# Community Detection in Complex Networks exploiting Spectral Graph Sparsification for Efficient Disaster Response

Annalisa Socievole$^{1[0000-0001-5420-9959]}$ and Clara Pizzuti$^{1[0000-0001-7297-7126]}$

National Research Council of Italy (CNR), Institute for High Performance Computing and Networking (ICAR), Via Pietro Bucci, 8-9C, 87036 Rende (CS), Italy
`{annalisa.socievole, clara.pizzuti}@icar.cnr.it`

**Abstract.** One of the most significant challenges faced by disaster preparedness and response is creating efficient strategies to mitigate the impact of catastrophic events. The phases of disaster planning and strategy development usually necessitate extensive cooperation among communities and individuals. As such, both online and offline social network dynamics play a vital role in the formation of effective disaster management plans. In this paper, we focus on efficient community detection in disaster networks. The input network graph, which is usually a dense graph, is initially sparsified through a spectral-based edge sampling technique. This sparsification procedure is applied to remove less important edges while preserving those critical for maintaining a modular structure, aiming at realizing a faster, greener and reliable community detection. Next, we run a genetic algorithm-based community detection method that maximizes modularity as the objective function on the sparsified graph which is a proxy of the initial graph. Experimental results on synthetic networks demonstrate the effectiveness of our approach compared to other benchmark methods.

**Keywords:** Community Detection · Spectral Sparsification · Genetic Algorithms · Graph Sparsification · Disaster Response.

## 1 Introduction

Disaster network science [11] is an interdisciplinary field that combines principles from network science, disaster management, and social science to study, model, and improve the response to and recovery from disasters. It involves the analysis of various types of networks, such as social networks, communication networks, and infrastructure networks, to understand how information, resources, and people move and interact during and after catastrophic events.

Key aspects of disaster network science include:

– Modeling and Simulation: using mathematical and computational models to simulate the behavior of networks during disasters, helping predict the spread of impacts and identify critical points of failure.

- Network Analysis: examining the structure and dynamics of networks to identify key nodes and links that are crucial for effective disaster response and recovery. This can include identifying influential individuals or organizations in social networks or critical infrastructure components in physical networks.
- Resilience and Robustness: assessing and enhancing the resilience of networks to withstand and quickly recover from disasters. This involves studying how networks can be designed or modified to be more robust against disruptions.
- Information Dissemination: understanding how information spreads through networks during disasters, which can help improve communication strategies and ensure that critical information reaches those in need promptly.
- Coordination and Collaboration: analyzing how different entities (such as government agencies, NGOs, and communities) interact within networks to improve coordination and collaboration in disaster response and recovery efforts.
- Resource Allocation: optimizing the distribution of resources (such as food, water, medical supplies, and personnel) through networks to ensure that aid reaches affected areas efficiently.

Disaster network science leverages data from various sources, including social media, satellite imagery, and sensors, to provide insights that can inform better disaster preparedness, response, and recovery strategies. The ultimate goal is to reduce the impact of disasters on society by making systems more adaptive and resilient. Those systems are usually modeled as network graphs through the Complex Network Theory. Within such networks, the nodes are usually connected by numerous edges, most of which are weighted, leading to highly dense graphs [1]. Rapid data mining is crucial in such systems to enable prompt response and action from public organizations and decision-makers. *Edge pruning* is a strategy that can simplify the analysis and management of large graphs by removing the weak connections. This makes the graph-theoretical analysis of dense weighted structures more manageable. Algorithms generally run much faster on sparser graphs, sometimes by orders of magnitude, and the network graph can be stored in a more compact form. This is particularly important in disaster situations, where storage resources are often constrained by limited energy availability.

In the field of *graph sparsification*, various strategies have been proposed [16,17,19,20]. The goal of edge-cutting in graphs is to create a graph $G'$, known as a *sparsifier*, which is a condensed version of the original graph $G$ with the same set of vertices. If $G'$ closely resembles $G$ according to certain metrics, then $G'$ can serve as an approximation of $G$ for computational purposes without introducing significant errors. *Spectral sparsification* [2] is a type of sparsification in which a dense graph is transformed into a sparser graph while preserving its essential spectral (eigenvalue) properties. This technique aims to reduce the number of edges in the graph significantly, resulting in a sparse graph that approximates the original graph's behavior with respect to certain properties, especially those related to graph Laplacians and eigenvalues. A classic example of spectral sparsification is the *Spielman-Srivastava* algorithm [17], which produces a sparse graph that approximates the original graph's Laplacian matrix. The algorithm selects edges with probabilities based on their effective resistances [7, 12] and ensures that the resulting sparsified graph has a similar spectral profile to the original.

