Efficient community detection of network flows for varying Markov times and bipartite networks

Masoumeh Kheirkhahzadeh,[1,2] Andrea Lancichinetti,[2] and Martin Rosvall[2,*]

[1]*Department of IT and Computer Engineering, Iran University of Science and Technology, Teheran, Iran*

[2]*Integrated Science Lab, Department of Physics, Umeå University, SE-901 87 Umeå, Sweden*

(Received 4 November 2015; revised manuscript received 17 January 2016; published 9 March 2016)

Community detection of network flows conventionally assumes one-step dynamics on the links. For sparse networks and interest in large-scale structures, longer timescales may be more appropriate. Oppositely, for large networks and interest in small-scale structures, shorter timescales may be better. However, current methods for analyzing networks at different timescales require expensive and often infeasible network reconstructions. To overcome this problem, we introduce a method that takes advantage of the inner workings of the map equation and evades the reconstruction step. This makes it possible to efficiently analyze large networks at different Markov times with no extra overhead cost. The method also evades the costly unipartite projection for identifying flow modules in bipartite networks.

DOI: 10.1103/PhysRevE.93.032309

I. INTRODUCTION

Researchers often represent interactions between components in social and biological systems with networks of nodes and links, and use community-detection algorithms to better understand their large-scale structure. Depending on the system under study and the particular research question, the scale of interest varies. For an initial investigation, a bird's-eye-view of the entire system may be most appropriate, while a more detailed study most likely will require a finer scale. Methods for extracting hierarchically nested modules at different scales do exist [1,2], but there may still be a need for identifying large-scale structures at specific scales [3,4].

When the links represent network flows, modeling the dynamics at different Markov times is a natural way to capture the large-scale structures at different scales [5]. In this approach, the original network is rebuilt such that one flow step along a link of the rebuilt network corresponds to the desired number of flow steps on the original network. However, this approach is inefficient for large networks, because the rebuilt network can be dense to the degree that storage and further analysis is infeasible. To overcome this problem, we introduce an efficient method that operates directly on the original network. The method takes advantage of the mechanics of the information-theoretic community-detection method known as the map equation [6] with no extra overhead cost.

Integrating the Markov time scaling with the map equation also allows for efficient community detection of network flows in bipartite networks. Most approaches for bipartite networks build on configuration models, in particular modularity [7–9], or stochastic block models [10,11]. An alternative is to project the bipartite network into a unipartite network and perform the analysis on the unipartite network. For most assortative networks, such a projection does not destroy any valuable information [12]. However, the projection can give an overload of links and be infeasible for large networks. Therefore, the analysis of network flows derived from bipartite networks, such as unipartite collaboration networks obtained from projections of author-paper bipartite networks [13], can greatly benefit from evading the projection into overly dense networks. With the map equation for varying Markov times, we can achieve this because a bipartate to unipartite projection corresponds to doubling the Markov times.

We begin by explaining the generalization of the Map equation to different Markov times and then introduce the bipartite generalization.

II. NETWORK FLOW MODULES AT DIFFERENT MARKOV TIMES

The map equation measures how well a partition of nodes in possibly nested and overlapping modules can compress a description of flows on a network. Because compression is dual to finding regularities in the data [14], the modules that give the best compression also are best at capturing the regularities in the network flows. The network flows can be explicit flow data, such as the number of passengers traveling between cities, or be modeled by a random walker guided by the constraints set by a directed, weighted network, such as information flows on a citation network.

In the standard formulation of the map equation, a random walker is modeled as a *discrete-time Markov process* and its position in the network is encoded at every transition. In this way, the transition rate of a random walker as well as the encoding rate is 1. Specifically, the discrete-time transition matrix associated with the network, T_{D}, labeled with subscript D for *discrete*, induces flows between nodes visited with probability $\mathbf{p}$ by the discrete-time Markov process,

$$\mathbf{p}_{k+1} = \mathbf{p}_k T_{\mathrm{D}}. \tag{1}$$

Schaub *et al.* generalized the map equation to different Markov times by using the corresponding *continuous-time Markov process*,

$$\dot{\mathbf{p}} = -\mathbf{p}(I - T_{\mathrm{D}}), \tag{2}$$

with I for the identity matrix [5]. The continuous-time Markov process has exponentially distributed holding times at each node that correspond to Poisson-distributed transitions at average rate 1 [3,15]. With uniform time steps t, the continuous-time Markov process is therefore equivalent to the

