General optimization technique for high-quality community detection in complex networks

Stanislav Sobolevsky[*] and Riccardo Campari
SENSEable City Laboratory, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, Massachusetts 02139, USA

Alexander Belyi
Belarusian State University, 4 Nezavisimosti Avenue, Minsk, Belarus and SENSEable City Laboratory, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, Massachusetts 02139, USA

Carlo Ratti
SENSEable City Laboratory, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, Massachusetts 02139, USA

(Received 4 November 2013; revised manuscript received 13 May 2014; published 23 July 2014)

Recent years have witnessed the development of a large body of algorithms for community detection in complex networks. Most of them are based upon the optimization of objective functions, among which modularity is the most common, though a number of alternatives have been suggested in the scientific literature. We present here an effective general search strategy for the optimization of various objective functions for community detection purposes. When applied to modularity, on both real-world and synthetic networks, our search strategy substantially outperforms the best existing algorithms in terms of final scores of the objective function. In terms of execution time for modularity optimization this approach also outperforms most of the alternatives present in literature with the exception of fastest but usually less efficient greedy algorithms. The networks of up to 30 000 nodes can be analyzed in time spans ranging from minutes to a few hours on average workstations, making our approach readily applicable to tasks not limited by strict time constraints but requiring the quality of partitioning to be as high as possible. Some examples are presented in order to demonstrate how this quality could be affected by even relatively small changes in the modularity score stressing the importance of optimization accuracy.

DOI: 10.1103/PhysRevE.90.012811 PACS number(s): 89.75.Fb, 89.20.Ff, 02.70.−c, 02.10.Ox

I. INTRODUCTION

The increasing availability of large amounts of data has motivated an enormous general interest in the burgeoning field of network science. In particular, the broad penetration of digital technologies in different spheres of human life provides substantial sources of data sets which explore the intricacies of manifold aspects of human activity. The topics they cover range from personal relationships among individuals to professional collaborations, from telephone communication to data exchange, from mobility and transportation to economical transactions and interactions in social media. Analyzing such data sets often leads to the construction of complex networks describing relations among individuals, enterprises, locations, or more abstract entities, such as the buzzwords and hashtags employed in social media; whenever the resulting structures are geographically located, they can then be studied at different scales, including global, countrywide, regional, and local levels. Furthermore, complex networks can arise from the study of biological phenomena, including neural, metabolic, and genetic interactions.

Community detection is one of the pivotal tools for understanding the underlying structure of complex networks and extracting useful information from them; it has been used in fields as diverse as biology [1], economics (the World Trade Net is analyzed in Ref. [2]) human mobility [3–7], communications [8,9], and scientific collaborations [10]. Many algorithms were devised in the field of community detection, ranging from straightforward partitioning approaches, such as hierarchical clustering [11] or the Girvan-Newman [12] algorithm, to more sophisticated optimization techniques based on the maximization of various objective functions.

The most widely used objective function for partitioning is modularity [13,14]: it relies on comparing the strength of inter- and intracommunity connections with a null model in which edges are randomly rewired. In order to obtain partitions yielding optimal values for modularity, researchers have suggested a large number of optimization strategies: well-known algorithms include the simple greedy agglomerative optimization by Newman [15] and faster Clauset-Newman-Moore heuristic [16]; Newman's spectral division method [13] and its improvements (which employ an additional Kernighan-Lin-style [17] step) [14]; a similar method by Sun *et al.* [18], in which partitions are iteratively refined by considering all possible moves of single nodes to all existing or new communities; the aggregation technique commonly referred to as the Louvain method, extremely fast even on large-scale networks [19]; simulated annealing [20,21]; extremal optimization [22]; and many others [23]. In the last few years, researchers have shown that modularity suffers from certain drawbacks, including a resolution limit [21,24] which prevents it from recognizing smaller communities (a proposed multiscale workaround which involves modifying the network can be found in Ref. [25]).

At least three of the several alternative objective functions deserve to be mentioned: description code length, block model likelihood measure, and surprise. The description code length of a random walk on a network, upon which the Infomap algorithm [26,27] by Rosvall and Bergstrom is based, is a

[*]stanly@mit.edu

well-known information-theoretical measure, reputed to be among the best available [28]; it appears, however, that code length optimization also suffers from a resolution limit, as discussed in Ref. [29], where a workaround is proposed. The second approach is based on the likelihood measure for the stochastic block model, variations of which were suggested in Refs. [30–35]. Finally, Surprise [36] compares the distribution of intercommunity links to that emerging from a random network with the same distribution of nodes per community. For a detailed, if not up-to-date, review of existing community detection methods, the reader can refer to Ref. [23].

