Chauhan *et al. Applied Network Science* (2019) 4:46
https://doi.org/10.1007/s41109-019-0142-3

Applied Network Science

RESEARCH Open Access

Multiscale planar graph generation

Varsha Chauhan[1*†], Alexander Gutfraind[2] and Ilya Safro[1†]

*Correspondence:
varshac@clemson.edu
†Varsha Chauhan and Ilya Safro contributed equally to this work.
[1]School of Computing, Clemson University, Clemson, SC, USA
Full list of author information is available at the end of the article

Abstract

The study of network representations of physical, biological, and social phenomena can help us better understand their structure and functional dynamics as well as formulate predictive models of these phenomena. However, due to the scarcity of real-world network data owing to factors such as cost and effort required in collection of network data and the sensitivity of this data towards theft and misuse, engineers and researchers often rely on synthetic data for simulations, hypothesis testing, decision making, and algorithm engineering. An important characteristic of infrastructure networks such as roads, water distribution and other utility systems is that they can be (almost fully) embedded in a plane, therefore to simulate these system we need realistic networks which are also planar. While the currently-available synthetic network generators can model networks that exhibit realism, they do not guarantee or achieve planarity. In this paper we present a flexible algorithm that can synthesize realistic networks that are planar. The method follows a multi-scale randomized editing approach generating a hierarchy of coarsened networks of a given planar graph and introducing edits at various levels in the hierarchy. The method preserves the structural properties with minimal bias including the planarity of the network, while introducing realistic variability at multiple scales.

Reproducibility: All datasets and algorithm implementation presented in this work are available at https://bit.ly/2CjOUAS

Keywords: Planar graphs, Multiscale graph generation, Graph generators

Introduction

A network is a representation of a set of entities and the relationships between them. The network paradigm is often used to represent physical, biological, engineered and social systems (Newman 2018). Networks can help us better understand the structural and functional dynamics of these systems and formulate predictive models. However, collecting real-world network data often requires time and can be expensive. Also, for many applications, the sensitivity of real-world data towards theft and misuse further adds to the cost of protection and security of the data, which sharply limits its availability.

The problem of data scarcity can be tackled by using synthetic data which can mimic both the properties and diversity of real world networks. Such synthetic data can be used for simulations, analysis, and performance/quality verification of algorithms - a crucial task in algorithm engineering. Synthetic network generation is one of the most important fields in network science from both theoretical and practical perspectives. We refer the reader for an in-depth discussion to recent reviews in Gutfraind et al. (2015); Staudt et al. (2017).

Planar graph generation

Planar graphs are the class of graphs that can be embedded in a two-dimensional plane without edge crossings. Designing efficient algorithms for planar graphs is an important subfield in the area of algorithm development and optimization (Meinert and Wagner 2011). From the practical perspective, the planarity is also an important characteristic of many physical networks such as roads, utilities, water distribution systems, and some circuit designs. Many of these networks are, in fact, almost planar, that is, one can remove typically small fraction of edges to make them exactly planar.

The wide range of real-world applications of planar networked systems has created a demand for planar graph generators. Although the planar graphs share the property that they can be embedded in a plane, a planar graph generator should also be able to replicate other properties exhibited in a real-world networks. *However, the currently available synthetic network generators can either generate networks that exhibit realism with no planarity guarantees, or give planar networks with otherwise random structure that lack the structural characteristics of real-world networks.* Also, most of the existing research in general purpose network generation covers models related to scale-free networks, heavy-tailed degree distributions, and relatively high clustering coefficient that are not typical to real-world (almost) planar networks.

Our contribution

In this paper, we present a flexible algorithm that can synthesize realistic planar replicas of a known planar graph that can be rescaled to much larger graphs. The method follows the multi-scale editing approach (Gutfraind et al. 2015) in which a given graph is projected into a hierarchy of its coarsened representations (coarse graphs) that are then perturbed by edits at various scales of coarseness in the hierarchy. The method preserves the structural properties including the planarity with *controllable* bias, while introducing realistic variability at multiple scales of coarseness. Because the method belongs to the family of multiscale editing approaches, it generates planar graphs that attempt to replicate properties of the original graph at all levels of its coarse-grained resolutions which is the main property of the multiscale editing approach.

Throughout this paper we refer to the term "realistic" network multiple times. Realism of a generated similar network is not a uniquely defined notion as its meaning obviously depends on the application in which generating a similar to the original network is required. The question of realism definition is beyond the scope of this paper. We refer the reader to a discussion in Gutfraind et al. (2015) and its preliminary extended version (Gutfraind et al. 2012). Intuitively, the multiscale generative method suggests that a realistic network is the one that replicates some properties of the original network at multiple scales of coarseness (in contrast to many different methods that generate similar networks with predefined properties such as clustering coefficient and degree distribution only at the finest scale). We advocate that preserving them at multiple scales is at least as important for a variety of applications as at the finest scale. Technically, in many cases, preserving just a couple of such properties as the degree and second shortest distance distributions, will imply preservation of many path-based metrics such as betweenness and diameter which does not necessarily happen at the finest level only methods.

Network generation algorithms

The field of network science and, in particular, network synthesis is vast and cannot be comprehensively reviewed here. Hence, we focus on several particularly illuminating approaches for modeling realistic networks that presumably may be applied as or changed to the first step in realistic planar graph generator. In contrast to the different versions of random planar graph generators, there is an obvious lack (Barthélemy 2011) of planar graph generators that generate graphs that are similar to the original planar graphs. This is the reason, why practitioners and decision makers use other graph generators in combination with planarization postprocessing to generate planar and hopefully realistic graphs. This is also a reason for our comparison with these algorithms in the next sections. These approaches fall into categories, namely, sampling models, generative models and editing models.

Sampling models

Sampling models are typically used for large scale networks. In this technique we pick a subset of vertices and/or edges from original graph and calculate the distribution of various graph properties such as degree distribution or link probabilities. The network is then generated by sampling from estimated distribution. One of the important examples of this model is the **Exponential Random Graph Models (ERGM)** model.

The ERGM models (Hunter et al. 2008) are a class of statistical models, earlier called p-star models, that are popular in the study of large-scale social networks. To build a network, the ERGM first estimates certain parameters by fitting an observed social network and then constructs new networks by sampling from the estimated distribution. For example, in the Bernoulli and Erdös-Rényi ERGM models which generate homogeneous networks, the parameter space is based on same probabilities for each added connection, whereas the Chung-Lu ERGM model (Aiello et al. 2000) for large random graph with given degree distribution, the probability of connection of two nodes is proportional to the product of the degree of the nodes. The model can generate large graphs which depict some of the behaviors of massive realistic graphs and also predict the size and number of large components in the graph. ERGM models are successful in generating social networks and exhibit realistic degree distributions and small world structures. Also there are several ERGMs with community structure (Karrer and Newman 2011; Fronczak et al. 2013; van Lidth de Jeude et al. 2019) but none of them give any planarity guarantees, and normally violate planarity. While potentially, this model could serve as the first step in planar network generation (the planarity could be one of the properties or it can be applied with subsequent planarization of synthesized network), we emphasize that it is extremely slow and cannot be applied even on medium size networks, so we cannot experiment with it and compare to our generator.

Generative models

Generative models typically construct a network starting with an empty or small seed network and then iteratively add network elements (such as nodes and edges) to match some properties of a network that have to be preserved. These algorithms attempt to preserve the real network properties over the evolution and growth of the synthetic network. Important examples of generative models are the following.

BTER Block Two-Level Erdös-Rényi model (BTER) (Seshadhri et al. 2012) is based on the idea that a network contains communities that are Erdös-Rényi graphs in which each pair of vertices is independently connected with some probability. BTER graphs contain dense Erdös-Rényi communities that are found in real-world networks. The algorithm is two-phased. In the first phase, a collection of blocks or Erdös-Rényi communities with specified degree distribution is created. Then the blocks made interconnected and excess degree nodes are removed based on Chung-Lu (CL) model (Aiello et al. 2001) such that each subnetwork is well modeled by CL. BTER has been shown to model realistically a variety of network properties, but as with ERGM, it gives no guarantees of planarity. Also, whether communities in (almost) planar networks have hierarchical and connectedness structure similar to BTER model or not is not explored.

RMAT and Stochastic Kronecker Graphs The Recursive Matrix graph generator introduced by Chakrabarti et al. (2004) and its extensions AutoMAT-fast (Chakrabarti et al. 2004) can generate large-scale complex realistic networks. The generator is based on a recursive algorithm that operates on the adjacency matrix of the graph by dividing it into four equal-sized partitions and distributing edges to each partition based on fitting a set of parameters.