In this paper, we exploit the Spielman-Srivastava sparsification in order to improve the *community detection task* in disaster network scenarios. Understanding the community structure has many advantages. First, subdividing a system into groups helps the various agencies and NGOs to collaborate more effectively, ensuring that their efforts are complementary rather than duplicative. Moreover, by identifying communities within a social network, authorities can pinpoint influential groups or leaders within those communities who can disseminate information quickly and effectively. Finally, detecting communities helps in understanding the population distribution and their specific needs, enabling more efficient allocation of resources like food, water, and medical supplies. In those contexts, a preliminary spectral sparsification of the network graph will be able to reduce the complexity of graph-based computations while preserving the modules of the original graph. In other words, we uncover the communities of a network in a faster way since a lower number of edges is taken into account, and at the same time, the communities found are highly or totally similar to the ground truth since the sparsification applied guarantees that the spectral features of the original graph are kept.

In [15], we introduced *OmeGANet*, a community detection method based on *Genetic Algorithms (GAs)* that exploits the effective resistance as metric to weight edges and, subsequently, weight thresholding as sparsification procedure of the weighted graph to eliminate a proper percentage of edges below the threshold. We therefore run the GA over a sparse weighted adjacency matrix by evolving a population of individuals and maximizing *modularity* [8] as objective function. In this paper, we propose a different approach, namely *SRGA* (*Spielman and sRivastava based Genetic Algorithm for community detection*), that exploits spectral clustering based on effective resistance sampling and then applies a genetic algorithm to uncover the communities. Similarly to *OmeGANet*, the effective resistance drives the edge selection but differently from *OmeGANet*, the Spielman-Srivastava sparsification uses a probabilistic approach and edge sampling, a more sophisticated approach which preserves the graph properties reproducing a high-quality sparsifier. On the contrary, the weight thresholding used by *OmeGANet* is an easier and easy-to-implement graph sparsification strategy that removes the edges below a threshold with the disadvantage that if the threshold is not accurately chosen, the sparsified graph can loose important properties. By comparing *SRGA* with *Louvain* [4] on synthetically generated networks, we demonstrate that this sparsification does not alter the community structure thus allowing to process a lower number of edges while producing accurate partitionings.

The paper is organized as follows. The next section describes the most relevant research works in this area. We then recall the Spielman-Srivastava spectral sparsification in Section 3, describe the proposed community detection algorithm in Section 4, the dataset used and the experiments performed to validate *SRGA* in Section 5. The last section concludes the paper and discusses the future work.

## 2 Related Work

In their recent study, Hashemi et al. [10] provide a thorough overview of graph reduction techniques, encompassing *graph sparsification*, *graph coarsening*, and *graph condensation*. Graph sparsification, the focus technique in our current work, was originally introduced by Bencz'ur and Karger [3] to expedite cut algorithms whose efficiency hinges on the quantity of edges.

Yan et al. [20] propose weight thresholding as a straightforward technique for graph sparsification, aiming to reduce the number of edges in a graph by removing all edges with weights below a specified threshold. Their analysis of various synthetic and real-world networks indicates that while some local and global network properties are compromised by edge removal, the community structure remains preserved.

Spielman and Srivastava introduced effective resistance-based graph sparsification in their influential paper [18], presenting an algorithm that operates in nearly-linear time to generate high-quality sparsifiers. This algorithm computes a subgraph of the input graph in $O(m)$ time, with $m$ number of edges, using random sampling, where each edge's inclusion probability is determined by its effective resistance.

The *effective resistance* was proposed by Klein and Randic [12] as a distance measure in graph theory derived from the field of electrical networks. The effective resistance measures the resistance between two nodes in a graph, considering the graph as a network of resistors. Their work provides foundational insights into how effective resistance can be calculated efficiently using tools from linear algebra and spectral graph theory. This metric is pivotal in understanding connectivity, flow dynamics, and other structural properties within complex networks, proving that it characterizes the *commute time* [5], the expected length of a random walk between two end points. This is why the effective resistance is also named *commute time distance.*

Zhang and Bu [21] employ a Gaussian function of effective resistance to weight the input graph for detecting community structure in complex networks. They utilize a bisection spectral method to partition nodes into communities, iterating the procedure until the desired number of communities is achieved. Experiments on real-world networks demonstrate the stability and reliability of their approach. However, the method suffers from high time complexity, making it impractical for large networks, and requires the number of communities to be predetermined.

