tuned pretty much independently from the clustering algorithm employed. Most probably, it depends instead from the type of social network considered (e.g., as observed, whether triangles should be promoted or demoted); in all the graphs we are using here, however, the behavior was the same.

In Figure 4 we show the values of PRI_u for different combinations of α and β; we did a similar evaluation for PRI_n, PRI_d and for the number of communities obtained (the corresponding graphs are not shown).

- For $\alpha = 0$, the weights v_T of (2) become constant and the behavior of the clustering algorithm degrades (for the sake of readability, this is not shown in the figure);
- As long as $\alpha > 0$, its value does not seem to impact much on the local quality measures (PRI_n, PRI_d) but the overall quality PRI_u decreases for large α's: our interpretation for this behavior is that larger values of α produce a more fragmented clustering (as also witnessed by the number of communities obtained) because infrequent teleporting reduces transitivity.

As far as β is concerned, small values of β (i.e., more importance to triangles) produce the best results. As a rule of thumb, we think that α should be taken small (in the remaining experiments, we set $\alpha = 0.1$) at least for sparse networks; on denser graphs, larger values of α can be a better option to avoid that few communities flood all the arcs. As for β, we used $\beta = 0.01$ in our experiments, but the actual value should be adapted to the specific network under examination, as already discussed.

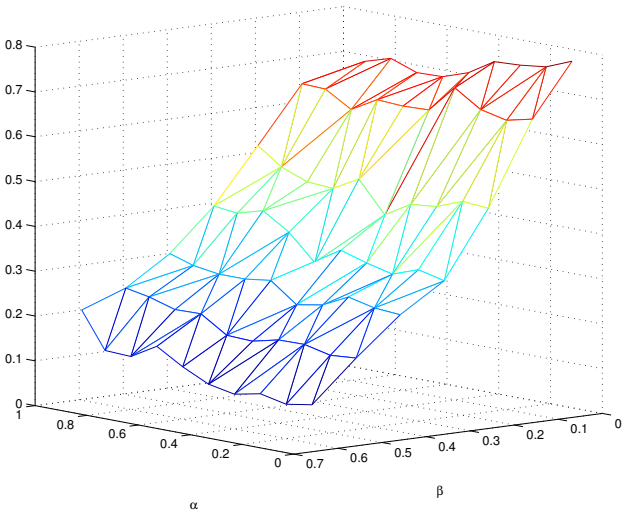

Fig. 4. PRI_u computed on DBLP (using the weights v_T of (2) and ALP for clustering) as a function of α and β.

c) Quality: We then faced the problem of directly evaluating the clustering quality, for the values of the parameters determined above ($\alpha = 0.1$ and $\beta = 0.01$). We performed our experiments on DBLP and on the Hollywood graph: the latter was obtained from the Internet Movie Database[8]; this undirected graph has, in its current version (July 2011), 2 180 759 nodes (actors and actresses) and 114 492 816 edges corresponding to having acted together in some movie. Here the edge xy is labelled with the multiset of directors that directed the movies co-acted by x and y, with the interpretation that a specific actor may have worked in many different movies, but directors tend often to collaborate with the same set of "trusted" actors. Similarity between arcs is once again computed using TF-IDF (here, the vocabulary is made by director IDs); the idea, this time, is to individuate the "clans" that typically pop up in the film industry around the figure of most directors. Note that, again, this idea would not fit with node clustering (because an actor is often part of more clans, but typically co-actorship individuates a clan in a quite specific way).

For this set of experiments, and for each of the two networks, we clustered the arcs in various ways (see below).

[7]http://www.informatik.uni-trier.de/~ley/db/.

[8]http://www.imdb.com/.