A few more strategies for community detection follow: the replica correlation method introduced in Ref. [37], which is also an information-based measure; two recently proposed algorithms, which infer community structures by using generalized Erdős numbers [38] and by focusing on the statistical significance of communities [39]; a recent approach for modularity optimization (conformational space annealing [40]) which delivers acceptable results very quickly and is scalable to larger networks, as is the modification to the algorithm by Clauset, Newman, and Moore [16] proposed in Ref. [41].

A key point in the evaluation of algorithms for community detection is the choice of meaningful benchmarks. Benchmarks can be roughly divided into two groups. In the first, one compares the final scores achieved by different algorithms for the optimization of the same objective function on a variety of networks. In the second type of benchmark, resulting partitions are checked against imposed or well-known structures in synthetic or real-world networks; this kind of benchmark is fundamental for the evaluation of different partitioning techniques not necessarily based on the optimization of the same objective function. Other methods to obtain independent evaluations of the reliability of communities found, without relying on the known community structure nor objective function scores, focus (among other parameters) on recurrence of communities under random walks [42,43], and their resilience under perturbations of the network edges [44].

In the present work we suggest a universal optimization technique for community detection, which we apply to two of the aforementioned objective functions: modularity and description code length. We also present the results of a two-stage benchmark. First, we compare the performance of our algorithm, in terms of the resulting values for objective functions, with a host of existing optimization strategies, separately for modularity and description code length; we show in this way that we consistently provide the best modularity scores, and results on par with Infomap when optimizing description code length. Next, by employing in each case the best available algorithm, we compare the performances of modularity and description code length as objective functions in reconstructing underlying structures on a large set of synthetic networks, as well as the known structures on a set of real-world networks.

II. THE ALGORITHM

The vast majority of search strategies take one of the following steps to evolve starting partitions: merging two communities, splitting a community into two, or moving nodes between two distinct communities. The suggested algorithm involves all three possibilities. After selecting an initial partition made of a single community, the following steps are iterated as long as any gain in terms of the objective function score can be obtained: (1) for each source community, the best possible redistribution of every source nodes into each destination community (either existing or new) is calculated; this also allows for the possibility that the source community entirely merges with the destination; (2) the best merger, split, or recombination is performed. As the proposed technique combines all three possible types of steps, in the following we will refer to it as Combo.

The fulcrum of the algorithm is the choice of the best recombination of vertices between two communities, as splits and mergers are particular cases of this operation: for each pair of source and (possibly empty) destination communities, we perform a shift of all the vertices fashioned after Kernighan and Lin's algorithm [17]. Specifically, we recombine the two communities starting from several initial configurations, which include (a) the original communities, (b) the case in which the whole source community is moved to the destination, (c) a few intermediate mergers, in which a random subset of the source community is shifted to the destination. For each starting configuration, we iterate a series of Kernighan-Lin shifts until no further improvement is possible; each is performed by (1) initializing a list of available nodes to include all the nodes from the original source community and (2) iterating the following steps until list is empty: (a) find the node i in the list for which switching community entails the largest gain or the minimum loss (if no gains are available) and (b) switch i to the other community, remove i from the list of available nodes, and save the intermediate result. After a series of Kernighan-Lin improvements has been completed for each of the starting configurations, we select the intermediate result which yields the best score in terms of objective function. See Algorithm 1 for schematic pseudocode of Combo [45].

It is also worth mentioning that the inclusion of random initial configurations is usually essential to the algorithm performance. The experiments reported in the Supplemental Material [46] in Fig. S5 show that on average considering random configurations increases the resulting modularity score by 2%, which could sometimes correspond to quite a considerable partitioning improvement. As we can see in Table I even much smaller changes to modularity score result in significant variations in the partitioning. Also Fig. S6 from the Supplemental Material [46] shows that despite this randomness results of Combo are very stable (varying in bounds of 0.1%). However, processing random configurations also takes time: without them the algorithm appears to be on overage 4.2 times faster, which makes it possible to suggest this simplified version of the algorithm for the applications when execution time is more crucial. At the same time, replacing such random configurations with partitioning produced via other methods, e.g., spectral division, makes the algorithm more prone to being captured by local maxima.