*martin.rosvall@umu.se

discrete-time process,

$$\mathbf{p}_{k+1} = \mathbf{p}_k T_{\mathrm{C}}(t), \tag{3}$$

with the continuous-time transition matrix,

$$T_{\mathrm{C}}(t) = e^{-t(I-T_{\mathrm{D}})} = \sum_{i=0}^{\infty} \frac{t^i e^{-t}}{i!} T_{\mathrm{D}}^i, \tag{4}$$

labeled with subscript C for *continuous*. By using this transition matrix, Schaub *et al.* showed the effects of shorter and longer Markov times t between encodings [5]. Shorter Markov times than 1 mean that the average transition rate of a random walker is lower than the encoding rate of its position, such that the same node will be encoded multiple times in a row. As a result, the map equation will favor more and smaller modules. Oppositely, longer Markov times mean that the average transition rate is higher than the encoding rate, such that not every node on the trajectory will be encoded, and the map equation will favor fewer and larger modules. When a two-level solution is preferred over hierarchically nested modules of different sizes, changing the Markov time can in this way highlight salient flow modules at specific scales [5].

A. The map equation for varying Markov times

In detail, for a given partition of nodes into modules, the original map equation for a discrete process at Markov time 1 measures the per-step minimum modular description length of flows on the network. For unique decoding of the flow trajectory from one step to another, the modular coding scheme is designed to only require memory of the previously visited module and not the previously visited node. The map equation therefore has one or, for hierarchically nested modules, more *index codebooks* for encoding steps between modules and *modular codebooks* for encoding steps within modules. Minimizing the map equation over all possible network partitions therefore gives the assignments of nodes into modules that best capture modular flows on the network. That is, the map equation can identify modules in which flows stay for a relatively long time.

As input, the map equation takes the ergodic node visit-rates p_α, module exit-rates $q_{i\curvearrowright}$, and module enter-rates $q_{i\curvearrowleft}$ of the flow trajectory for nodes $\alpha = 1 \dots n$ and modules $i = 1 \dots m$. It estimates the average code length of each codebook from the Shannon entropy, which sets the theoretical lower limit according to Shannon's source code theorem [14]. With $p_{i\circlearrowright} = q_{i\curvearrowright} + \sum_{\alpha \in i} p_\alpha$ for the total rate of use of module codebook i, the per-step average code length of events $\mathcal{P}^i$ in module i is

$$H(\mathcal{P}^i) = -\frac{q_{i\curvearrowright}}{p_{i\circlearrowright}} \log \frac{q_{i\curvearrowright}}{p_{i\circlearrowright}} - \sum_{\alpha \in i} \frac{p_\alpha}{p_{i\circlearrowright}} \log \frac{p_\alpha}{p_{i\circlearrowright}}. \tag{5}$$

Similarly, with $q_{\curvearrowleft} = \sum_{i=1}^m q_{i\curvearrowleft}$ for the total rate of use of the index codebook in a two-level description, the per-step average code length of module enter-events $\mathcal{Q}$ is

$$H(\mathcal{Q}) = -\sum_{i=1}^{m} \frac{q_{i\curvearrowleft}}{q_{\curvearrowleft}} \log \frac{q_{i\curvearrowleft}}{q_{\curvearrowleft}}. \tag{6}$$

With modular map M and the rate of use of each codebook taken into account, the map equation takes the form

$$L(\mathsf{M}) = q_{\curvearrowleft} H(\mathcal{Q}) + \sum_{i=1}^{m} p_{i\circlearrowright} H(\mathcal{P}_i). \tag{7}$$

For an efficient generalization of the map equation to Markov times other than 1, we first linearize in t and T_{D} the continuous-time transition matrix $T_{\mathrm{C}}(t)$ in Eq. (4). For $t < 1$, $(1-t)I + tT_{\mathrm{D}}$ is a valid approximation, but we are also interested in Markov times greater than 1. Thus, we consider the linearized transition matrix,

$$\tilde{T}_{\mathrm{C}}(t) = \begin{cases} (1-t)I + tT_{\mathrm{D}} & t < 1 \\ tT_{\mathrm{D}} & t \geqslant 1 \end{cases}, \tag{8}$$

which captures Markov times below 1 with self-links and Markov times above 1 with transition rates proportional to the average rate of the underlying Poisson process. Moreover, at Markov time 1 it recovers the discrete-time transition matrix in Eq. (1).