We considered the following combinations:

- we tried our weighting schemes w_T and v_T of (1) and (2), and for comparison the standard random surfer weights w_S and v_S (see Section III), as well as the unweighted version;
- for each weighting scheme above, we used two clustering algorithms: ALP (Section IV), that is fed directly with G (and computes $L(G)$ and its weights only implicitly) and the Louvain [3] algorithm, that is given the weighted version of $L(G)$ instead ([3] clusters the nodes of an arc-weighted graph);
- as baseline, we tried to cluster the arcs using the system proposed by [8] (that works on the undirected version of the link graph) and *LINK*, a link clustering technique proposed in [27][9]; both algorithms are specifically aimed at arc-clustering so they are the natural competitors of our method; unfortunately (as better explained below) we could run them only on the smallest of the two datasets, because of their lack of scalability;
- finally, as further baseline, we tried to cluster the arcs indirectly, through some of the best node clustering techniques; we transform a node clustering into an arc clustering with the following strategy: since a node clustering algorithm produces a labeling function $f : V_G \to \mathbb{N}$, we map each arc xy to the pair $(f(x), f(y)) \in \mathbb{N}^2$, and use the latter as arc label. If the original graph is symmetric, we can forget about the order of labels and assign an unique identifier to each unordered pair of labels.

The results obtained for DBLP [10] are shown in Table II, along with the computation time[11]: when using the Louvain [3] algorithm, we highlight the pre-computation time required to produce the weighted line graph to be fed to the algorithm; note also that for the PageRank-based weights v_T, there is some pre-computation time needed to obtain the PageRank vector (this is true also for ALP). As for Hollywood, the only arc-clustering method that can be applied is ALP and the results obtained are also shown in Table II—building explicitly the line graph is out of question and anyway it would be far too large to be handled by (the current implementation of) [3]; hence, our only baseline is Louvain run on the base graph G (we did not get any result from Infomap on the base graph, and we decided to stop it after 60h). Some comments are in order:

- Our weighting schemes aim at capturing local communities more than global ones, and indeed the local measures of quality (PRI_n and PRI_d) we obtain outperform significantly all other approaches; the best competitors, that still do not quite reach the same results, are Evans et al. [8] and LINK [27]. Both, however, do a rather poor job when the results are considered globally, but for opposite reasons: [27] seems to fragment the communities too much (many of them constitute of a single arc), whereas [8] produces too few communities (putting together too many "dissimilar" arcs). Apparently this problem presents itself also when we use our weighting scheme with [3], whereas ALP is able to produce a more balanced output, giving good results even on a global scale.
- Comparing our results with all the node-oriented approaches, it seems clear that arc-communities have a much more distinct structure than node-communities in the networks we examined.
- As far as the difference between the two types of weights, the gain in using the arc-stationary state v_T instead of the simple triangular weights w_T is marginal; yet PageRank computation is so fast that the effort is anyway worth.

d) Karate club (revisited): To visually appreciate the results of our clustering technique, we tried it on the karate club dataset; we set $\alpha = 0.1$ as usual, but this time the density of the network suggests using a larger β than we did with the other graphs. Figure 5 shows the outcome obtained for $\beta = 0.2$ (smaller values of β tend to fragment the network too much). The algorithm finds 6 communities, but two of them (the red and green arcs) are definitely dominant and correspond largely to the edges between homogeneous members. The two second-largest communities, in blue and violet, are rather dense internally but poorly linked to the other nodes. For comparison, in Figure 5 you can see the same network clustered with LINK, that individuates 22 communities.

e) Clustering of the word association network: For this experiment, we considered the Free Word Association network [16]; this is a directed graph describing the results of an experiment of free word association performed by more than 6 000 participants in the United States: its nodes correspond to words and arcs represent a cue-target pair (the arc xy means that the word y was output by some of the participants based on the stimulus x). This graph contains 10 617 words and 71 176 associations (arcs). We used ALP and Louvain to cluster it according to our two schemes (as usual, we set $\alpha = 0.1$ and $\beta = 0.01$). For comparison, we considered also the communities found by Evans et al. [8] and by LINK [27] on the same graph. In this case we do not have any ground truth to compare to, hence our analysis can only be based on some preliminary observations.

The number of communities found by ALP is 7 070 with w_T and 7 221 with v_T, showing that the use of PageRank tends in this case to obtain slightly smaller communities (the average size passes from 10.06 to 9.86). As for the other methods, [8] produces only 33 huge communities (the average size is 2 157), whereas [27] fragments the graph into 43 182 communities (the average size is 1.65).

An interesting observation is that of the 8 384 reciprocal arcs (an arc xy is reciprocal if also yx is an arc, the 11.8% of

[9]We used the LINK Python implementation that automatically optimizes its parameters. We also experimented with the software described in [12], but could not have it work on networks of more than about 100 nodes.