Experimental tests show a striking regularity in the dependence of Combo execution time on the number of nodes of the network; Fig. 1 demonstrates that this behavior is close to a power law with exponent 1.8. As one can see from the figure

Algorithm 1: Combo.

```
input  : A network $net$ containing $n$ nodes, initial partition $initial\_communities$ (by default initially
         all nodes in one community), the maximal number of communities $max\_communities$
         ($infinity$ by default)
output: A partition of the network into $communities$
1  Initialize variables for storing partitions and their gains;
2  for each pair ($origin, dest$) of communities do // dest may be empty community
       // Calculate best gain from moving nodes from origin to dest
3      ReCalculateGain(origin, dest);
4  while BestGain() $> THRESHOLD$ do
5      PerformMove(best_origin, best_dest, best_partition);
       // Update gains for changed communities
6      for each community $i$ do
7          ReCalculateGain(best_origin, i); ReCalculateGain(i, best_origin);
8          ReCalculateGain(best_dest, i); ReCalculateGain(i, best_dest);
9  Procedure PerformMove(origin, dest, partition)
10     Move nodes from origin to dest according to partition;
11 Procedure BestGain()
12     Select from remembered partitions one with the best gain;
13     Return this gain and corresponding best_origin, best_dest and best_partition;
14 Procedure ReCalculateGain(origin, dest)
15     if dest is new community and we already have max_communities then
16         return;
17     Define and initialize number_of_tries;
18     for $tryI \leftarrow 1$ to number_of_tries do
19         foreach vertex $v$ from origin community do
20             move $v$ to dest or leave in origin with equal probability;
21         Calculate new gain, assign zero to previous gain;
22         while new gain $>$ previous gain do
23             PerformKernighanLinShifts(origin, dest);
24         if achieved gain is greater then current maximum then
25             Remember current partition and gain;
26 Procedure PerformKernighanLinShifts(origin, dest)
27     Calculate gains from moving each node to opposite community;
28     for $i \leftarrow 1$ to size of origin community do
29         Perform temporary movement that produces maximal gain;
30         Remember current gain and moved node;
31         Recalculate all gains;
32     Retrieve the movements leading to a maximal gain among intermediately calculated and perform
       them;
```

Combo can deal with networks of up to 30 000 nodes in time of up to a few hours (on an iMac machine with Core i7 3.1 GHz CPU and 16 GB memory). However, memory availability is a bottleneck for the current implementation, and for the bigger networks the code slows down even more whenever it starts using the computer's virtual memory.

As the sequence of operations in Combo is strongly dependent on the specific network, sharp evaluations of its computational complexity are difficult to obtain; the regularity of the dependence observed in Fig. 1, however, hints at some robust mechanism acting under the hood. In the Supplemental Material [46], we justify an upper bound to the execution time of $O[N^2 \log(\mathcal{C})]$, where N is the number of nodes, and $\mathcal{C}$ the number of communities in the network.

III. MODULARITY OPTIMIZATION BENCHMARKS

We first evaluated the performance of Combo for modularity optimization. We selected six algorithms for the comparison: (a) the Louvain method [19]; (b) Le Martelot [43]; (c) Newman's greedy algorithm (NGA) [15]; (d) Newman's spectral algorithm with refinement [14]; (e) simulated annealing [20]; and (f) extremal optimization [22]. The set of algorithms we have chosen offers a good sample of the current state of the art. Simulated annealing is reputed to be capable of getting very close to real maxima, and extremal optimization offers a good tradeoff between speed and performance [23,47,48]; they provided the best-performing algorithms in at least one benchmark [49]. The recursive Louvain method is fast and relatively effective [28] and

TABLE I. Difference in the modularity score and corresponding NMI similarity between best and alternative partitioning produced by different algorithm.

	Modularity score			
Network	Best	Alternative	Deviation	NMI
1	0.419790	0.418803	0.000987	0.923345
2	0.526799	0.518828	0.007971	0.732029
3	0.566688	0.565416	0.001272	0.924726
4	0.527237	0.498632	0.028605	0.784013
5	0.310580	0.290605	0.019975	0.553769
6	0.605445	0.602082	0.003363	0.919872
7	0.507642	0.493481	0.014161	0.741351
8	0.432456	0.432057	0.000399	0.651622
9	0.955014	0.954893	0.000121	0.971705
10	0.850947	0.846159	0.004788	0.816490

has therefore been applied in various real-world network analyses [50,51]. Newman's greedy algorithm and spectral algorithms can be considered classical approaches, since they were suggested right after modularity was introduced about 10 years ago and were therefore used in a number of previous benchmarks [19,23,28,48]. The technique by Le Martelot is a more recent approach, for which a benchmark already exists [52].

We ran each algorithm on three sets of networks: (1) widely available data sets found in the literature; (2) five graphs, obtained from NDA-protected telecom data, in which the weight of each edge corresponds to the total duration of telephone calls between two locations; and (3) 10 synthetic networks generated using the Lancichinetti-Fortunato-Radicchi approach [53,54]. Detailed descriptions and references can be found in the Supplemental Material [46].