This linearization also has an appealingly simple effect on the map equation. For Markov time t, all node visit rates p_α remain the same, since the relative visit rates at steady state do not depend on how often the visits are sampled. However, the module exit-rates $q_{i\curvearrowright}$ and module enter-rates $q_{i\curvearrowleft}$ change linearly with the Markov time, since the number of random walkers that moves along any link between nodes during time t is directly proportional to t as shown in Eq. (8). Therefore,

$$q_{i\curvearrowright} \to tq_{i\curvearrowright} \equiv q_{i\curvearrowright}(t), \tag{9}$$

$$q_{i\curvearrowleft} \to tq_{i\curvearrowleft} \equiv q_{i\curvearrowleft}(t). \tag{10}$$

The rescaled module exit- and enter-rates affect both the module code length in Eq. (5) and the rate of use of all codebooks. With (t) for the Markov time, the map equation for Markov time t takes the form

$$L(\mathsf{M},t) = q_{\curvearrowleft}(t) H(\mathcal{Q}) + \sum_{i=1}^{m} p_{i\circlearrowright}(t) H[\mathcal{P}_i(t)]. \tag{11}$$

The simple flow rescaling enables efficient community detection at different Markov times with the search algorithm Infomap [16]. While Infomap is designed to minimize the original map equation over possible network partitions, it can be applied to the reconstructed network that corresponds to the transition matrix for a given Markov time. This works for the continuous-time transition matrix $T_{\mathrm{C}}(t)$ in Eq. (4) [5], as well as for its linearized form in Eq. (8). While reconstructing the linearized transition matrix is much faster and does not densify the network, further improvement is possible. In fact, the reconstruction can be completely evaded. Since the self-links only indirectly affect the map equation for Markov time t in Eq. (11) by reducing the transition rates between nodes and modules, exactly the same effect can be achieved by directly rescaling Infomap's internal representation of flows along links by a factor t. This is the approach we take. Infomap takes as input the original network and the Markov time t, calculates the ergodic node visit and transition rates, and then rescales the transition rates by a factor t without any network reconstruction at all.

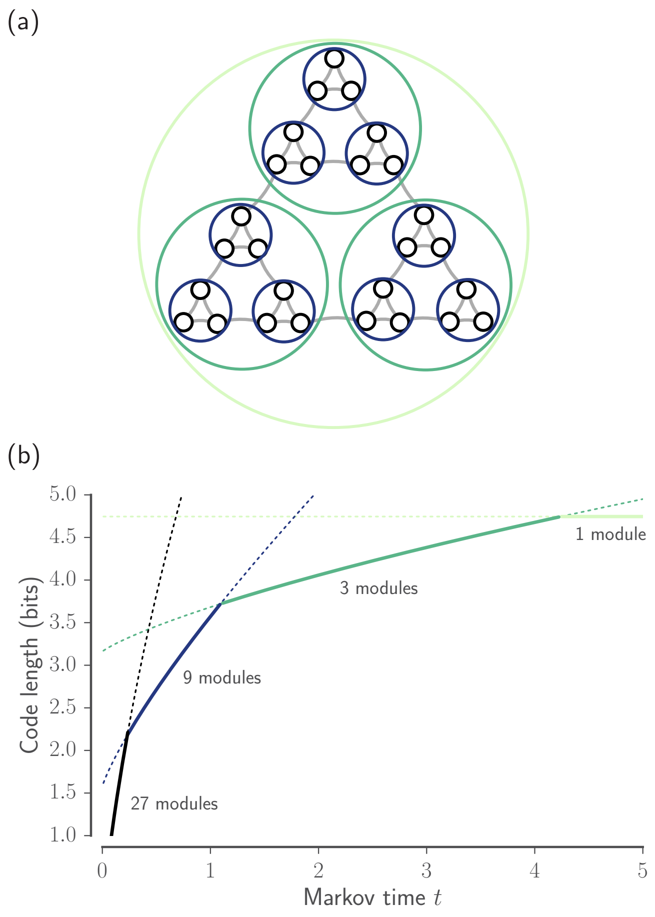

FIG. 1. The Markov time sets the scale of the flow modules. (a) A schematic Sierpinski network with hierarchically nested modules. (b) The code length for different partitions indicated in the network as a function of the Markov time. The partition with the shortest code length for a given Markov time is highlighted.