The Stochastic Kronecker Graphs (SKG) (Mahdian and Xu 2007) extends the methods of RMAT. Similarly to RMAT it is a recursive model, which starts with a small initiator matrix and recursively produces large graphs by applying Kronecker products. SKG can be interpreted as network which is a hierarchy of communities which grow recursively to create copies of themselves and every node has unique set of attributes values. The model can generate graphs with static patterns such as degree distribution as well as temporal patterns such as diameter over time. As before, planarity is not guaranteed as well as the community structure similarity with real-world networks that have one.

Multifractal Network Generator In 2010, Pallaa et al. (2010) introduced the multifractal network generator which can generate realistic networks with specified statistical features. The method starts with defining a generative measure on a single fractal or unit square and calculating link probability. The network is then scaled to the infinite limit by recursively dividing the fractal into a number of rectangles and introducing connections between them based on the link probability. Although this method was able to generate small scale realistic graphs the recursive method was slow for large complex networks. It is unknown if the generated networks can be constructed to have planar or quasi-planar structure, but the random nature of the construction suggests that planarity would be uncommon even in small graphs. However, the backbone networks generated by this model could be planar and thus possibly relevant to some infrastructure networks (for example, see major gas pipes in Newman (2010)). Unfortunately, these networks have layers of fractals and do not exhibit properties of infrastructure networks such as small diameters, shortcut edges and redundancy in paths. Thus making comparison of these networks with infrastructure networks impossible.

Editing models

The editing models approach starts with a given (real or empirical) network and controllably introduces random changes to its elements (such as nodes and edges) until the network becomes sufficiently different from the original network. These changes are designed to introduce variability while preserving key structural properties during the

random editing. Such methods are a promising direction for a relatively more realistic modeling of networks, and that includes properties such as planarity or near-planarity.

Edge-swapping The edge-swapping method (Tabourier et al. 2011; Rao et al. 1996) is perhaps the first important algorithm in the class of editing models, and it is based on the insight that the degree distribution of a graph is preserved under a chain of edge-swapping operations. Such a chain of edge swaps can even asymptotically achieve important mixing properties giving high variability. Despite these successes, edge-swapping operations can be very disruptive to planarity and other global properties of the graph, and there are no good post-selection methods for achieving planarity.

Multiscale Network Generation In Gutfraind et al. (2015), several of us proposed a strategy termed MUSKETEER (Multiscale Entropic Network Generator) for realistic graph generation. The main idea was based on the observation that the properties of real networks that should be preserved during generation are not only those measured at the finest resolution but also those that can be measured at the coarse resolutions. Multiscale generation leverages coarsening schemes used in highly-accurate multiscale solvers for combinatorial optimization such as linear arrangement, compression and partitioning (Ron et al. 2011; Hager et al. 2018; Safro et al. 2006; 2008; Safro and Temkin 2011). In such coarsening schemes, nodes in a network are assigned into aggregates (or, typically, very small communities) which are themselves parts of larger aggregates and so on in a hierarchical manner. The algorithm was successful in generating a number of replicas for several real-world original networks, but did not guarantee planarity. This paper continues this line of research and offers an implementation of the multiscale strategy that actually produces planar networks.

ReCoN Staudt et al. (2017) later used principles similar to those of multiscale method and developed a fast network generator that could generate large-scale replicas of real complex network that are structurally similar to original network. Instead of leveraging multiscale coarsening schemes, ReCoN generated synthetic networks by randomizing the edges between communities which were detected by the community detection methods while keeping the same degrees of nodes. ReCoN is built on top of the LFR generator implemented in Staudt et al. (2014).

Planar network generators

Planar networks with underling graphs have attracted a lot of attention since a landmark paper by Tutte (1963). Most of the research was dedicated on the study of structural properties (including their generation) of random planar graphs or uniform random planar graphs such as triangulations, and meshes. However, the currently available planar graph generators usually generate uniform random graphs by interpolation of planar subgraphs or generate planar subgraphs of a non-planar graph. Unfortunately, they are very far from being practically important for such tasks as generating graphs underlying infrastructure networks since they fail to present most other properties that are viewed as significant in this area, such as the degree distribution, the community structure and others. Some important available planar graph generators are discussed below.

Plantri and Fullgen software. Plantri (Brinkmann et al. 2007) can generate triangulations, quadrangulations, and convex polytopes using recursive algorithm which is efficient and fast. Fullgen (Brinkmann 2011) generates fullereness which are planar cubic graphs with 5 or 6 faces. The important characteristic of this software is that it generates

only one graph as output from a family of isomorphic graphs saving the space needed to store them. The software also offers the user the option to restrict adjacent pentagons using an input parameter.

Markov Chain Planar Graph Generator. This algorithm was proposed by Denise et al. (1996) and is based on Markov Chain that generates planar subgraphs from a non-planar graph. The algorithm defines a Markov Chain on the state space of all subgraphs of the original graph and transitions as follows. If an edge exists in space, it is deleted. If it is not present it is added in case it maintains planarity otherwise it is discarded. The method can successfully generate a planar subgraph in polynomial time.

Delaunay Triangulation and refinement method. This method has been widely used by researchers to generate mesh networks. In Shewchuk (1996), Shewchuk presented an implementation of 2-dimensional constraint Delaunay triangulation and Ruppert's (Ruppert 1995) Delaunay refinement algorithm for mesh generation.

Geometric graphs. Gilbert (1961) proposed a model to construct random plane networks by first selecting points in infinite plane based on Poisson process with a specific density and then connecting points based on their distance (a parameter) from each other. The random geometric graphs closely represent the graphs generated by percolation process through various porous materials and therefore these graphs are extensively utilized by physicists to study continuum percolation models. Random geometric graphs also have application in communication networks (Barthélemy 2011).

Planar Erdös-Rényi graph. In 1959, Erdös and Rényi (1959) introduced a method to generate a random graph with N nodes and m edges by connecting the edges randomly with independent probability p. The Erdos-Renyi planar graph generator generates random planar graph with uniform probability (Denise et al. 1996) by rejecting the non planar edges thereby preserving planarity (Denise et al. 1996; Gerke and McDiarmid 2004; McDiarmid et al. 2005; Barthélemy 2011). This is the most basic planar model which cannot be directly used for practical replication purposes.

Domain specific planar network generation

Important applications of planar networks are infrastructure networks such as roads, water distribution systems and power grids. There is a shortage of data for these networks owing to various reasons such as time and cost involved for collecting the data. Also the available data for infrastructure networks such as water distribution systems, and other utilities cannot be published due to confidentiality issues. As a result, the study of these networks and their simulation is highly dependent on the creation of high-fidelity synthetic data.

In Cura et al. (2015), Cura et al. proposed a unified framework called StreetGen that works on real Geographic Information System (GIS) data and modeling hypothesis which automatized street reconstruction and generated a street network model which was coherent to real-world street model. StreetGen required parameters for specific street objects and needed specialization for different objects.

In order to generate a simulation data for grid networks, Wang et al. (2008) proposed an algorithm that generates random but realistic topology power grid networks that could be used as test power grids. The generator used probability distribution for defining for number of nodal locations, then the parameters for distance was used to generate simple topology which was connected.

There is a similar shortage for data on water distribution systems (WDS) and the researchers typically need to rely on synthetic data to run simulation, test hypothesis and decision-making. The earlier methods for synthetic network generation involved manually creating realistic networks based on real data. Sitzenfrei et al. (2013) developed a software package DynaVIBe-Web that automated the generation of synthetic WDS networks. The generator used street networks, GIS data and real data from more than one network. Muranho et al. (2012) developed an interactive application WaterNetGen as an extension to well-known WDS optimization tool EPANET (Rossman 1994), which could generate network topologies, which is based on user-defined parameters and constraints.

Though, the above network generators produce realistic networks, they are specialized to specific domain and do not follow a generic approach to planar network generation. Also, replicating structural properties of a given network is not among the features of such generators. To our knowledge no work exists on cross-domain graph generators that generate realistic planar graphs that attempt to create graphs that are controllably similar to the given instances.

Notation

Throughout the paper, we will use the notation $G = (V, E)$ for a graph, where V is a set of n nodes, and E is a set of m edges. We consider simple, undirected graphs, where $ij \in E$ is an edge connecting nodes i and j, and the weight of ij is denoted as $w(ij)$ and node volume (total weight of aggregated nodes) is denoted by $v(i)$. Both weight of edge and node are non-negative. (Although, our generator is not expected to work with weighted graphs, the weights will be used at the coarse levels to reflect aggregated nodes and edges).

The subscript (such as G_i) used with a variable denotes the number of level in multiscale hierarchy. We denote an edited network in the hierarchy at level i using superscript and subscript (such as G_i^d). A finally generated network at the level i is denoted by G_i^g.