As a measure of the comparative quality of partitioning, we computed the average rank of each algorithm over all the networks on which it has been tested. When multiple algorithms yielded the same modularity, we equated their rank to the best among them (1 for the highest modularity score). For ranks based on execution time we scored zero all those algorithms that didn't converge within 12 hours.

As summarized in Figs. 2 and 3, Combo significantly outperforms other algorithms, with an average rank score of 0.98; the next best placements are simulated annealing (0.67), Louvain (0.55), and spectral method (0.51); other algorithms show considerably less consistent outcomes. Figure 4 shows that Combo is not as fast as the greedy aggregation algorithms (Louvain, Le Martelot), but faster than other algorithms, both complex, such as simulated annealing, and simple, such as NGA (for which we are, however, using a MATLAB implementation). In the worst cases (usually when the resulting number of communities is big enough), Combo finalizes computation in a matter of hours for networks of thousands to tens of thousands of nodes. That is why in cases where the network is big enough and the computational time is crucial, while the resulting partitioning quality is not, using faster approaches might be the better choice.

Often, however, the reliability of the final community structure is of paramount importance: in such cases, we will want to aim at the highest possible value of the objective function, as even small differences in the resulting modularity score can translate into macroscopic variations in the quality

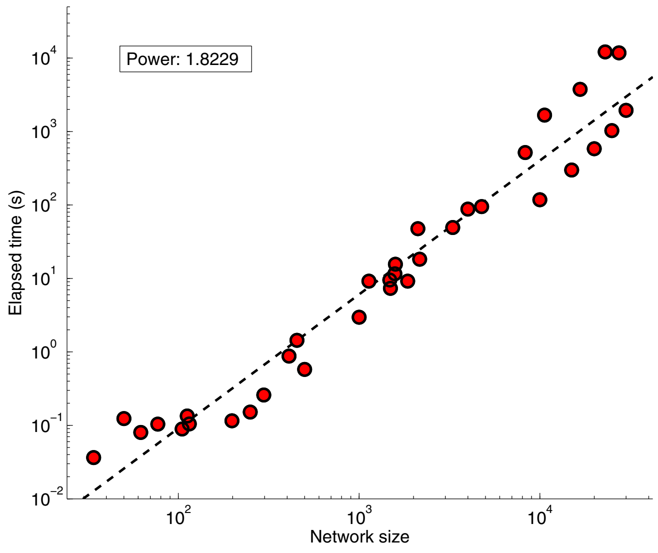

FIG. 1. (Color online) Dependence of Combo execution time on the network size (for all the benchmark networks described below) showing a power law relation.

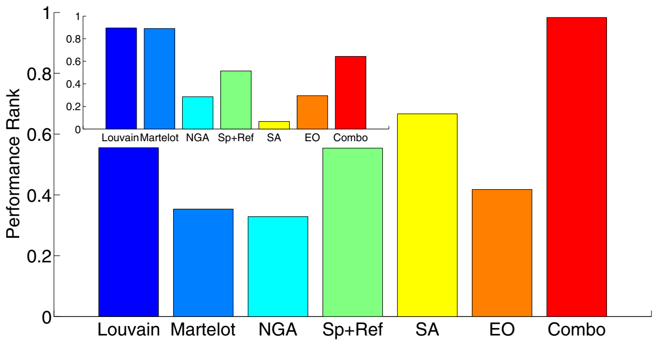

FIG. 2. (Color online) Average normalized performance rank of each algorithm in terms of partitioning quality (main plot) and speed (subplot): values ranging from 0 (worst performance) to 1 (best) are attributed to each algorithm, and their average computed.

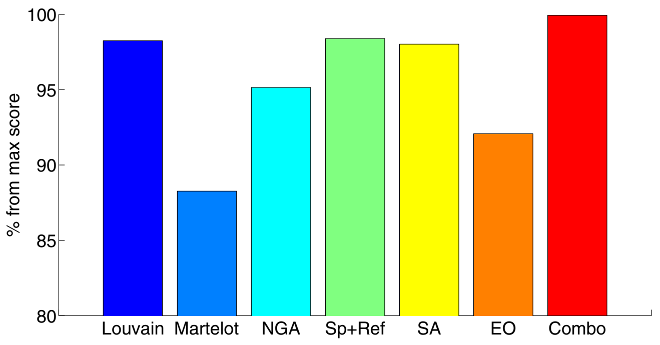

FIG. 3. (Color online) Performance of algorithms as average percentage of their resulting modularity score to the maximum, achieved by the best algorithm.