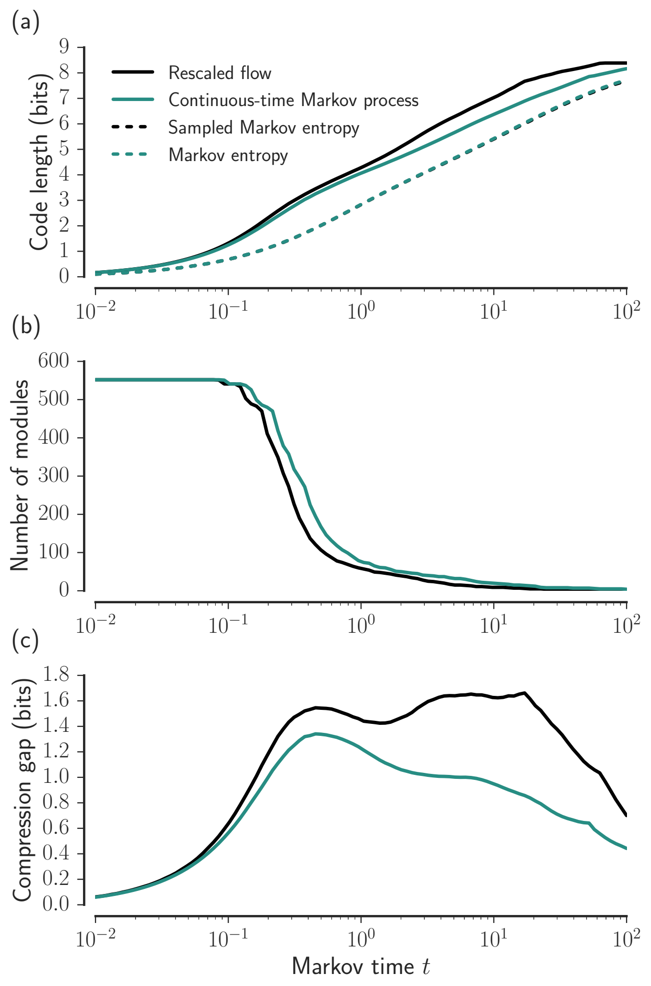

FIG. 2. Comparing flow rescaling with a continuous-time Markov process. Panels (a)–(c) show the effect on a weighted, undirected coauthorship network with 552 physicists [17]. Standard deviations are smaller than the line width.

Figure 1 shows an example with a Sierpinski network. For the shortest Markov times, putting every node in its own module gives the shortest code length. For longer Markov times, solutions with larger and larger modules give the shortest code length.

The simple flow rescaling gives a slightly different encoding of the dynamics than the continuous-time Markov process [5]. The flow rescaling only operates on transitions between nodes directly connected in the original network and does only indirectly consider transitions between nodes connected by multistep trajectories. Contrarily, the continuous-time Markov process directly considers a spectrum of these trajectories. Their lengths are given by the transition matrix power in the expanded continuous-time transition matrix in Eq. (4),

$$T_{\mathrm{C}}(t) = e^{-t}I + te^{-t}T_{\mathrm{D}} + \frac{t^2e^{-t}}{2}T_{\mathrm{D}}^2 + \frac{t^3e^{-t}}{6}T_{\mathrm{D}}^3 + \dots, \quad (12)$$

such that they are Poisson distributed with mean length t. From a coding perspective, the continuous-time transition matrix allows a random walker on a multistep journey on the original network to move out of a module and back again between two encodings without triggering any module exit- and enter-codewords. In the flow rescaling approach, however, such moves will indeed be encoded. As a result, the continuous-time Markov process allows flows to stay longer within a given module and therefore typically gives smaller modules and shorter description length. Figure 2 illustrates the effects of the different dynamics on a weighted, undirected coauthorship network with 552 physicists [17]. While the flow rescaling gives longer code lengths especially for longer Markov times [Fig. 2(a)], and somewhat larger modules for the same Markov time [Fig. 2(b)], the overall patterns are the same.

The network and problem at hand may set a natural Markov time, but often the most appropriate Markov time is unknown. Based on the rationale that good modular solutions should give good compressions, Schaub *et al.* suggested to compare the code length of the modular description by the map equation at a given Markov time with the entropy rate of the corresponding Markov process,

$$h_{\mathrm{C}}(t) = -\sum_{\alpha\beta} p_\alpha T_{\mathrm{C}\alpha\beta}(t)\log T_{\mathrm{C}\alpha\beta}(t), \quad (13)$$

which sets the lower limit on the description length [5]. We use the same compression gap approach, but for better performance instead obtain the entropy rates at different Markov times by sampling random walks on the original network. That is, we repeatedly sample start nodes proportional to their ergodic visit-rates, and, for each start node, repeatedly perform random walks of lengths sampled from a Poisson distribution with expected length t. By averaging over the entropy of the final node for each start node, we can estimate the entropy rate of the continuous-time Markov process without constructing the corresponding continuous-time transition matrix $T_{\mathrm{C}}(t)$. Note that we cannot use the linearized transition matrix in Eq. (8), because the corresponding entropy rate is only a good estimate for $t < 1$ and does not converge to the entropy rate of the independent and identically distributed process for long Markov times, $\lim_{t\to\infty} h_{\mathrm{C}}(t) = -\sum_\alpha p_\alpha \log p_\alpha = H(\mathcal{P})$, which is also the one-module solution of the map equation for any Markov time. Figure 2(a) shows that the sampled estimate performs well and practically overlaps with the Markov entropy obtained from the continuous-time transition matrix. Schaub *et al.* looked at the relative compression gap [5], but to avoid inflating small differences for short Markov times, in Fig. 2(c) we show the absolute compression gap, $L(\mathsf{M},t) - h_{\mathrm{C}}(t)$. For this coauthorship network, the compression gaps indicate a local minimum just shorter than Markov time 2 for the rescaled flow and a local quasiminimum just longer than Markov time 2 for the continuous-time Markov process. Interestingly, these Markov times correspond to about the same number of modules, since the flow rescaling generates slightly larger modules for the same Markov time.