Multiscale planar graph generation

Multiscale network generation (MNG) introduced in Gutfraind et al. (2015) is an editing model that generates realistic networks. The proposed multiscale planar graph generator follows the main ideas of MNG and makes them applicable on planar graphs. For the completeness of the paper we also overview basic components of the original method.

MNG follows a multilevel coarsening/uncoarsening scheme shown in Fig. 1. We start with an input graph G and generate a hierarchy of next coarser graphs, $G_0 = G, G_1, ..., G_k$, where k is the number of coarsest level. The number of coarsened levels depends on the structure and size of G, and user input, which is a vector where each value determines the required edit or growth rate at each level of the hierarchy, and the length of vector determines the number of desired coarsening levels. The hierarchy construction (coarsening) is terminated if the graph is too small or too dense (density of graph > 0.9) at some level (i.e., the coarsest level is reached). The coarse level construction is generic and based on the weighted aggregation method for combinatorial optimization problems (Safro et al. 2006; Ron et al. 2011). Currently, it does not depend on the application predefined aggregates in the network such as knowledge about real communities. However, this process can be adjusted as we did in Staudt et al. (2016).

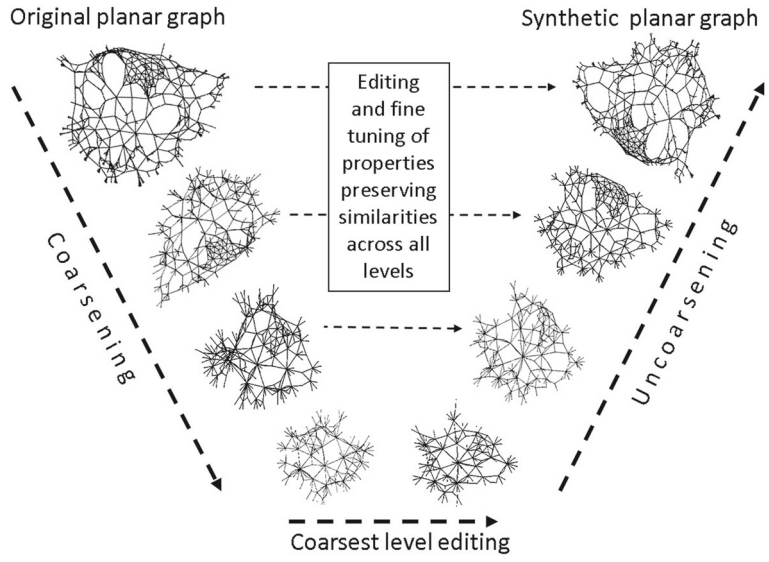

Fig. 1 The V-model for multiscale planar network generation. The original input planar network is coarsened to generate a hierarchy of coarse networks, the process is then reversed generating fine-level networks. The number of level (here 5) depends on the size of input network or the user input

In order to generate a synthetic graph, during the uncoarsening stage, we introduce a series of local randomizations at different levels whose amounts can be specified by user's input. As mentioned previously, the user can control the number of levels coarsening and amount of edits or perturbations at each of these levels using parameters edge edit rate and edge growth rate. If user is interested in only local changes without destroying the global structure of the network, only fine levels are specified for randomizations. Otherwise, any realistic changes in global structure will require randomizations at coarse levels. During the uncoarsening, these randomizations are carried forward to the next finer level in the hierarchy. In Algorithm 1, we describe the sequence of steps in generating planar graph. We will now discuss each phase and notation in detail and our approach to generate planar graphs using the multiscale method.

Algorithm 1 Multiscale Planar Network Generator MPNG(G_i)

if G_i is not small or too dense or perturbations are required for G_i at level i by user **then**
  $G_{i+1} \leftarrow$ create aggregated network from G_i ▷ see Alg. 2
  $G^g_{i+1} \leftarrow$ MPNG(G_{i+1}) ▷ Return coarser edited network from recursive call
  $G^{d'}_i \leftarrow$ interpolateUneditedAggregates from G^g_{i+1}
  $G^d_i \leftarrow$ interpolateEditedAggregates$\left(G^g_{i+1}, G^{d'}_{i+1}\right)$
end if
$Q_i \leftarrow$ measure properties of G_i
$G^g_i \leftarrow$ editing G^d_i preserving Q_i
Return G^g_i

Coarsening

Since the input graph G_0 is planar, the aggregation algorithm that creates coarsened graphs G_i makes them also planar, so we follow the same coarsening scheme as that in the original MNG. Algorithm 2 lists the steps involved for generating coarse level graph G_{i+1} from G_i.

Algorithm 2 Coarsening(G_i)

1: **if** G_i is not the coarsest graph **then**
2: Find set of seed nodes (C) for coarse network G_{i+1}
3: Find fine-level nodes that belong to each aggregate
4: Calculate weight of edges connecting aggregates and weights of coarse nodes
5: Return G_{i+1}
6: **end if**

At each level of the hierarchy, we start with finding set of seed nodes, C, and its complement fine-level nodes F which is based on two rules: (a) nodes with high volume and connectivity (i.e., major aggregates) are more likely to be included in C, and (b) the remaining nodes in F should be "strongly" coupled to enough neighbors in C. At level i, given a graph G_i, to generate coarse level nodes for G_{i+1} we begin with $C = \emptyset$ and $F = V_i$ where V_i is set of nodes at fine level G_i. Next, we iteratively transfer some nodes from F to C (visiting them one after another in random order), such that currently visited node $i \in F$ is added to C if it is not well connected to those already chosen to C (Safro et al. 2006). The connection strength between nodes i and j is determined by means of normalized weight of edge ij with respect to C, namely, if node $i \in F$ is not connected strongly enough to the currently chosen C, i.e.,

$$\frac{\sum_{j\in C} w(ij)}{\sum_{j\in V} w(ij)} \leq \alpha, \tag{1}$$

then we move i to C and transfer our attention to the next node in F. Thus, instead of requiring for a certain number of F-nodes to be transferred to C, we scan them iteratively, and decide based on Eq. (1). The connection strength is parametrized using threshold α which determines the speed and number of coarse nodes (and implicitly edges). Big values of α (in multiscale algorithms, typically, 0.7 or bigger) will result in small changes in coarsened graphs that are created from level to level as most of nodes wont be strongly connected to C (according to Eq. 1). In contrast, small values (in multiscale algorithms, typically, 0.3 or smaller) will cause a decreased number of levels as not too many nodes will be transferred to C. In all our experiments we have used α as 0.5, which guarantees uniform coarsening. However, we note that the strength of connectivity criterion in network generation requires further investigation similar to that in multiscale optimization (Brandt and Ron 2003) where it plays a crucial role in the solution quality.

The final phase of coarsening is computing the connection strength between the coarse nodes. Here we define the algebraic multigrid interpolation matrix P of size $|V| \times |C|$ (for details see Safro et al. (2006)) in which P_{uv} represents the likelihood of $u \in V_i$ to belong to the aggregate seeded with node $v \in V_i$. The Laplacian of the coarse graph G_{i+1}, L_{i+1},

can be calculated by the algebraic multigrid coarsening operator $L_{i+1} \leftarrow P^T L_i P$ where, L_i is the Laplacian of ith level graph, and

$$P_{uv} = \begin{cases} 1, \text{ for } u \in C, v = u \\ 0, \text{ otherwise.} \end{cases} \tag{2}$$

The edge st connecting two coarse nodes $s \in V_{i+1}$ and $t \in V_{i+1}$, is assigned with the weight

$$\sum_{k \neq l} P_{ks} w(kl) P_{lt}$$

and the volume of the coarse aggregate seeded by $u \in V_i$ is $\sum_j v(j) P_{ju}$.

This step finalizes creating the $(i+1)$th level graph, and we can measure the properties of ith level graph and store them in Q_i. As in the original MNG, we attempt to preserve the small loop structure of the network by avoiding any insertion of edges that connect nodes that were previously separated by the long distances in the original network. This is done by estimating the empirical probability of closed random walks (of limited length) that start at some node whose degree is at least two (for details see Gutfraind et al. (2015)). In general, this step is application dependent as in different applications the preserved properties may vary. Because, in planar graphs of infrastructures it is important to generate realistic path lengths (e.g., not to create shortcuts that connect distant regions in a graph), we are sampling using random walks the distribution of path lengths and shortcuts (second shortest distance between nodes) and store them in Q_i.

Uncoarsening

Once the coarsest level is reached, we start the uncoarsening. During this process, at each level $i+1$ we choose nodes and edges to be edited (randomized while keeping some properties preserved), to generate edited network G^g_{i+1} at level $i+1$ and then project the newly created graph to generate the next finer level G^g_i.