[10]All tests on DBLP were run only on the giant component of the graph because some of the baseline algorithms (in particular, LINK) requires the input graph to be connected; we verified, however, that the quality obtained by ALP is consistently the same even outside of the giant component.

[11]All experiments were performed on a Linux server equipped with Intel Xeon X5660 CPUs (2.80 GHz, 12 MB cache size) for overall 24 cores and 128 GB of RAM.

			clusters	PRI_u	PRI_n	PRI_d	computing time
DBLP	ALP (Section IV)	$\mathbf{v_T}$	613 203	**0.74**	**0.71**	**0.75**	1s+32s
		$\mathbf{w_T}$	592 562	**0.72**	**0.75**	**0.75**	32s
		v_S	48 025	0.02	0.16	0.18	24s
		w_S	38 498	0.02	0.08	0.03	22s
		-	38 498	0.02	0.08	0.03	22s
	Louvain [3]	$\mathbf{v_T}$	1 493	**0.01**	**0.69**	**0.53**	157s+337s
		$\mathbf{w_T}$	2 116	**0.02**	**0.71**	**0.53**	122s+334s
		v_S	230∗	0.01	0.44	0.39	137s+943s
		w_S	232	0.01	0.43	0.39	114s+914s
		-	250	0.01	0.16	0.15	92s+224s
	Evans et al. [8]	-	200	0.01	0.58	0.44	46min
	LINK [27]	-	1 415 245	0.28	0.31	0.51	50h
	Infomap [20]	-	62 680	0.05	0.27	0.29	874s
	Louvain (on G) [3]	-	6 442	0.01	0.28	0.28	13s
Hollywood	ALP (Section IV)	$\mathbf{v_T}$	383 780	**0.80**	**0.78**	**0.56**	1h+16h
		$\mathbf{w_T}$	424 094	**0.77**	**0.71**	**0.48**	13h
		v_S	255 247	0.00	0.03	0.03	3h
		w_S	277 859	0.00	0.02	0.01	3h
		-	277 859	0.00	0.02	0.01	3h
	Infomap [20]	-	-	-	-	-	> 60h
	Louvain (on G) [3]	-	23 807	0.01	0.18	0.19	242s

TABLE II

CLUSTERING QUALITY OBTAINED USING DIFFERENT TECHNIQUES ON THE DBLP AND HOLLYWOOD GRAPHS (IN BOLDFACE, THE TWO TRIANGULAR WEIGHTS SUGGESTED IN THIS PAPER, USING $\alpha = 0.1$ AND $\beta = 0.01$). THE UPPER GROUP REFERS TO THE APPLICATION OF THE ALP OR LOUVAIN ALGORITHM TO VARIOUS (WEIGHTED OR UNWEIGHTED) VERSIONS OF $L(G)$ (IN THE CASE OF LOUVAIN, THE LINE GRAPH MUST BE EXPLICITLY BUILT); THE MIDDLE GROUP CONSISTS OF ALGORITHMS THAT PRODUCE AN ARC-CLUSTERING ON G; THE BOTTOM GROUP, INSTEAD, PRODUCE A NODE-CLUSTERING ON G, THAT WE INTERPRET AS AN ARC-CLUSTERING.

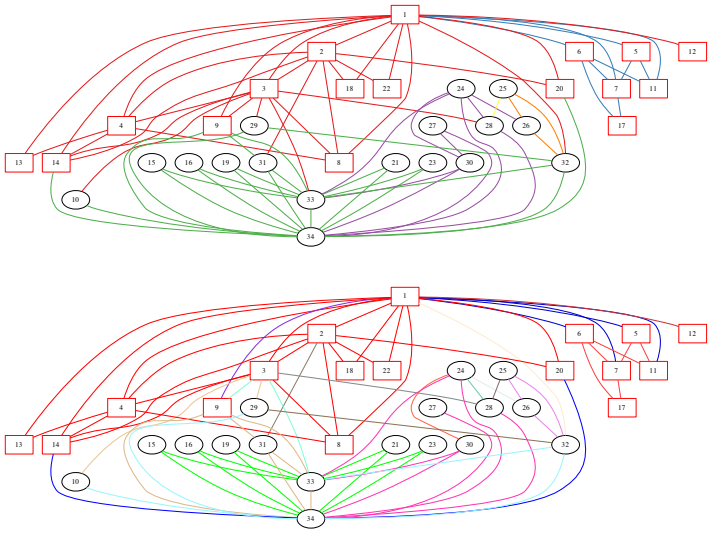

Fig. 5. Clustering of the karate club dataset: (top) using triangular weights (v_T) and the ALP clustering algorithm, (bottom) using LINK [27].