In a subsequent study, Gancio and Rubido [6] introduce an unsupervised community detection algorithm based on Zhang and Bu's method [21], eliminating the need to specify the number of communities beforehand. They enhance the spectral partitioning with modularity optimization, allowing the algorithm to iterate without predefined community counts or control over outcomes. Evaluation on benchmark networks like Girvan-Newman (GN) and Lancichinetti-Fortunato-Radicchi (LFR) with varying node counts (128 and 1000 nodes) demonstrates the method's high accuracy.

## 3 Spectral sparsification by Effective Resistance Sampling

In general, the key concepts of spectral sparsification [2] can be summarized as follows.

– *Graph Laplacian*: the graph Laplacian is a matrix representation of the graph that is crucial in spectral graph theory. It reflects the structure of the graph and is used to study various properties such as connectivity and flow.
– *Preservation of spectral properties*: the goal is to create a sparsified graph $G'$ from the original graph $G$ such that the Laplacian matrices of $G$ and $G'$ are close to each other. This closeness is typically measured using spectral norms or other metrics related to eigenvalues.
– *Approximation*: the sparsified graph should approximate the original graph in terms of key metrics such as cut values, effective resistances, and random walk behaviors. This means that computations performed on $G'$ should yield results that are close to those obtained from $G$.

Spectral sparsification reduces the computational complexity for algorithms that operate on graphs, such as those for solving linear systems, computing eigenvalues, and performing machine learning tasks. In large-scale network analysis, for example, sparsified graphs allow for faster and more efficient processing while preserving critical structural properties.

The Spielman-Srivastava algorithm [18] [2], that is used in the current work, is a type of spectral sparsification that select the edges according to their effective resistance. This metric can be viewed as a network distance that encapsulates the network's overall structure. For community detection, links between different communities exhibit higher effective resistance compared to links within the same community.

To determine the effective resistance, the undirected and connected graph $G = (V, E)$ is modeled as an electrical circuit, where each edge $(i, j) \in E$ represents a resistor. The weight $w_{ij}$ of each edge signifies its conductance, which is the inverse of the electrical resistance $\omega_{ij}$. Therefore, $\omega_{ij} = \frac{1}{w_{ij}}\ ohm$ [12]. The effective resistance between two nodes can be computed as follows [12]. Let $A$ be the adjacency matrix of the input graph $G$, with $a_{ij} = 1$ if nodes $i$ and $j$ are connected. Let $\Delta = diag(d_i)$ be the $N \times N$ diagonal degree matrix, where $d_i = \sum_{j=1}^{N} a_{ij}$, and $L$ the Laplacian matrix of $G$ defined as the $N \times N$ symmetric matrix $L = \Delta - A$, with elements

$$l_{ij} = \begin{cases} d_i & \text{if } i = j \\ -1 & \text{if the edge } (i, j) \in E \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

The effective resistance $\omega_{ij}$ between nodes $i$ and $j$ is given by

$$\omega_{ij} = l^+_{ii} + l^+_{jj} - 2l^+_{ij} \tag{2}$$

where $l^+_{ij}$ are the elements of the Moore-Penrose *pseudoinverse* matrix $L^+$ of the weighted Laplacian matrix of $G$.

Spielman and Srivastava propose to sample a graph by choosing a random edge of $G$ with a probability $p_{ij}$ proportional to $w_{ij}\omega_{ij}$ (see Theorem 5 in [2]) and weighting the edge with $w_{ij}/qp_{ij}$, where $q$ is the number of samples that the sparsification algorithm takes independently with replacement summing the weights if an edge is chosen multiple times. The chosen edge $(i, j)$ will be added to the edge set of the sparsifier $G'$.

## 4 SRGA

To efficiently compute the communities in a disaster management scenario, we propose a community detection method that runs on a Spielman-Srivastava sparsifier $G'$ of the original graph $G$, namely *SRGA (Spielman and sRivastava based Genetic Algorithm for community detection)*, using the effective resistance as metric to select the edges and drive the community detection. The method runs a GA [9] on $G'$ to solve the following optimization problem: *find a partition $C = \{C_1, ..., C_k\}$ of $G'$ in $k$ communities such that the weighted modularity of $C$ is maximized*. The weighted modularity $Q$ is computed as

$$Q = \frac{1}{2m} \sum_{ij} \left( r_{ij} - \frac{d_i d_j}{2m} \right) \delta(c_i, c_j) \tag{3}$$

where $m$ is the sum of the edge weights of $G'$, $r_{ij} = w_{ij}/qp_{ij}$ is the weight of edge $(i, j)$ in $G'$, $d_i$ and $d_j$ are the weighted degrees of nodes $i$ and $j$ respectively, $c_i$ and $c_j$ are the communities of the corresponding nodes, and $\delta$ is the Kronecker function which yields 1 if $i$ and $j$ are in the same community, that is $c_i = c_j$, zero otherwise. Networks with high modularity exhibit dense connections among nodes within the same community, while connections between nodes in different communities are sparse.