The projection is done in two steps. First, we interpolate the unedited aggregates (nodes and edges) in **interpolateUneditedAggregates** (see Algorithm 3) from G^g_{i+1} to generate graph $G^{d'}_i$. This process is just a reverse interpolation of aggregates based on aggregation data stored during the coarsening phase. *Because the input network at all levels is planar, the interpolated edges do not violate planarity.* This helps in preserving structural properties of original input network, as after this step we have a subgraph $G^{d'}_i$ of original network coarsened at level i.

Algorithm 3 interpolateUneditedAggregates $\left(G^g_{i+1}\right)$

1: $G^{d'}_i \leftarrow$ uncoarsen nodes from G^g_{i+1} using data stored during coarsening at level i
2: $G^{d'}_i \leftarrow$ uncoarsen unedited edges G^g_{i+1} using data stored during coarsening at level i
3: Return $G^{d'}_i$

In the next step, we interpolate the edited aggregates in function **interpolateEditedAggregates** to generate graph G^d_i by adding edited nodes and edges to graph generated with Algorithm 3. The pseudocode for this step is presented in Algorithm 4. We first interpolate the edited nodes and add edges that were trapped within the aggregates (or coarse) nodes connecting the interpolated fine nodes, i.e., these are edges that connect fine nodes

that are coarsened within the same coarse node. Next we interpolate edited or new edges introduced during editing phase at level $i + 1$. In this step, we introduce new fine level edges for every coarse edge connecting a pair of aggregates u and v. This is done by randomly selecting the fine level nodes generated by interpolating u and v. The number of fine level edges added for each coarse edge is based on the degree distribution of nodes in aggregates u and v at level i as stored in data structure during coarsening. This interpolation is likely to introduce crossing over edges, therefore, when we add an edge ij to G_i^d, we check if the network is still planar. If it is not, the edge is discarded. If an edge is discarded, we perform several iterations and find an edge which is similar to the edge ij using properties stored in Q_i during coarsening in Algorithm 1 (i.e., in our implementation, the short loop structure).

Algorithm 4 interpolateEditedAggregates (G_{i+1}^g , $G_i^{d'}$)

1: $G_i^d \leftarrow$ uncoarsen nodes from G_{i+1}^g
2: $G_i^d \leftarrow$ uncoarsen edited edges G_{i+1}^g
3: Return G_i^d

After the interpolation is complete and we have a fine-level graph G_i^d, on which we introduce randomizations or editing (discussed below in detail) specified by the user at level i to generate a finer-level random planar network G_i^g. The topology of the final network depends on the level at which the changes are introduced and the number of edited network elements both dependent on user input. At the coarsest level, every network element is an aggregate which interpolate of many network elements at fine level, a small change introduced at this level may generate high-entropy changes which are carried forward to the next fine level, whereas addition of an element at fine levels may introduce elements to the final synthetic network. In general, the changes introduced to deeper levels of aggregation, the more significant changes are introduced in the topology.

Editing

In the final phase we measure the properties of the generated graph G_i^d and compare with the properties of original graph G_i coarsened at level i which is stored in Q_i, thus preserving the local topological structure of the network and preventing addition of edges between nodes which were separated by long distance in original network at coarse level G_i stored in Q_i. We then use an editing process which introduces randomizations in the network to generate a synthetic network. This is a process of deleting and adding new edges whose both amounts depend on the user input for level i (namely, how much randomization is required at each level, i.e., a value from 0 - no randomization, to 1 - everything is randomized). In particular, we are interested in two properties, namely, the second shortest path length distribution (to prevent generating unrealistic shortcuts) and planarity. The first property, second shortest path length $\mathsf{spath}(u, v)$ for an edge uv is the number of edges in the shortest path $u \rightarrow v$ that does not include edge uv. Before introducing edits we estimate the distribution of spath at level i, $\mathbb{P}_i$ by random sampling of edges $R \subset E_i$ where E_i is set of edges in graph G_i as,

$$\mathbb{P}_i[\,d] \approx \frac{|\{\{u, v \in R : \mathsf{spath}(u, v) = d\}|}{|R|} \tag{3}$$

When we delete an edge u_1v_1 such that $\mathsf{spath}(u_1, v_1) = d$, we choose a random node u and v at randomly drawn distance from u using $\mathbb{P}_i$. This sampling of distances preserves multiple structural properties such as clustering coefficient and average shortest path (see more details in Gutfraind et al. (2015)).

The second critical property is planarity. In order to preserve planarity, if inserting the new edge makes the network non-planar we discard it and find an alternate edge that preserves the desired structural properties (in this case the first property) as well as planarity. Technically, it is done by verification of existence of Kuratowski subgraph (Thomassen 1981) after adding a new edge. This step is repeated until we find a non-crossing edge that preserves the planarity of the network and thus generating synthetic planar graph G_i^g at coarse level i.

Rescaling

Rescaling is a part of the editing phase in which we add new elements (edges and nodes) to the synthetic network in order to increase its size. The scaling factor and the coarsened level at which the network is rescaled is controlled by node growth parameter which is provided as an input from the user depending on the user requirement. In general, rescaling at coarsest levels will preserve the local structure of the input network, i.e. the generated network will have increased number of communities whereas rescaling at finer levels will increase the size of communities. The scaling factor ranges from 0 to any number which decides the percentage of new nodes that are to be added at the level i. This is a two step process. First, we introduce a new node u and connect to an existing node v in the network deleting an existing edge from v to restore the degree of node v. In the next step, we find neighbors of v iteratively over increasing distance from v and connect the newly added node u to the neighbors of node v thus preserving the local topological structure of the network at coarse level G_i stored in Q_i. The process is terminated when the desired number of network elements are added and a rescaled network G_i^g is generated at coarse level i.

Computational experiments

In this section we show the computational results summarizing the performance of our multiscale planar network generator in replicating the original and also generating rescaled networks. To test the variability of the generator we used real-world infrastructure networks such as water distribution system, power grid and road network that are either planar or have very few edge crossings that we removed. We used the water network from "The Battle of the Water Networks II" (Ostfeld et al. 2008) and for road network we used a sub graph of Texas (Leskovec et al. 2009) road network from (Leskovec and Krevl 2014). We also used a finite element large planar sub-graph of a finite-element graph from Boeing collection in Davis (1997). In case of the power grid (Leskovec and Krevl 2014) which was not completely planar, we generated approximate maximal planar subgraph of the network using Open Graph Drawing Framework (OGDF) (Chimani et al. 2013) to be used as an input to our algorithm.

Replication

We tested our implementation on three sets of parameters, namely, "Musketeer Coarse" (at only two coarsest levels 5% randomizations are allowed), "Musketeer Fine" (at only

two finest levels 5% randomizations are allowed), and "Musketeer All" (small 1% randomizations are allowed at all levels). The amount of edits for each set of parameters are controlled such that the number of randomizations in final generated graphs for each set of parameters are comparable. Because randomizations and editing are introduced at all levels, even very little changes at the coarse levels will result in significant changes at the finest level in generated synthetic graph.

We generated 30 network replicas for each network and compared the replicas with the original network based on the following metrics: number of nodes and edges, number of components, clustering coefficient, average degree, total degree-degree assortativity, average harmonic distance, modularity, pagerank and average betweenness centrality. We also compared our results with the existing generative models implemented in Staudt et al. (2014), namely, ReCoN, RMAT and BTER and stochastic Kronecker graphs by generating replicas of same input network. Since, these models do not necessarily generate planar network, *we post-processed the generated networks to find the maximal planar subgraph of the replicas* using OGDF library which uses edge removing technique, i.e., it adds one edge at a time while preserving planarity, if addition of the edge results in a non-planar graph then the edge is discarded thus generating a planar subgraph. We compared the generated planar graphs with the original graphs for the structural properties mentioned above. Clearly, one may argue that these generators were not developed to planar networks. We, however, note that these methods with planarization post-processing were chosen because there is no other acceptable solution to generate more or less realistically looking planar network that is similar to the original. As mentioned earlier, the available planar graph generators are generative models which either create specific classes of graphs with restricted values for minimum degree and connectivity (e.g., plantri and fullgen or Delaunay triangulation methods) or generate random realistic spatial networks based on give probability p (e.g., planar Erdos–Renyi, and spatial Watts-Strogatz generator). Other examples include domain specific generators for road networks (e.g., StreetGen) and power grid random networks that are not necessarily planar networks. To the best of our knowledge, there is no domain independent generator whose goal is to preserve similarity with the input network.