the arcs are such in this graph), only a minority are assigned by ALP the same label in the two directions (3 038 for w_T, 3 028 for v_T): this witnesses the fact that ALP does not behave "as if" the graph was symmetric.

On a purely anecdotal base, we present in Figure 6 the subgraph induced by the word "KEYBOARD" and its successors, as it is clustered by Louvain with v_T (top) and by Evans et al. [8] (bottom); note that the algorithm used is actually the same, so the difference is only in the weighting scheme. Although there is clearly a group of successors that are related to music and another one that is related to computers, [8] puts all arcs going out of "KEYBOARD" in the same community (even if the community of computer-related word is in fact recognized, because the internal arcs connecting the three words "COMPUTER", "TYPEWRITER" and "TYPE" have a different colour than the other ones). With our weighting scheme the arcs going toward the music group are clearly separated from the other.

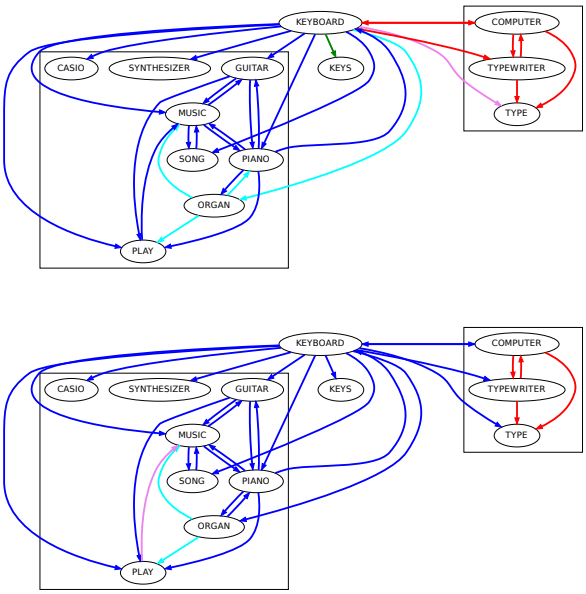

Fig. 6. Clustering of the word association network (subgraph around "KEYBOARD"): (top) using Louvain with v_T as weighting scheme, (bottom) using Evans et al. [8].

VII. Conclusions and Future Work

In this paper, we introduced a new kind of random process that helps in singling out arc communities in social networks; this can be seen as a Markov chain on the line graph whose arc-stationary state contains a big deal of information on the communities, and can be fruitfully used to gain a more accurate and fine-grained resolution, at least at a local level. In our experiments, using this information ended up in producing more reasonable and significant clusters, with a limited computational cost. These results are preliminary but very encouraging; we also believe that the weights proposed here can be beneficial for other types of mining tasks. Such tasks can be made reasonably scalable by exploiting the possibility (here explored with ALP) of writing implicit versions of mining algorithms that work on the weighted line graph without having to build it explicitly.

Acknowledgements

We thank Hawoon Jeong, Youngdo Kim, Dario Malchiodi and Federico Pedersini for their help in preparing the paper.

References

[1] Ricardo A. Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[2] Jeffrey Baumes, Mark K. Goldberg, Mukkai S. Krishnamoorthy, Malik Magdon-Ismail, and Nathan Preston. Finding communities by clustering a graph into overlapping subgraphs. In *IADIS AC'05*, pages 97–104, 2005.

[3] V.D. Blondel, J.L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008:P10008, 2008.

[4] Paolo Boldi, Violetta Lonati, Massimo Santini, and Sebastiano Vigna. Graph fibrations, graph isomorphism, and PageRank. *RAIRO Inform. Théor.*, 40:227–253, 2006.