As genetic algorithm, *SRGA* generates a population $P$ of individuals, where each individual $I$ represents a partition into communities. Initially, these partitions are generated randomly. The population then evolves through genetic operators of variation and selection, optimizing the weighted modularity while exploring the search space. Each individual is encoded using the *locus-based* adjacency representation, where $I$ is represented as a vector of $n$ genes, with each gene corresponding to a node and taking a value in the range $\{1,...,N\}$. A value $z$ assigned to a gene $i$ indicates an edge between $i$ and $z$. At the end, a decoding step identifies all connected components of the graph, which correspond to the detected communities.

The *uniform crossover* is employed as crossover operator, creating a random binary mask with a length equal to the number of nodes. The offspring is produced by selecting genes from the first parent where the mask is 0, and from the second parent where the mask is 1. For mutation, *SRGA* randomly assigns the value of the $i$-th gene to one of its neighbors.

The steps of the algorithm are as follows. *SRGA* receives in input the graph $G = (V, E)$, the maximum number of generations $maxGen$, the population Size $popSize$, the crossover fraction $cf$, the mutation rate $mr$, the number of samples $q$ and:

1. initializes a population of random individuals by assigning to each node one of its neighbors;
2. computes the Laplacian $L$ of $G$;
3. computes the Moore-Penrose *pseudoinverse* matrix $L^+$ of $L$;
4. computes the effective resistances from $L^+$ as $\omega_{ij} = l^+_{ii} + l^+_{jj} - 2l^+_{ij}$;
5. computes the sampling probability $p_{ij}$ for each edge $(i, j)$;
6. takes $q$ edge samples from $G$ independently with replacement, choosing a random edge of $G$ with probability $p_{ij}$, summing the edge weights if an edge is chosen multiple times, and includes these samples in $G'$;
7. runs the Genetic Algorithm on $G'$ for a number of iterations by using modularity as fitness function to maximize, uniform crossover and neighbor mutation as variation operators;
8. obtains the partition $C = \{C_1, \dots, C_k\}$ corresponding to the solution with the highest fitness value.

## 5 Experimental Evaluation

We conducted multiple simulations using Matlab 2020a and the Global Optimization Toolbox to evaluate the effectiveness of our proposal. In the subsections that follow, we describe the datasets, the algorithm used for comparison together with the quality index exploited for evaluating the partitions found, and the results of the experiments.

### 5.1 LFR-networks

We generated networks that mimic the complexity and heterogeneity of real-world networks by exploiting the Lancichinetti-Fortunato-Radicchi (LFR) benchmark [13], a robust and realistic test bed for community detection algorithms. The benchmark can generate networks with a power-law degree distribution, similar to those observed in real-world networks. This means some nodes have many connections while most have relatively few. The adjustable parameters used for creating the network graphs are shown in Table 1. In particular, for controlling the the ratio of intra-community edges to inter-community edges, the *mixing parameter* $\mu$ is used. A low $\mu$ value means that most edges are within communities, while a high $\mu$ means many edges connect different communities.

**Table 1.** LFR networks parameter settings.

| **Parameter** | **LFR-128** | **LFR-1000** |
|---|---|---|
| Number of nodes | 128 | 1000 |
| Node average degree | 8 | 10 |
| Node maximal degree | 9 | 60 |
| Exponent for degree distribution | 2 | 2 |
| Exponent for community size distribution | 1 | 2 |
| Mixing parameter $\mu$ | 0.1 | 0.1 |
| Maximal community size | 40 | 600 |
| Minimal community size | 20 | 100 |

## 5.2 Algorithm in comparison and quality indexes

We assess the quality of the solutions using three metrics: (1) Normalized Mutual Information (NMI), (2) modularity, and (3) the number of communities in the resulting partitions. *SRGA* is compared to the benchmark *Louvain* [4], called here Louvain_S, since we apply the community detection algorithm to the graph sparsified with the Spielman-Srivastava algorithm. The Louvain community detection algorithm is a method used to identify community structures within large networks by maximizing modularity. Initially, each node is assigned to its own community. The algorithm then iteratively examines each node and moves it to the community of a neighboring node if such a move results in an increase in modularity. This process continues until no further modularity improvements can be achieved by moving any single node. Once the optimal modularity is reached for the current configuration, a new network is created where each community is represented as a single node, and the process repeats on this aggregated network. These steps are iterated until modularity can no longer be improved, effectively revealing a hierarchical community structure. The Louvain method is known for its efficiency and ability to handle very large networks.