The structural properties of the replicas were normalized such that 1 denotes the property of original network. We performed 30 experiments for each set of parameters the results for which is graphically represented in Figs. 2, 3, 4, 5, 6, 7, 8 and 9. Our results indicate that multiscale planar graph generator can generate replicas that preserve almost all the properties of the original networks with relatively small deviation. Also, we observe that graphs generated by BTER and RMAT after planarization are close to original network (within $0 - 2$, where 1 represents the property of original input network after normalization) for properties such as average degree and mean harmonic distance whereas the properties for networks generated by stochastic Kronecker graphs (SKG) are far from those in the original graphs. As such the plots for properties for the networks generated by SKG are not represented in the plots. However, we note that the distortion of properties on the replicas by other network generators may have been the result of the post-processing step (maximal planar sub-graph of the generated replica), which often created more than one connected components.

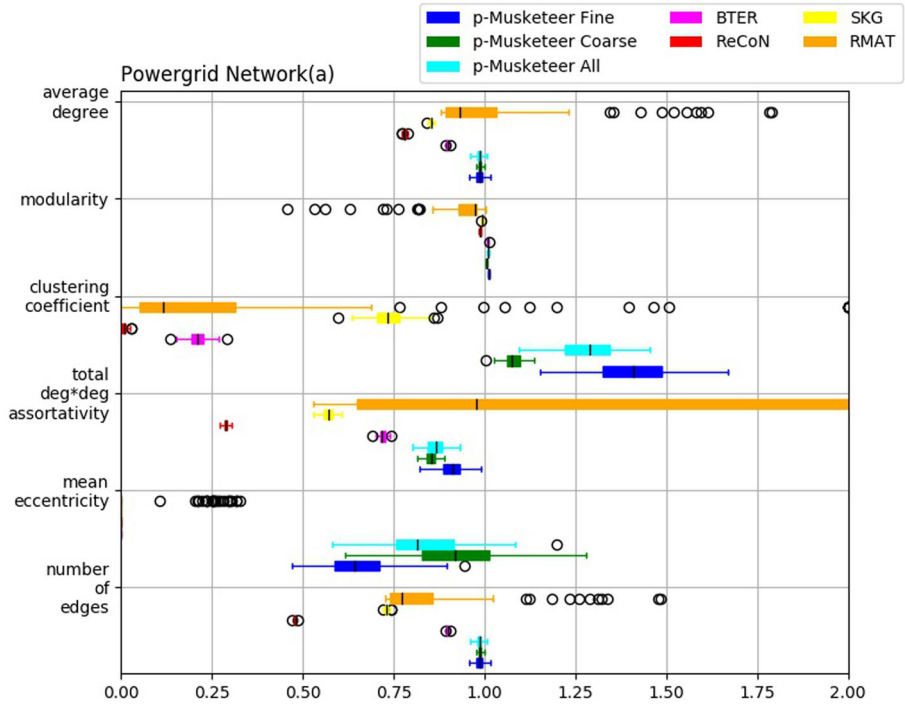

Fig. 2 Computational results on performance of planar Musketeer on power grid graph opsahl-powergrid with 4941 nodes and 6211 edges for clustering coefficient, number of edges, mean eccentricity, total degree *degree assortativity, modularity and average degree

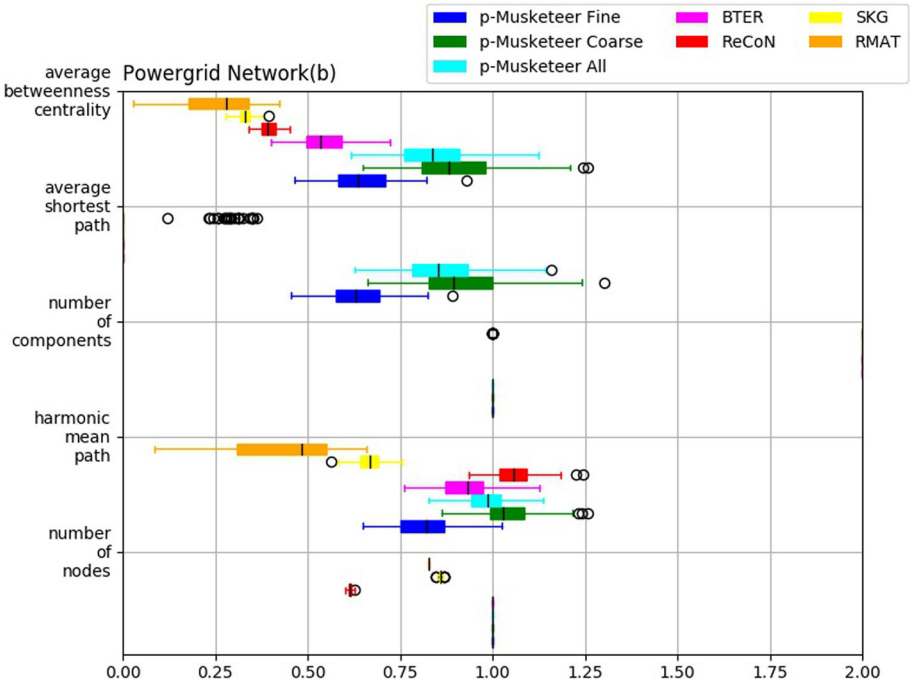

Fig. 3 Computational results on performance of planar Musketeer on power grid graph opsahl-powergrid with 4941 nodes and 6211 edges for number of nodes, harmonic mean path, number of components, average shortest path and average betweenness centrality

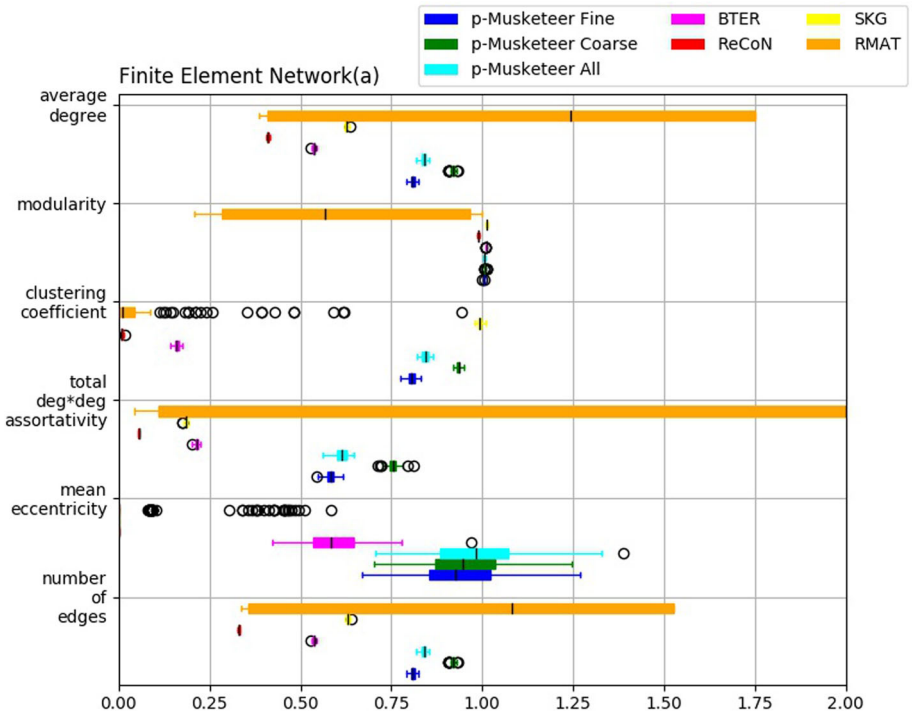

Fig. 4 Computational results on performance of planar Musketeer on finite-element graph with 4704 nodes and 13427 edges for clustering coefficient, number of edges, mean eccentricity, total degree *degree assortativity, modularity and average degree

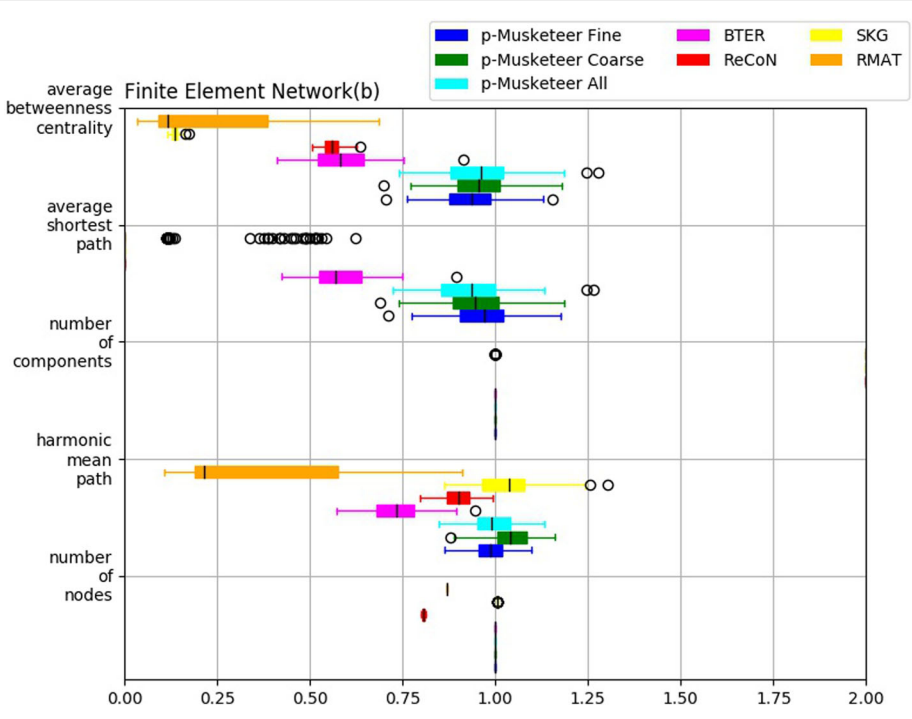

Fig. 5 Computational results on performance of planar Musketeer on finite-element graph with 4704 nodes and 13427 edges for number of nodes, harmonic mean path, number of components, average shortest path and average betweenness centrality