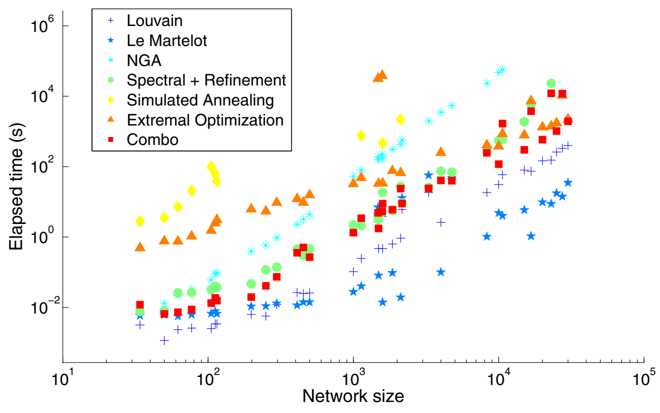

FIG. 4. (Color online) Execution times by network size and algorithm.

TABLE II. NMI similarity to the original network structure and corresponding modularity scores for partitioning of LFR synthetic networks produced by different algorithms.

Network	Modularity score			NMI	
size	Best	Alternative	Deviation	Best	Alternative
1000	0.376667	0.342281	0.034386	0.989395	0.705298
2000	0.339416	0.243512	0.095904	0.998217	0.158417
3000	0.569376	0.556105	0.013271	1.000000	0.969851
4000	0.570596	0.563286	0.007310	0.997617	0.975451
5000	0.616145	0.609881	0.006264	0.996095	0.976030
6000	0.571150	0.556786	0.014364	0.996035	0.954530
7000	0.565559	0.549285	0.016274	0.996824	0.944399
8000	0.614574	0.608583	0.005991	0.991510	0.968284
9000	0.575881	0.566198	0.009683	1.000000	0.961747
10 000	0.605243	0.581807	0.023436	0.996522	0.943804

of partitioning. In the next section, we show that a variation as small as 0.5% can have a sizable impact on the community structure of a network, and Fig. 3 demonstrates that Combo outperforms its nearest rivals by around 2% on average in terms of achieved modularity score. While at the moment it is impossible to guarantee that an achieved partition is a global maximum, we can assume that choosing the one sporting the highest score is the best option.

IV. IMPORTANCE OF PRECISION: THE EFFECT OF SMALL CHANGES IN MODULARITY VALUES ON PARTITIONS

Here in order to stress the importance of looking for even the minor gains in the modularity score, we would like to show that relatively small changes in this partition quality function can be reflected by macroscopic variation of the communities involved. To illustrate this point, first, we compared the partition with the highest modularity score of 10 first networks (incidentally for all 10 networks it is the one obtained by using Combo) from our modularity benchmark (their descriptions can be found in Supplemental Material [46]) with the partitioning obtained by Louvain method being one of the closest competitors. As shown in Table I, differences in modularity score that one might consider to be relatively low can correspond to sizable variations of partition. In order to quantify that difference we used normalized mutual information (NMI) [49] (introduced in detail in the Supplemental Material [46]). It is scaled from 0 to 1, and the more similar partitions are the higher NMI they have, for identical partitions NMI equals to 1. We see that quite often difference in the modularity score less than 0.01 or even 0.001 which one might perhaps consider to be the minor deviation at the first glance, could actually result in substantial variations of the corresponding community structure with the corresponding NMI similarity values sometimes as low as 0.6–0.7.

Another important question of course is whether those noticeable changes in community structure sometimes coming along with the small gains in the modularity scores one could achieve by using the higher performance algorithm, actually improve the partitioning quality in a certain sense. This is a complex question lying mostly beyond the scope of the current article as in fact it requires one to understand to which extent the modularity score itself could be trusted as the partitioning quality function. There is an ongoing debate in the literature about advantages and limitations of the modularity optimization approach including the modularity resolution limit [21,24]. Also the question of what to take for a partitioning quality is not always obvious: even if for some of the real-world networks we possess a knowledge of their actual underlying community structure there is no guarantee it would be indeed optimal in any theoretical sense including modularity score optimization. But just as a simple illustration to that question we introduce a second experiment where we used networks generated by Lancichinecchi-Fortunato-Radicchi's method [53,54] having a pretty much straightforward imposed community structure. For each of the networks we compared two partitions obtained by Combo and Louvain method with this original community structure based on which the network was created. Table II shows that while the results of Combo providing the better modularity score appear to be 99–100% similar to the original community structure, the results of the other method that might seem to be just slightly worse in terms of modularity already demonstrate a much less convincing match: usually around 95–97% but sometimes down to 70% or even 15% in terms of NMI. And better modularity score always comes together with the better NMI.