## 5.3 Results

Figure 1 shows a 128-nodes network with 503 initial links and divided into 5 communities. Note that the nodes are colored according through their ground truth. In Figure 2, we report the effect of graph sparsification through the Spielman-Srivastava algorithm. Note that if the number of edge samples placed in $G'$ is too low, for example $q = 100$, the network is oversparsified loosing the community structure. As $q$ increases, the algorithm keeps a larger number of edges of $G$ and place them in $G'$: in other words, a lower number of edges are cut from $G$. To maintain a community structure, we can observe that $q = 300$, for example, is still not enough since the blue and the red communities interleave. This holds also for $q = 400$, while for values of $q$ greater or equal to 500 the communities of the sparsified graph are well distinct and more similar to the ground truth.

Table 2 shows the results of the community detection for *SRGA* and Louvain_S. We report the number of samples considered (column 1), the corresponding number of edge cuts made to $G$ (column 2), the modularity of the

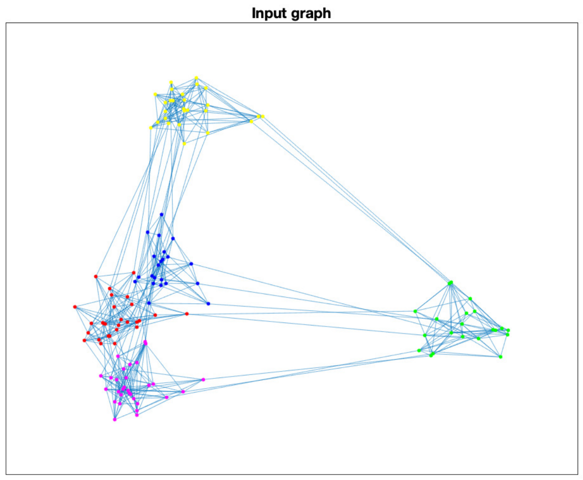

**Fig. 1.** A 128-nodes network generated through the LFR benchmark with mixing parameter $\mu = 0.1$. The network nodes are grouped into 5 communities.

sparsified network (column 3), the performance of Louvain applied to the sparsified graph (column 4), and the performance of SRGA (column 5). For SRGA, the applied genetic parameters are population size = 500, number of generations = 100, crossover fraction = 0.9, mutation rate = 0.1. When no sparsification is used, SRGA obtains modularity 0.703, NMI 1 and 5 communities. As such, when starting to cut edges it would be desirable to obtain the aforementioned results or at least, values of the quality indexes very near to those obtained without sparsification. For $q = 600$ and 150 edge cuts, the sparsified graph achieves the maximum modularity of 0.721, and in this case $SRGA$ is able to perfectly match the ground truth while Louvain_S only achieves NMI 0.958. The interesting result is that $SRGA$ is able to obtain a very good community detection performance from few edge cuts, as expected, until 150 cuts and even around 200 edge cuts with a very high NMI. On the contrary, the sparsification on Louvain_S has unexpectedly a poor community detection performance even for a low number edge cuts. As an example, for 34 edge cuts, Louvain_S has NMI 0.897, a modularity of 0.648 and finds 4 communities. We hence observe that the GA-based algorithm performs better.

As a second case study, we considered a pool of LFR 1000-nodes networks. For these experiments, the genetic parameters considered are again population size = 500, number of generations = 100, crossover fraction= 0.9, mutation rate = 0.1. In Figure 3, one instance of these networks is plotted. This network has 9295 initial links and is characterized by a partitioning divided into 2 communities, the green and the red one. A sparsifier $G'$ for this network is plotted in Figure 4,