[5] Paolo Boldi, Marco Rosa, Massimo Santini, and Sebastiano Vigna. Layered label propagation: A multiresolution coordinate-free ordering for compressing social networks. In Sadagopan Srinivasan, Krithi Ramamritham, Arun Kumar, M. P. Ravindra, Elisa Bertino, and Ravi Kumar, editors, *Proceedings of the 20th international conference on World Wide Web*, pages 587–596. ACM, 2011.

[6] Paolo Boldi and Sebastiano Vigna. The WebGraph framework I: Compression techniques. In *Proc. of the Thirteenth International World Wide Web Conference (WWW 2004)*, pages 595–601, Manhattan, USA, 2004. ACM Press.

[7] Gianna M. Del Corso, Antonio Gullì, and Francesco Romani. Fast pagerank computation via a sparse linear system. *Internet Mathematics*, 2:118–130, 2004.

[8] T. S. Evans and R. Lambiotte. Line graphs, link partitions, and overlapping communities. *Phys. Rev. E*, 80(1):016105, Jul 2009.

[9] Santo Fortunato and Claudio Castellano. Community structure in graphs. In Robert A. Meyers, editor, *Encyclopedia of Complexity and Systems Science*, pages 1141–1163. Springer, 2009.

[10] R. L. Hemminger and L. W. Beineke. Line graphs and line digraphs. In L. W. Beineke and R. J. Wilson, editors, *Selected Topics in Graph Theory*, pages 271–305. Academic Press Inc., 1978.

[11] Chen Jianbin, Fang Deying, and Shi Tong. A graph partition-based soft clustering algorithm. In *Proceedings of the 2008 Second International Symposium on Intelligent Information Technology Application - Volume 02*, pages 572–577, Washington, DC, USA, 2008. IEEE Computer Society.

[12] Youngdo Kim and Hawoong Jeong. The map equation for link community. *CoRR*, abs/1105.0257, 2011.

[13] Amy N. Langville and Carl D. Meyer. Deeper inside PageRank. *Internet Mathematics*, 1(3):355–400, 2004.

[14] Silvio Lattanzi and D. Sivakumar. Affiliation networks. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, STOC '09, pages 427–434, New York, NY, USA, 2009. ACM.

[15] Bo Long, Mark Zhang, Philip S. Yu, and Tianbing Xu. Clustering on complex graphs. In *Proceedings of the 23rd national conference on Artificial intelligence - Volume 2*, pages 659–664. AAAI Press, 2008.

[16] D. L. Nelson, C. L. McEvoy, and T. A. Schreiber. The university of south florida word association, rhyme, and word fragment norms. http://www.usf.edu/FreeAssociation/, 1998.

[17] Gergely Palla, Illes J. Farkas, Peter Pollner, Imre Derenyi, and Tamas Vicsek. Directed network modules. *New J.Phys.*, 9:186, 2007.

[18] Usha N. Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 76(3), 2007.

[19] Peter Ronhovde and Zohar Nussinov. Local resolution-limit-free potts model for community detection. *Phys. Rev. E*, 81(4):046114, Apr 2010.

[20] Martin Rosvall and Carl T. Bergstrom. Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems. *PLoS ONE*, 6(4):e18209, 04 2011.

[21] Satu Elisa Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007.

[22] E. Seneta. *Non-negative matrices and Markov chains*. Springer–Verlag, New York, 1981.

[23] Ranjith Unnikrishnan and Martial Hebert. Measures of similarity. In *7th IEEE Workshop on Applications of Computer Vision / IEEE Workshop on Motion and Video Computing (WACV/MOTION 2005)*, pages 394–400. IEEE Computer Society, 2005.

[24] Sebastiano Vigna. Spectral ranking, 2009.

[25] Stanley Wasserman and Katherine Faust. *Social network analysis: Methods and applications*. Cambridge Univ Press, 1994.

[26] Wenpu Xing and Ali Ghorbani. Weighted pagerank algorithm. *Communication Networks and Services Research, Annual Conference on*, 0:305–314, 2004.

[27] Sune Lehmann Yong-Yeol Ahn, James P. Bagrow. Link communities reveal multiscale complexity in networks. *Nature*, 466(7307):761–764, August 2010.

[28] W. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33:452–473, 1977.