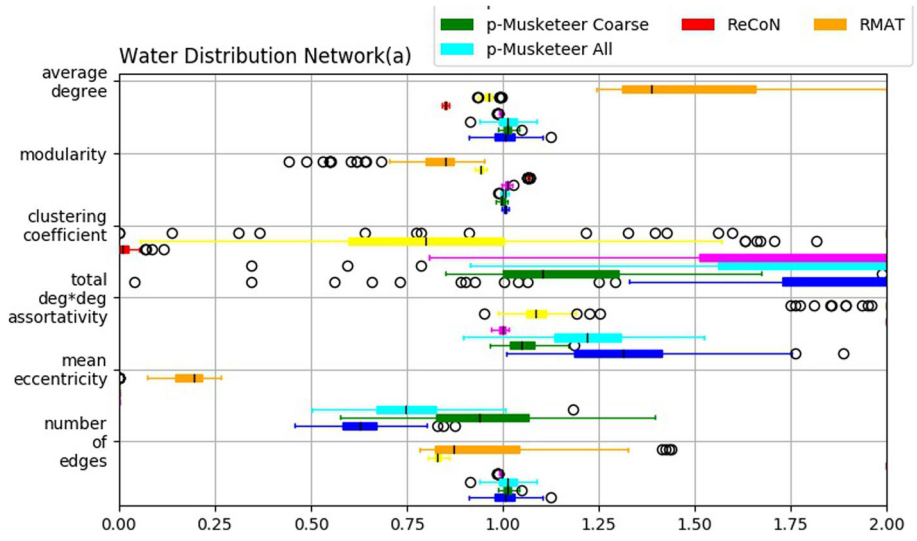

Fig. 6 Computational results on performance of planar Musketeer on real water network with 407 nodes and 459 edges for clustering coefficient, number of edges, mean eccentricity, total degree *degree assortativity, modularity and average degree

Rescaling

Our second set of experiments was designed to generate rescaled networks. We tested our implementation on three sets of parameters, namely, "Musketeer Coarse" (30% edge and node addition on 4 coarsest levels are allowed), "Musketeer Fine" (30% edge and node addition on 4 finest levels are allowed), and "Musketeer All" (15% edge and node

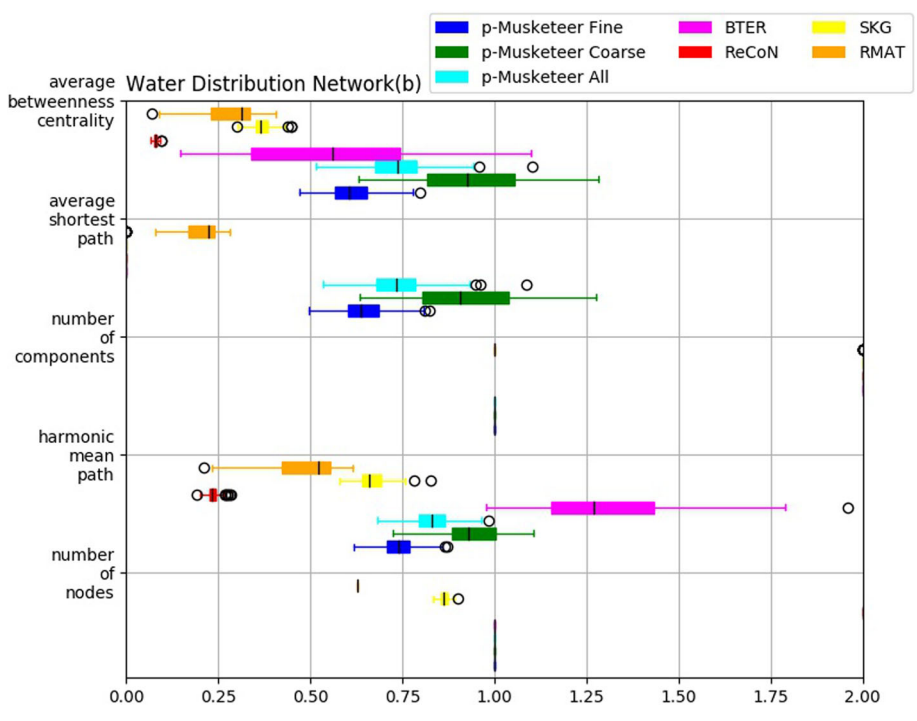

Fig. 7 Computational results on performance of planar Musketeer on real water network with 407 nodes and 459 edges for number of nodes, harmonic mean path, number of components, average shortest path and average betweenness centrality

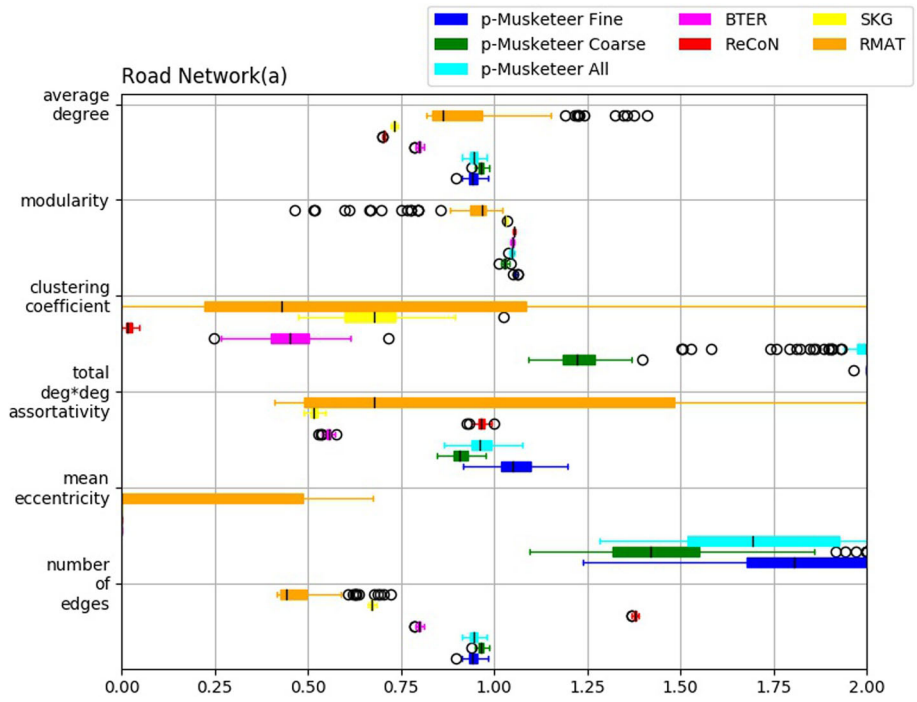

Fig. 8 Computational results on performance of planar Musketeer on road network from roadNet-TX with 2001 nodes and 2957 edges for clustering coefficient, number of edges, mean eccentricity, total degree *degree assortativity, modularity and average degree