Just to give a visual example of how the partitioning changes corresponding to the minor modularity improvement could look like we show two different partitions for the United Kingdom telephone network studied in Refs. [8,9] in which link weights represent the number of telephone calls between locations: one obtained with Combo, the other with the Louvain method (Fig. 5). Although the modularity gain is only 0.0043, which at a first glance may suggest that the quality of two partitioning is actually comparable, a number of macroscopic differences are visible. Slightly higher modularity score also translates into a lower level of noise in the spatial structure of the resulting communities and a better agreement with the official administrative divisions of Great Britain being quantified by NMI similarity measure: 0.804 against 0.703.

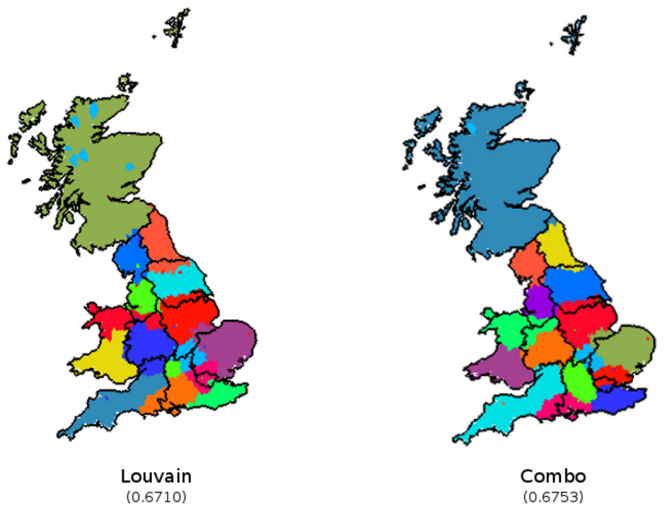

FIG. 5. (Color online) Partitioning of a network based on the number of all calls entertained between each pair of locations. A minimal variation in modularity (less than 1%) can turn into a sizable difference in partitioning. Here the Combo results show cleaner geographical separation of communities and are substantially more similar to official administrative divisions with modularity equal to 0.6753 and NMI = 0.804, compared to modularity = 0.6710 and NMI = 0.703 for Louvain.

V. MINIMUM DESCRIPTION CODE LENGTH BENCHMARKS

In our second benchmark, we use the Combo algorithm to optimize description code length compression and compare the results to those obtained using the original Infomap implementation by Rosvall and Bergstrom [26,27]. Because of longer execution time of Combo for code length, we ran the comparison on the set of networks of size up to 8000 from the previous benchmark. Since Infomap is a greedy algorithm and results are dependent on a random seed, we ran it 10 times for each network and picked the best result.

Unlike for modularity, final values for code length are very close, with a single network in which their difference is about 5%, and less than 3% in all other cases; Combo yields a better code length in 8 networks, Infomap in 9, the results being the same in all other cases. Detailed results are reported in the Supplemental Material [46]. Combo thus results in a valid alternative and an ideal complement to Infomap, as in several cases it is proved capable of finding better solutions.

The analysis above proves that Combo is efficient in terms of optimization of both objective functions: modularity and code length. Now having such a high-performance universal optimization technique opens a new research opportunity worth additional consideration. Some attempts at comparing multiple partitioning algorithms dealing with different objective functions are already present in the literature [55]. However, if done using different optimization techniques it is not possible to clearly judge whether higher performance of a certain approach is due to the objective function relevance or just the optimization technique performance. As Combo efficiently yields near-optimal results for both modularity and code length, we can now for the first time fairly compare modularity and code length as community detection objective functions. As a simple initial criteria for such a comparison we consider the ability of reproducing the existing preimposed community structure in synthetic networks. Results are presented in the Supplemental Material (see the section Modularity vs. Description Code Length Comparison) [46]. Overall we found that modularity yields more reliable community reconstruction in more complex cases as the level of noise increases. Also code length performs surprisingly poorly for smaller networks, while for bigger networks with relatively low level of noise its performance already exceeds the one of modularity. Based on that, one could recommend using modularity for discovering community structure in networks with a weaker clustering effect, while code length might be a better choice for larger networks with relatively strong communities.

VI. CONCLUSIONS

We have presented Combo, an optimization algorithm for community detection capable of handling various objective functions, and we analyzed its performance with the two most popular partitioning quality measures: modularity and description code length. With regard to modularity, Combo consistently outperforms all the other algorithms with which we have compared it, including the current state of the art. For what concerns the code length optimization, Combo provides results on par with those of Infomap, which is the defining algorithm for this objective function.