Overall, the flow rescaling is in practice computationally much more efficient than the continuous-time Markov process, since the network must not be rebuilt for each Markov time. The continuous-time Markov process generates dense networks for long Markov times, which results in infeasible solutions for large networks. Contrarily, the flow rescaling has similar fast performance for all Markov times. However, for networks so sparse that random fluctuations can cause quenched modules [18], it can pay off to incorporate longer trajectories. Then extending the linearized transition matrix in Eq. (8) with quadratic terms from the continuous-time transition matrix $T_{\mathrm{C}}(t)$ in Eq. (4) can provide an efficient compromise between the slower continuous-time Markov process, which makes the network denser, and the faster flow rescaling, which maintains the network density.

B. The map equation for bipartite networks

A complete projection of a bipartite network with *primary nodes* and *feature nodes* into a unipartite network with only primary node gives an overload of links already for moderately dense networks [13]. Here we explore three ways to overcome this problem for the map equation framework: projecting by rescaling the Markov time, treating the network as unipartite, and projecting by sampling important links.

Flow rescaling makes a projection effortless, because projecting a bipartite network into a unipartite network essentially corresponds to a rescaling of the Markov time. With Markov time 2, a random walker will take two steps between two encodings such that the exit and enter rates according to Eqs. (9) and (10) become

$$q_{i\curvearrowright}(2) = 2q_{i\curvearrowright}, \tag{14}$$

$$q_{i\curvearrowleft}(2) = 2q_{i\curvearrowleft}. \tag{15}$$

If such random walkers with a cycle of two are released on the primary nodes, only the primary node visits will be encoded. In this way, the map equation takes exactly the same form as in Eq. (11) with $t = 2$,

$$L(\mathsf{M},2) = q_{\curvearrowleft}(2)H[\mathcal{Q}(2)] + \sum_{i=1}^{m} p_{i\circlearrowright}(2)H[\mathcal{P}_i(2)], \tag{16}$$

with the only difference that the visit rates of primary nodes double and the visit rates of feature nodes become 0. Therefore, with subscript p for primary nodes and f for features nodes,

$$p_{\alpha,p}(2) = 2p_\alpha(1), \tag{17}$$

$$p_{\alpha,f}(2) = 0. \tag{18}$$

Even if visits to feature nodes do not contribute to the code length, the flow rates between modules depend on their module assignments. Therefore, both primary nodes and feature nodes are clustered. Since the flow rescaling treats movements in and out of modules differently than with a projected and fully rebuilt network as described above, the projection with rescaled Markov time best approximates the full projection for small flows between modules (see Fig. 3).

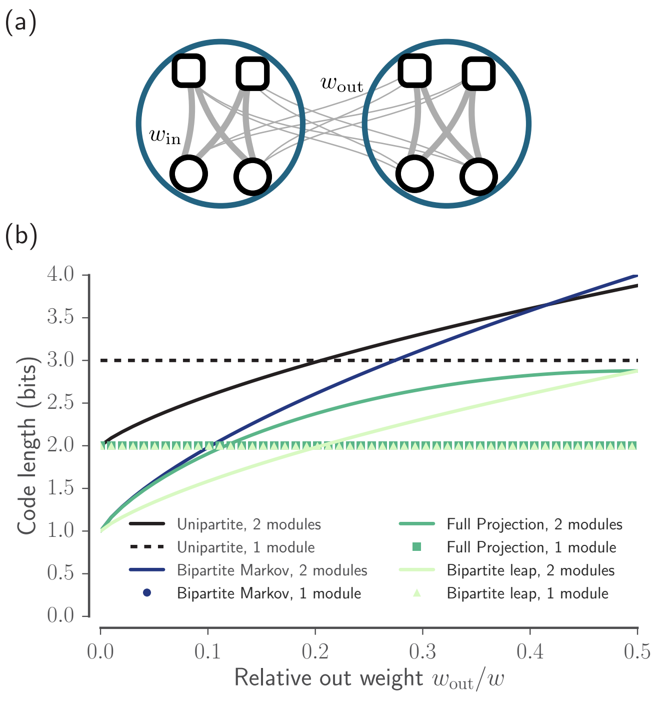

FIG. 3. Projecting bipartite networks corresponds to doubling the Markov time and increases the scale of flow modules. (a) A schematic bipartite network with link weight w_{in} between primary nodes (circles) and feature nodes (squares) in the same community and link weight w_{out} between nodes in different communities. (b) The code length for different bipartite dynamics and coding schemes as a function of the relative out weight.