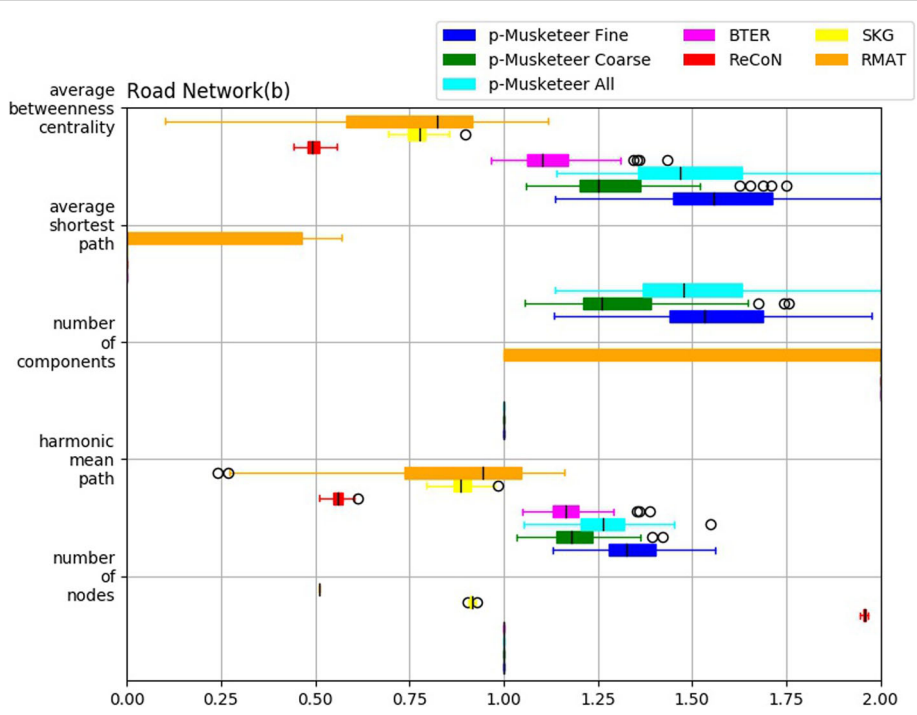

Fig. 9 Computational results on performance of planar Musketeer on road network from roadNet-TX with 2001 nodes and 2957 edges for number of nodes, harmonic mean path, number of components, average shortest path and average betweenness centrality

addition at all levels are allowed). The parameters are chosen such that the generated network has $3-4$ times the number of nodes and edges than the original network. We generated 30 rescaled replicas for the same dataset as used in our previous experiment and compared the generated networks with the original network based on the following metrics: number of components, clustering coefficient, average degree, total degree-degree assortativity, average harmonic distance, modularity, pagerank and average betweenness centrality.

The structural properties of the replicas were normalized such that 1 denotes the property of original network. The comparison for 30 experiments is presented in Figs. 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 13, 14 and 15. As depicted in the plots we are able to preserve almost all the properties of original network even when the network is rescaled to more than 3 times the original network. Also, there is no significant variance observed in properties for the three different sets of parameters (coarse,fine and all) used to generate rescaled networks. However, we observed that rescaling by introducing elements at finer levels results in high clustering coefficient in generated network. This is because the planarity constraint restricts addition of long edges (edges between nodes which are far from each other) which in turn forces the algorithm to connect new elements locally at each level i. In case the network elements are introduced at coarsest levels, the locally added edges and nodes are uncoarsened to several finer edges and nodes over the V-cycle of coarsening and uncoarsening, and the near neighbors at level i are drifted apart at level $i+1$.However, network elements added at fine levels are not drifted as a result of levels of coarsening and uncoarsening as described above, and the edges still connect the nodes locally. Hence, we observe an increased number of triangles (Fig. 16) or high clustering coefficient (Figs. 17 18, 19, 20, 21, 22 and 23) for networks generated by introducing elements at fine level as compared to coarse level. As depicted in Fig. 16 when the network is rescaled by introducing new elements at only coarse levels, we find larger

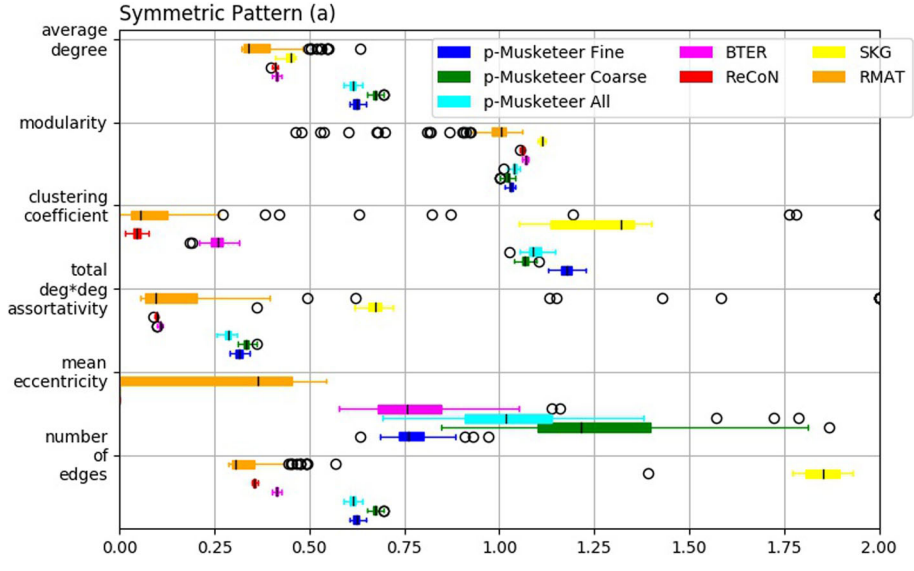

Fig. 10 Computational results on performance of planar Musketeer on symmetric pattern with 1141 nodes and 3162 edges for clustering coefficient, number of edges, mean eccentricity, total degree *degree assortativity, modularity and average degree

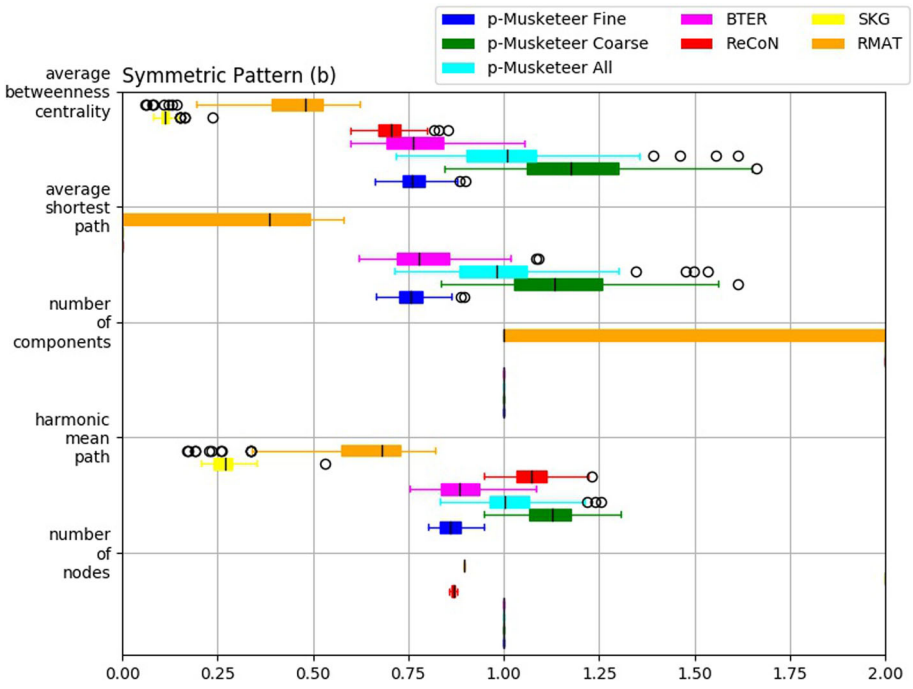

Fig. 11 Computational results on performance of planar Musketeer on symmetric pattern with 1141 nodes and 3162 edges for number of nodes, harmonic mean path, number of components, average shortest path and average betweenness centrality

communities (e.g., mesh structures in case of our input road network) in the generated network, whereas if the network is rescaled at fine level we observe smaller communities. The amount of new introduced elements can be controlled by user input which is provided as node growth parameters at certain levels. The size of replicated and edited aggregated clusters in the rescaled network can be controlled by choosing larger node