The current implementation of Combo, however, has limitations in terms of maximal network size it is able to handle within a reasonable time: due to memory constrains its current applicability limit is around 30 000 nodes on modern workstations. Running times are usually longer compared to the fastest greedy algorithms, but often considerably shorter than for other highly efficient optimization techniques: networks whose size is close to the above threshold can be handled within a few hours, while smaller networks of several thousand nodes only require minutes. Combo is thus an optimal choice when the quality of the resulting partition is of paramount importance, while the network is not too big and running time is not strictly constrained.

Combo as an optimization technique is flexible, in that it can be adapted to many other objective functions; possible extensions might be stochastic block model likelihood [56] and surprise [36]. Additional advantages include the possibility of limiting the number of resulting communities (e.g., to obtain the optimal bipartitioning of a network) and the algorithm applicability to further fine-tuning of results previously obtained using other algorithms.

Finally, by studying how well the most efficient optimization techniques for modularity and code length reproduce the known underlying community structure of the networks, we have provided as fair as possible a comparison between the two objective functions.

ACKNOWLEDGMENTS

The authors wish to thank Orange and British Telecom for providing some of the datasets used in this study. We further thank National Science Foundation, the AT&T Foundation, the

MIT SMART program, the Center for Complex Engineering Systems at KACST and MIT, Volkswagen, BBVA, the Coca Cola Company, Ericsson, Expo 2015, Ferrovial, the Regional Municipality of Wood Buffalo, AIT, and all members of the MIT Senseable City Lab Consortium for supporting the research. The authors also want to thank Paolo Santi for helpful discussions, and Kael Greco for his help with graphics.