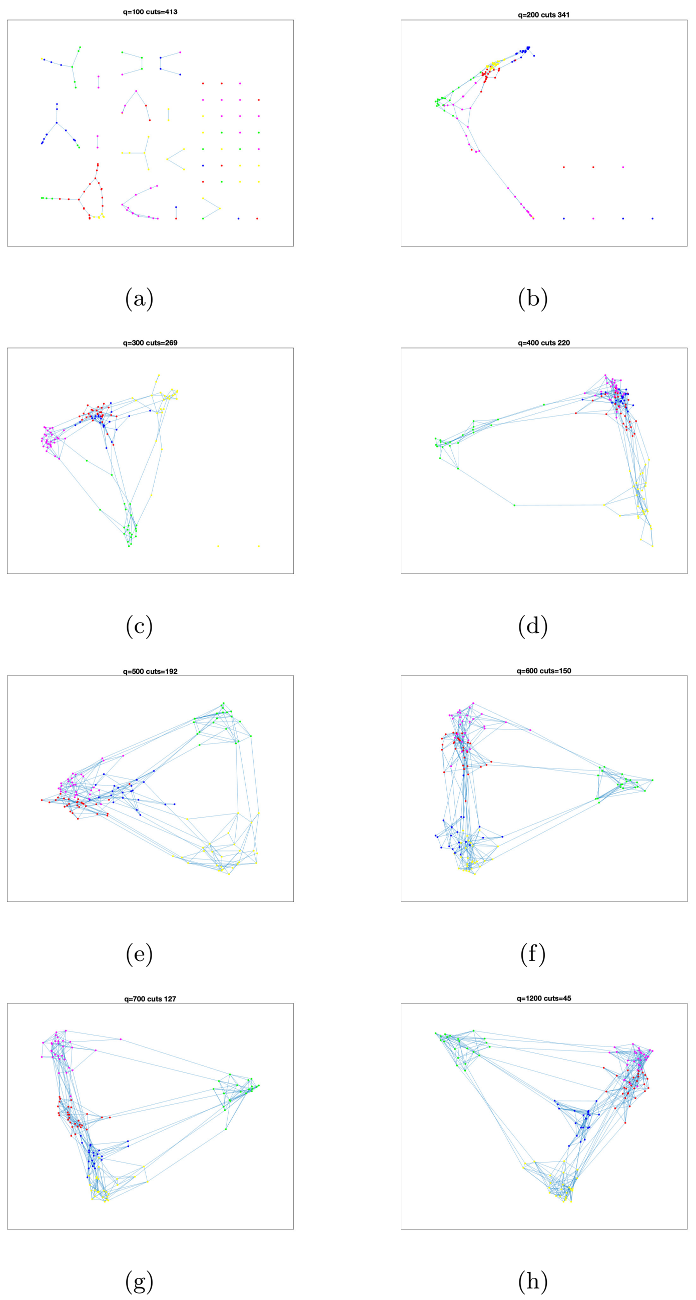

**Fig. 2.** The effect of sparsification on a LFR-128 network as $q$ increases.

**Table 2.** The effect of sparsification on the community detection performance on a 128 nodes LFR network as $q$ increases. Columns 4 and 5 show the results of the community detection algorithms in terms of [Modularity, NMI, # of communities].

| q | # of edge cuts | Modularity | Louvain_S | SRGA |
|---|---|---|---|---|
| 400 | 220 | 0.699 | [0.664, 0.806, 7] | [0.618, 0.958, 5] |
| 500 | 192 | 0.677 | [0.606, 0.681, 7] | [0.69, 0.979, 5] |
| **600** | 150 | **0.721** | [0.712, 0.958, 5] | [0.703, **1**, 5] |
| 700 | 127 | 0.717 | [0.707, 0.978, 5] | [0.703, 1, 5] |
| 800 | 104 | 0.696 | [0.687, 0.938, 5] | [0.703, 1, 5] |
| 900 | 83 | 0.697 | [0.617, 0.831, 4] | [0.703, 1, 5] |
| 1000 | 68 | 0.718 | [0.709, 0.965, 5] | [0.703, 1, 5] |
| 1100 | 66 | 0.698 | [0.676, 0.902, 5] | [0.703, 1, 5] |
| 1200 | 45 | 0.707 | [0.646, 0.912, 4] | [0.703, 1, 5] |
| 1300 | 44 | 0.698 | [0.648, 0.897, 4] | [0.703, 1, 5] |
| 1400 | 34 | 0.699 | [0.648, 0.897, 4] | [0.703, 1, 5] |

in this case 2285 edge cuts where made, corresponding to the setting $q = 15000$. When SRGA is executed on this network a resulting NMI of 0.4405 and 20 communities are found (Figure 5). This segmentation of the communities can be easily fixed by applying the merging post-procedure we proposed in our previous work [14] achieving an NMI of 0.653 and 2 communities.

The results of the experiments on a pool of LFR-1000 networks are shown in Table 3. We generated 100 instances of LFR networks with 1000 nodes and reported the average modularity, the average NMI and in terms of communities, the number of communities that is more frequently uncovered. We reported both the results of the classic *SRGA* and its version with the merging post-processing, namely SRGA_merging. We found that on larger networks our scheme with the merging is able to outperform Louvain_S, achieving the best NMI with $q = 12500$ (i.e., a cut of the 30% of the edges on average).