A similar approach to doubling the Markov time is to instead use random walkers that leap over every other node. That is, the dynamics take place on the full network with primary nodes and feature nodes as above, but only steps from feature nodes to primary nodes are accounted for. By rescaling the total visit rates to 1, the node visit rates take the same form as in Eqs. (17) and (18), but the transition rates in Eqs. (9) and (10) now depend on the relative amount of flow that moves between modules from feature nodes to primary nodes. For undirected networks, the flow is equal in both directions such that the bipartite leap dynamics correspond to Markov time $t = 1$ in Eqs. (9) and (10). That is, the bipartite leap dynamics effectively correspond to the standard unipartite dynamics in which the node type is ignored as shown in Fig. 3. While only encoding primary nodes offsets the code length compared to the unipartite dynamics, the compression gain between different modular solutions remains exactly the same for the schematic network in Fig. 3. In general, the difference is so small that an approach based on the bipartite leap dynamics is superfluous, and we will instead use the unipartite dynamics when comparing different approaches.

The research question at hand will determine which approach should be favored. In the example in Fig. 3, the two approaches that correspond to dynamics with Markov time 2, full projection and projection with rescaled Markov time, favor the two-module solution until about 10% relative out-weight. Therefore, they can work well for sparse networks or interest in large-scale structures. Instead, the two approaches that correspond to Markov time 1, the unipartite and bipartite leap dynamics, favor the two-module solution until about 20% relative out-weight. Therefore, they can work well for dense networks or interest in small-scale structures.

With two methods that can work well at different scales, we now turn to a fast projection approach based on sampling of important links that resembles the method we used for estimating Markov entropies above. It is an adaptive method that can work well at a wider range of scales. Sampling of important links works well in practice, because most links in a weighted projection will carry redundant information for community detection. Therefore, only the important and nonredundant links must be sampled. Much like the Minhash approach [19], we seek to identify similar nodes of one type. In our case, nodes that are frequently visited in sequence by a random walker that performs two-step dynamics on a bipartite network. In detail, we associate each feature node with the top X primary nodes selected by link weight, or randomly for ties as in unweighted networks. For each primary node, we take the top X primary nodes associated with each of its connected feature node and include them in a candidate set. For each node in the candidate set, we compute the two-step random walk probability to go to other nodes also in the candidate set and create links to the top Y nodes. For all experiments in this paper, we used $X = 1000$ and $Y = 10$. For these choices, we found that the sampling approach can be both fast and accurate for dense as well as sparse networks.

III. RESULTS AND DISCUSSION

To compare the three methods, we tested their performance on bipartite benchmark networks. To construct the bipartite benchmark networks, we built on the standard approach with a generative model for unipartite networks [20]. We assigned both *primary nodes* and *feature nodes* to communities and then added k unweighted and undirected links between each primary node and k_{in} randomly chosen feature nodes in the same community and $k_{\text{out}} = k - k_{\text{in}}$ randomly chosen feature nodes in other communities. Specifically, we used 32 communities, each with 32 primary nodes with average degree 16, and varied the number of links between communities and the number of feature nodes for more or less sparse networks.

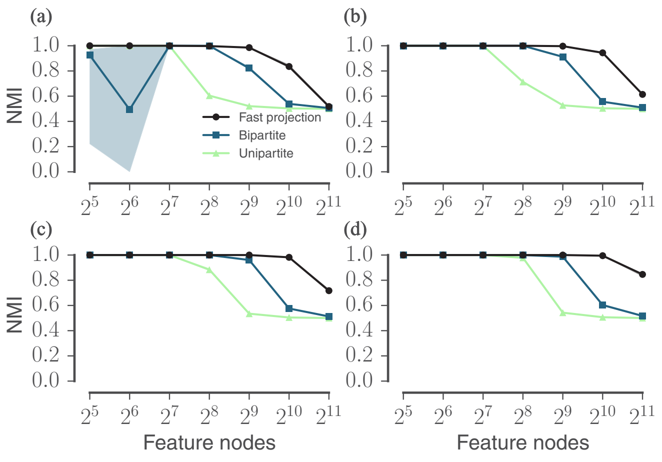

FIG. 4. Fast projection performs well on both sparse and dense bipartite benchmark networks. The performance of fast projection, the bipartite dynamics, and the unipartite dynamics measured by the normalized mutual information, NMI, as a function of the number of feature nodes and the number of links between communities, $k_{\text{in}} = 12$ in (a), 13 in (b), 14 in (c), and 15 in (d). Filled area represents standard deviation.