[1] R. Guimerà and L. A. Nunes Amaral, Nature (London) **433**, 895 (2005).
[2] C. Piccardi and L. Tajoli, Phys. Rev. E **85**, 066119 (2012).
[3] C. Thiemann, F. Theis, D. Grady, R. Brune, and D. Brockmann, PLoS ONE **5**, e15422 (2010).
[4] T. Hossmann, T. Spyropoulos, and F. Legendre, in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* (IEEE, Shanghai, China, 2011), pp. 876–881.
[5] A. Amini, K. Kung, C. Kang, S. Sobolevsky, and C. Ratti, EPJ Data Sci. **3**, 6 (2014).
[6] B. Hawelka, I. Sitko, E. Beinat, S. Sobolevsky, P. Kazakopoulos, and C. Ratti, Cartogr. Geogr. Inform. Sci. **41**, 260 (2014).
[7] C. Kang, S. Sobolevsky, Y. Liu, and C. Ratti, in *Proceedings of the 2nd ACM SIGKDD International Workshop on Urban Computing* (ACM, Chicago, IL, USA, 2013), p. 1.
[8] C. Ratti, S. Sobolevsky, F. Calabrese, C. Andris, J. Reades, M. Martino, R. Claxton, and S. H. Strogatz, PLoS ONE **5**, e14248 (2010).
[9] S. Sobolevsky, M. Szell, R. Campari, T. Couronné, Z. Smoreda, and C. Ratti, PloS ONE **8**, e81707 (2013).
[10] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, Nature (London) **435**, 814 (2005).
[11] T. Hastie, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction: With 200 Full-Color Illustrations* (Springer, New York, 2001).
[12] M. Girvan and M. Newman, Proc. Natl. Acad. Sci. USA **99**, 7821 (2002).
[13] M. E. J. Newman and M. Girvan, Phys. Rev. E **69**, 026113 (2004).
[14] M. Newman, Proc. Natl. Acad. Sci. USA **103**, 8577 (2006).
[15] M. E. J. Newman, Phys. Rev. E **69**, 066133 (2004).
[16] A. Clauset, M. E. J. Newman, and C. Moore, Phys. Rev. E. **70**, 066111 (2004).
[17] B. W. Kernighan and S. Lin, Bell Syst. Tech. J. **49**, 291 (1970).
[18] Y. Sun, B. Danila, K. Josić, and K. E. Bassler, Europhys. Lett. **86**, 28004 (2009).
[19] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, J. Stat. Mech.: Theory Exp. (2008) P10008.
[20] R. Guimerà, M. Sales-Pardo, and L. A. N. Amaral, Phys. Rev. E **70**, 025101(R) (2004).
[21] B. H. Good, Y.-A. de Montjoye, and A. Clauset, Phys. Rev. E **81**, 046106 (2010).
[22] J. Duch and A. Arenas, Phys. Rev. E **72**, 027104 (2005).
[23] S. Fortunato, Phys. Rep. **486**, 75 (2010).
[24] S. Fortunato and M. Barthélémy, Proc. Natl. Acad. Sci. USA **104**, 36 (2007).
[25] A. Arenas, A. Fernández, and S. Gómez, New J. Phys. **10**, 053039 (2008).
[26] M. Rosvall and C. T. Bergstrom, Proc. Natl. Acad. Sci. USA **104**, 7327 (2007).
[27] M. Rosvall and C. Bergstrom, Proc. Natl. Acad. Sci. USA **105**, 1118 (2008).
[28] A. Lancichinetti and S. Fortunato, Phys. Rev. E **80**, 056117 (2009).
[29] L. K. Branting, in *Proceedings of the Second International Conference on Advances in Social network Mining and Analysis*, SNAKDD'08 (Springer-Verlag, Berlin, 2010), pp. 114–130.
[30] B. Karrer and M. E. J. Newman, Phys. Rev. E **83**, 016107 (2011).
[31] B. Ball, B. Karrer, and M. E. J. Newman, Phys. Rev. E **84**, 036103 (2011).
[32] P. J. Bickel and A. Chen, Proc. Natl. Acad. Sci. USA **106**, 21068 (2009).
[33] A. Decelle, F. Krzakala, C. Moore, and L. Zdeborová, Phys. Rev. Lett. **107**, 065701 (2011).
[34] A. Decelle, F. Krzakala, C. Moore, and L. Zdeborová, Phys. Rev. E **84**, 066106 (2011).
[35] X. Yan, C. Shalizi, J. E. Jensen, F. Krzakala, C. Moore, L. Zdeborová, P. Zhang, and Y. Zhu, J. Stat. Mech.: Theory Exp. (2014) P05007.
[36] R. Aldecoa and I. Marìn, PLoS ONE **6**, e24195 (2011).
[37] P. Ronhovde and Z. Nussinov, Phys. Rev. E **80**, 016109 (2009).
[38] G. Morrison and L. Mahadevan, PLoS ONE **7**, e38704 (2012).
[39] A. Lancichinetti, F. Radicchi, J. J. Ramasco, and S. Fortunato, PLoS ONE **6**, e18961 (2011).
[40] J. Lee, S. P. Gross, and J. Lee, Phys. Rev. E **85**, 056702 (2012).
[41] K. Wakita and T. Tsurumi, in *Proceedings of the 16th International Conference on World Wide Web* (ACM, New York, 2007), pp. 1275–1276.
[42] J.-C. Delvenne, S. N. Yaliraki, and M. Barahona, Proc. Natl. Acad. Sci. USA **107**, 12755 (2010).
[43] E. Le Martelot and C. Hankin, in *Proceedings of the 2011 International Conference on Knowledge Discovery and Information Retrieval (KDIR 2011)* (SciTePress, Paris, 2011), pp. 216–225.
[44] A. Mirshahvalad, J. Lindholm, M. Derlén, and M. Rosvall, PLoS ONE **7**, e33721 (2012).
[45] The C++ implementation of the Combo algorithm used in this paper can be downloaded from http://senseable.mit.edu/community_detection/combo.zip
[46] See Supplemental Material at http://link.aps.org/supplemental/10.1103/PhysRevE.90.012811 for extended information about experiments and benchmarks.
[47] J. Liu and T. Liu, Physica A **389**, 2300 (2010).
[48] R. Aldecoa and I. Marín, Sci. Rep. **3**, 1060 (2013).
[49] L. Danon, A. Díaz-Guilera, J. Duch, and A. Arenas, J. Stat. Mech. (2005) P09008.
[50] D. Meunier, R. Lambiotte, A. Fornito, K. D. Ersche, and E. T. Bullmore, Front. Neuroinf. **3**, 37 (2009).
[51] L. Zhang, X. Liu, F. Janssens, L. Liang, and W. Glanzel, J. Informetrics **4**, 185 (2010).
[52] E. L. Martelot and C. Hankin, International Journal of Web Based Communities **9**, 323 (2013).

[53] A. Lancichinetti, S. Fortunato, and F. Radicchi, Phys. Rev. E **78**, 046110 (2008).
[54] A. Lancichinetti and S. Fortunato, Phys. Rev. E. **80**, 016118 (2009).
[55] T. van Laarhoven and E. Marchiori, Phys. Rev. E **87**, 012812 (2013).
[56] M. E. J. Newman, Europhys. Lett. **103**, 28003 (2013).