**Table 3.** The effect of sparsification on the 1000 nodes LFR network as $q$ increases. The modularity of the original network is 0.4044.

| q | Modularity | Louvain_S | SRGA | SRGA_merging |
|---|---|---|---|---|
| 7500 | 0.406 | [0.3793 0.5381, 3] | [0.233, 0.5075,39] | [0.3313 0.5988, 2] |
| 10000 | 0.408 | [0.3747 0.5136, 3] | [0.268, 0.364, 29] | [0.3032 0.5416, 2] |
| 12500 | 0.407 | [0.3654 0.6769, 2] | [0.3054, 0.5064, 17] | [0.2931 0.6896, 2] |
| 15000 | 0.4042 | [0.342, 0.559, 2] | [0.299, 0.44, 20] | [0.3277, 0.6535, 2] |
| 20000 | 0.4038 | [0.342, 0.559, 2] | [0.289, 0.4516, 18] | [0.3082, 0.6128, 2] |

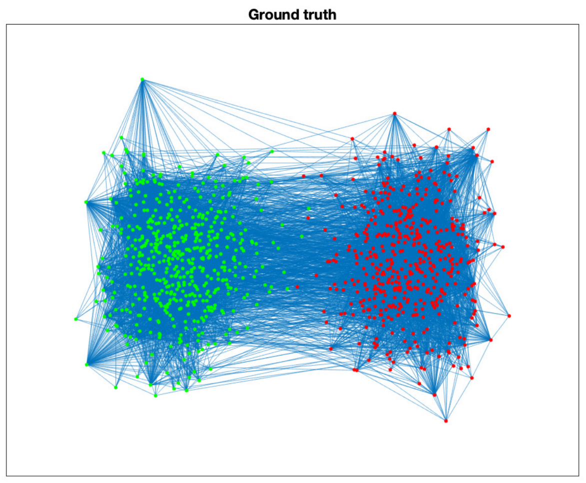

**Fig. 3.** A 1000-nodes network generated through the LFR benchmark with mixing parameter $\mu = 0.1$. The network has 9295 links and nodes are grouped into 2 communities.

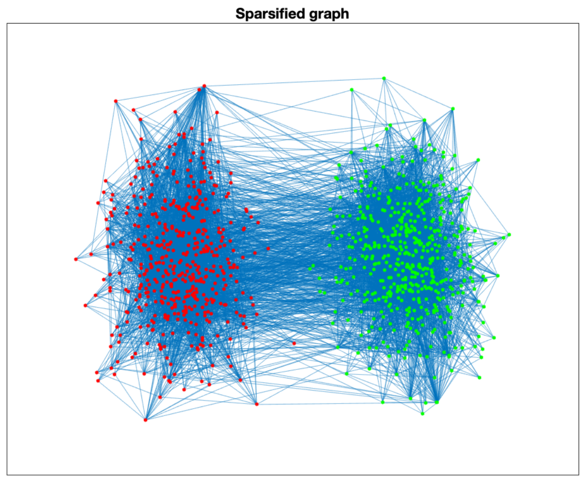

**Fig. 4.** The 1000-nodes network when 2285 cuts are made through effective resistance based sparsification.

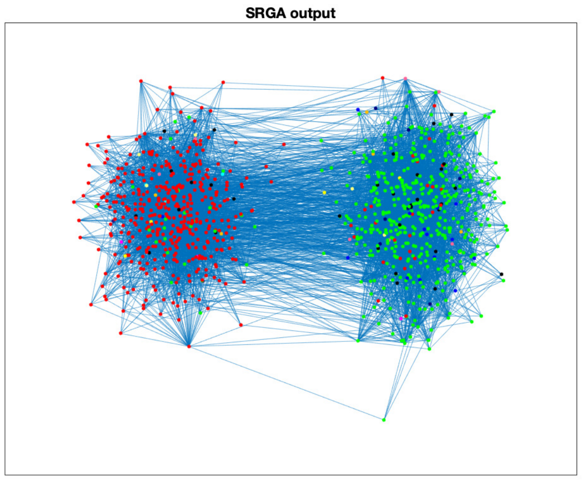

**Fig. 5.** *SRGA* run on the 1000-nodes network: 20 communities are found with NMI 0.4405 and modularity 0.299.

## 6 Conclusions

In this paper, we have presented a community detection method exploiting the Spielman-Srivastava spectral sparsification method to efficiently uncover communities in disasters response. This sparsification procedure eliminates less important edges, in terms of effective resistance distance, while preserving those essential for maintaining the modular structure, aiming at a faster, more energy-efficient, and reliable community detection. By carrying out simulations on synthetic networks that mimic the complexity of the real-world community networks and comparing our GA-based algorithm *SRGA* to Louvain, we have demonstrated how this strategy allows to cut a significant number of edges while uncovering communities with a high NMI.