The bipartite benchmark test reveals the effect of different effective Markov times (Fig. 4). Standard unipartite dynamics or the bipartite leap dynamics, which correspond to Markov time 1, work well down to relatively high number of links between communities as long as the number of feature nodes is limited. With increasing number of feature nodes, the network becomes sparser, and the the dynamics generate quenched modules. The bipartite dynamics, which approximates a projection of the network and corresponds to Markov time 2, cannot resolve communities as accurately as the unipartite approach for dense networks with high number of links between communities (Fig. 4). On the other hand, the bipartite dynamics can better handle sparse networks with many feature nodes. Finally, fast projection effectively adapts the Markov time and handles both dense and sparse networks on par or better than the approaches with fixed Markov times. Unless the research question calls for a specific Markov time, fast projection stands out as a good choice.

Finally, we applied the three different methods on four real-world bipartite networks (see Table I). For each network we report the number of primary and feature nodes and the number of links. We applied both two-level and multilevel community detection with the search algorithm Infomap [16]. In the first approach, we forced Infomap to find two-level solutions, while in the second approach we let Infomap

TABLE I. Comparing two-level and multilevel community detection of unipartite dynamics, bipartite dynamics, and fast projection applied to real-world bipartite networks. Modules for the multilevel solutions report the total number of modules across all levels. All result values are reported with two significant figures.

	arXiv collaboration			20 Newsgroups			YouTube			MovieLens		
Primary nodes			16 726			17 856			94 238			6 040
Feature nodes			22 015			78 198			30 087			3 900
Links			58 595			1 873 331			293 360			1 000 209
	Unipart.	Bipart.	F. proj.	Unipart.	Bipart.	F. proj.	Unipart.	Bipart.	F. proj.	Unipart.	Bipart.	F. proj.
Two-level												
Modules	3100	2200	2500	740	36	660	9500	7900	7100	250	1	35
NMI												
Unipartite	1.00			1.00			1.00			1.00		
Bipartite	0.91	1.00		0.04	1.00		0.59	1.00		0.00	1.00	
Fast projection	0.94	0.92	1.00	0.08	0.00	1.00	0.77	0.57	1.00	0.00	0.00	1.00
Multilevel												
Levels	6	5	6	2	2	4	5	3	4	2	1	2
Modules	7300	3100	4200	740	36	900	12 000	8000	8000	250	1	35
HNMI												
Unipartite	1.00			1.00			1.00			1.00		
Bipartite	0.66	1.00		0.04	1.00		0.25	1.00		0.00	1.00	
Fast projection	0.66	0.58	1.00	0.02	0.00	1.00	0.59	0.23	1.00	0.00	0.00	1.00

find the multilevel solution with the optimal number of nested levels for best compression of the dynamics. We report the standard NMI for the two-level approach [21] and the generalized NMI for the multilevel approach [22]. For the multilevel approach, we also report the number of levels for the best solution as well as the total number of modules across all levels. The real bipartite networks include an author-paper network, arXiv collaboration [23]; a document-word network, 20 Newsgroups [24]; a user-group network, YouTube [25]; and a user-movie network, MovieLens [26]. All networks are popular for performing benchmark experiments.

The comparison between the methods applied on real networks confirms the results from the synthetic benchmark tests: unipartite dynamics reveal more and smaller modules than bipartite dynamics because of the inherently shorter Markov time of unipartite dynamics (Table I). Again, fast projection effectively adapts its Markov time and the network determines whether fast projection most resembles unipartite or bipartite dynamics. For the 20 Newsgroups and MovieLens networks, the NMI scores are low because the solutions of the unipartite and bipartite dynamics basically have one dominating module and many tiny modules. The two-level results carry over to the multilevel solutions, and unipartite dynamics typically give deeper solutions than bipartite dynamics. Overall, fast projection adapts the effective Markov time and can handle both sparser and denser networks.

IV. CONCLUSIONS

We introduced an efficient method to perform community detection of network flows at different Markov times. The method takes advantage of the information-theoretic machinery of the map equation and handles projections of bipartite networks as well. In synthetic and real-world networks, we showed how modifying the Markov times influences the size of the identified communities. Depending on the network and question at hand, a shorter Markov time with smaller communities in deeper multilevel structures or longer Markov time with larger communities in shallower multilevel structures may be more appropriate. For bipartite networks, we also introduced a fast projection approach that effectively adapts the Markov time for robust communities. While current methods require expensive and often infeasible network reconstructions, the introduced methods offer efficient alternatives applicable to large networks.