We point out that this is a preliminary study and the selection of the fraction of edges to remove requires further investigation. More specifically, the resolution of the sparsifier in terms of error with the original graph and how to find an optimal value for $q$ is the research direction of our future work.

**Acknowledgments.** We acknowledge the support of the PNRR project FAIR - Future AI Research (PE00000013), Spoke 9 - Green-aware AI, under the NRRP MUR program funded by the NextGenerationEU.

**Disclosure of Interests.** The authors have no competing interests to declare that are relevant to the content of this article.

## References

1. Barrat, A., Barthelemy, M., Pastor-Satorras, R., , Vespignan, A.: The architecture of complex weighted networks. In: Proc. National Academy of Science. pp. 101,3747 (2004)
2. Batson, J., Spielman, D.A., Srivastava, N., Teng, S.H.: Spectral sparsification of graphs: theory and algorithms. Communications of the ACM **56**(8), 87–94 (2013)
3. Benczúr, A.A., Karger, D.R.: Approximating st minimum cuts in õ (n 2) time. In: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing. pp. 47–55 (1996)
4. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. Journal of statistical mechanics: theory and experiment **2008**(10), P10008 (2008)
5. Chandra, A.K., Raghavan, P., Ruzzo, W.L., Smolensky, R., Tiwari, P.: The electrical resistance of a graph captures its commute and cover times. Computational Complexity **6**(4), 312–340 (1996)
6. Gancio, J., Rubido, N.: Community detection by resistance distance: automation and benchmark testing. In: Complex Networks & Their Applications X: Volume 1, Proceedings of the Tenth International Conference on Complex Networks and Their Applications COMPLEX NETWORKS 2021 10. pp. 309–320. Springer (2022)
7. Ghosh, A., Boyd, S., Saberi, A.: Minimizing effective resistance of a graph. SIAM Rev. **50**(1), 37–66 (Feb 2008)
8. Girvan, M., Newman, M.E.: Community structure in social and biological networks. Proceedings of the national academy of sciences **99**(12), 7821–7826 (2002)
9. Goldberg, D.E.: Genetic algorithms in search. Optimization, and MachineLearning (1989)
10. Hashemi, M., Gong, S., Ni, J., Fan, W., Prakash, B.A., Jin, W.: A comprehensive survey on graph reduction: Sparsification, coarsening, and condensation. arXiv preprint arXiv:2402.03358 (2024)
11. Jones, E., Faas, A.: Social Network Analysis of Disaster Response, Recovery, and Adaptation. Butterworth-Heineman (Elsevier) (2016)
12. Klein, D.J., Randić, M.: Resistance distance. Journal of mathematical chemistry **12**(1), 81–95 (1993)
13. Lancichinetti, A., Fortunato, S., Radicchi, F.: Benchmark graphs for testing community detection algorithms. Physical review E **78**(4), 046110 (2008)
14. Pizzuti, C., Socievole, A.: Multiobjective optimization and local merge for clustering attributed graphs. IEEE transactions on cybernetics **50**(12), 4997–5009 (2019)
15. Pizzuti, C., Socievole, A.: An effective resistance based genetic algorithm for community detection. In: IJCCI. pp. 28–36 (2021)
16. Radicchi, F., Ramasco, J.J., Fortunato, S.: Information filtering in complex weighted networks. Physical Review E **E83**, 046101 (2011)
17. Spielman, D.A., Srivastava, N.: Graph sparsification by effective resistances. Siam Journal on Computing (40), 1913 (1996)
18. Spielman, D.A., Srivastava, N.: Graph sparsification by effective resistances. In: Proceedings of the fortieth annual ACM symposium on Theory of computing. pp. 563–568 (2008)
19. Tumminello, M., Aste, T., Matteo, T.D., , Mantegna, R.N.: A tool for filtering information in complex systems. In: Proc. National Academy of Science. pp. 102,10421 (2005)

20. Yan, X., Jeub, L.G.S., Flammini, A., Radicchi, F., Fortunato, S.: Weight thresholding on complex networks. Physical Review E **E98**, 042304 (2018)
21. Zhang, T., Bu, C.: Detecting community structure in complex networks via resistance distance. Physica A: Statistical Mechanics and its Applications **526**, 120782 (2019)