We have made the code available in the Infomap software package, which also includes efficient community detection for varying Markov times of higher-order Markov processes [16].

ACKNOWLEDGMENTS

M.R. was supported by the Swedish Research Council Grant No. 2012-3729. We are grateful to Carl Bergstrom and Renaud Lambiotte for helpful discussions.

[1] M. Rosvall and C. T. Bergstrom, Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems, PloS one **6**, e18209 (2011).

[2] T. P. Peixoto, Hierarchical Block Structures and High-Resolution Model Selection in Large Networks, Phys. Rev. X **4**, 011047 (2014).

[3] J-C. Delvenne, S. N. Yaliraki, and M. Barahona, Stability of graph communities across time scales, Proc. Natl. Acad. Sci. USA **107**, 12755 (2010).

[4] M. T. Schaub, J.-C. Delvenne, S. N. Yaliraki, M. Barahona *et al.*, Markov dynamics as a zooming lens for multiscale community detection: non clique-like communities

and the field-of-view limit, PloS one **7**, e32210 (2012).

[5] M. T. Schaub, R. Lambiotte, and M. Barahona, Encoding dynamics for multiscale community detection: Markov time sweeping for the map equation, Phys. Rev. E **86**, 026112 (2012).

[6] M. Rosvall and C. T. Bergstrom, Maps of random walks on complex networks reveal community structure, Proc. Natl. Acad. Sci. USA **105**, 1118 (2008).

[7] M. J. Barber, Modularity and community detection in bipartite networks, Phys. Rev. E **76**, 066102 (2007).

[8] R. Guimerà, M. Sales-Pardo, and L. A. Nunes Amaral, Module identification in bipartite and directed networks, Phys. Rev. E **76**, 036102 (2007).

[9] M. Crampes and M. Plantié, A unified community detection, visualization and analysis method, Adv. Complex. Syst. **17**, 1450001 (2014).

[10] T. P. Peixoto, Parsimonious Module Inference in Large Networks, Phys. Rev. Lett. **110**, 148701 (2013).

[11] D. B. Larremore, A. Clauset, and A. Z. Jacobs, Efficiently inferring community structure in bipartite networks, Phys. Rev. E **90**, 012805 (2014).

[12] M. G. Everett and S. P. Borgatti, The dual-projection approach for two-mode networks, Soc. Networks **35**, 204 (2013).

[13] T. Alzahrani, K. J. Horadam, and S. Boztas, Community detection in bipartite networks using random walks, in *Complex Networks V*, Vol. 549, edited by P. Contucci, R. Menezes, A. Omicini, and J. Poncela-Casasnovas (Springer International Publishing, Switzerland, 2014), pp. 157–165.

[14] C. E. Shannon, A mathematical theory of communication, Bell Syst. Tech. J. **27**, 379 (1948).

[15] R. Lambiotte, J.-C. Delvenne, and M. Barahona, Random walks, markov processes and the multiscale modular organization of complex networks, IEEE Trans. Network Sci. Eng. **1**, 76 (2014).

[16] D. Edler and M. Rosvall, The infomap software package (2016), http://www.mapequation.org.

[17] A. V. Esquivel and M. Rosvall, Compression of Flow can Reveal Overlapping-Module Organization in Networks, Phys. Rev. X **1**, 021025 (2011).

[18] A. Lancichinetti and S. Fortunato, Community detection algorithms: A comparative analysis, Phys. Rev. E **80**, 056117 (2009).

[19] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher, Min-wise independent permutations, J. Comput. Syst. Sci. **60**, 630 (2000).

[20] M. Girvan and M. E. J. Newman, Community structure in social and biological networks, Proc. Natl. Acad. Sci. USA **99**, 7821 (2002).

[21] L. Danon, A. Díaz-Guilera, J. Duch, and A. Arenas, Comparing community structure identification, J. Stat. Mech. Theor. Exp. (2005) P09008.

[22] J. I. Perotti, C. J. Tessone, and G. Caldarelli, Hierarchical mutual information for the comparison of hierarchical community structures in complex networks, Phys. Rev. E **92**, 062825 (2015).

[23] M. E. J. Newman, The structure of scientific collaboration networks, Proc. Natl. Acad. Sci. USA **98**, 404 (2001).

[24] J. Rennie, 20 newsgroups data set (2005), http://people.csail.mit.edu/jrennie/20Newsgroups/.

[25] A. Mislove, Youtube network dataset—konect (2015), http://konect.uni-koblenz.de/networks/youtube-groupmemberships.

[26] J. L. Herlocker, J. A. Konstan, Al Borchers, and J. Riedl, An algorithmic framework for performing collaborative filtering, in *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (ACM, New York, 1999), pp. 230–237.