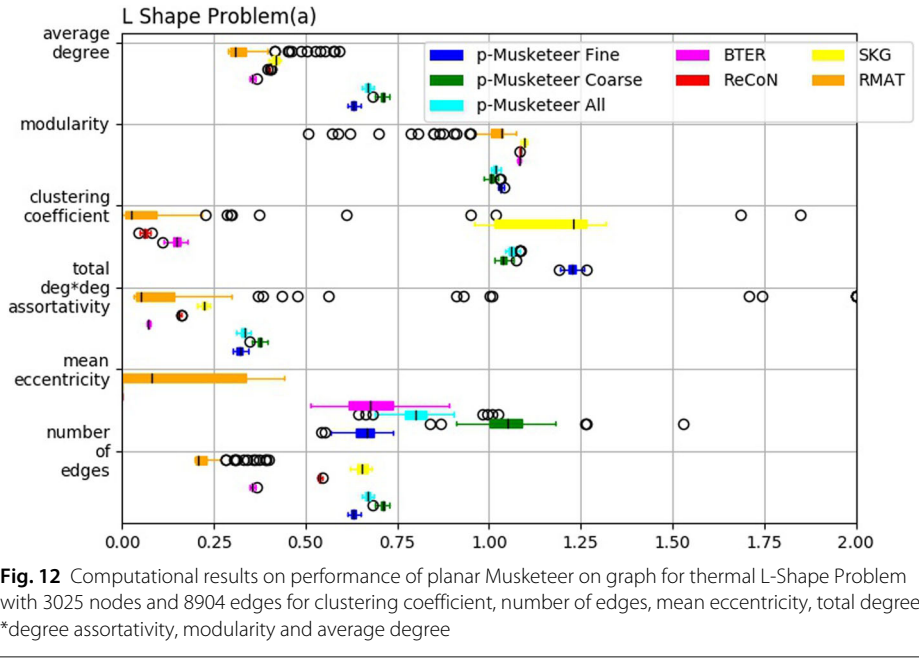

Fig. 12 Computational results on performance of planar Musketeer on graph for thermal L-Shape Problem with 3025 nodes and 8904 edges for clustering coefficient, number of edges, mean eccentricity, total degree *degree assortativity, modularity and average degree

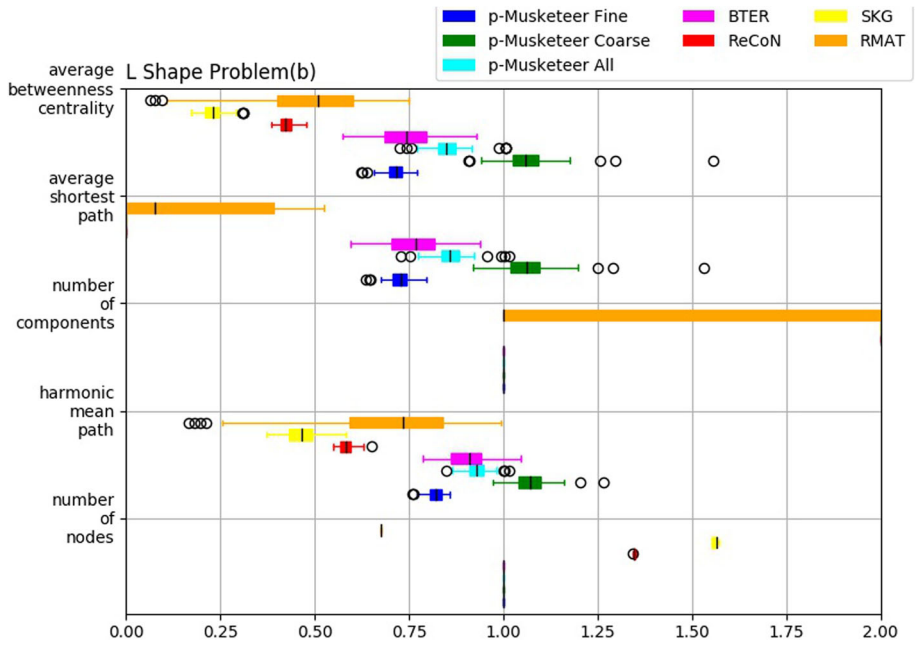

Fig. 13 Computational results on performance of planar Musketeer on graph for thermal L-Shape Problem with 3025 nodes and 8904 edges for number of nodes, harmonic mean path, number of components, average shortest path and average betweenness centrality

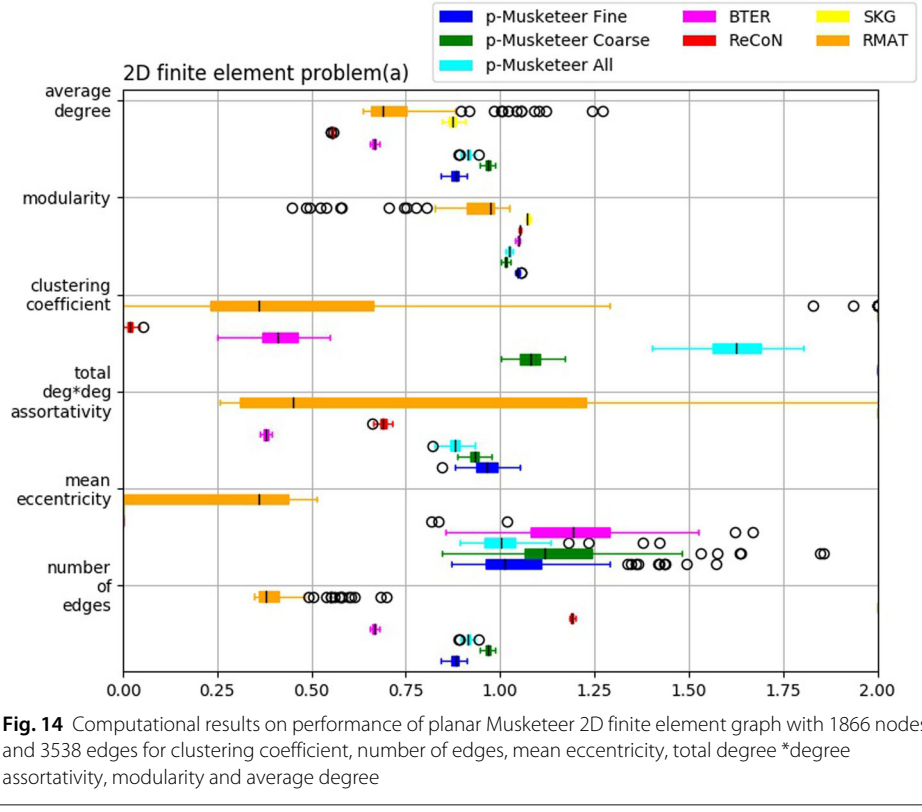

Fig. 14 Computational results on performance of planar Musketeer 2D finite element graph with 1866 nodes and 3538 edges for clustering coefficient, number of edges, mean eccentricity, total degree *degree assortativity, modularity and average degree

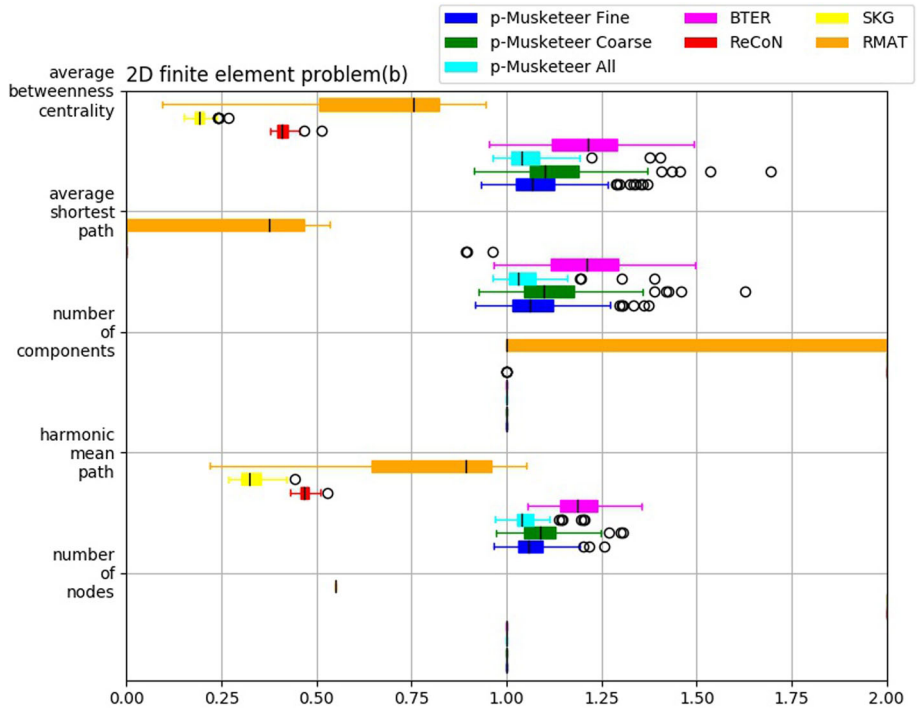

Fig. 15 Computational results on performance of planar Musketeer 2D finite element graph with 1866 nodes and 3538 edges for number of nodes, harmonic mean path, number of components, average shortest path and average betweenness centrality

growth parameter for example in Fig. 24 we used node growth parameter as 1.5 at the coarsest level to rescale the network to 1.5 times. Depending on the application, one may want to add a postprocessing step which will create longer loops as in the original network of Fig. 24. We created 13 links that close peripheral clusters by randomly choosing pairs of disconnected nodes, adding edge, and checking the planarity. However, although it may create a better visualization for the comparison with the original network, this step may not be desired by many application.

In our experiment we used 0.3 as node growth rate for coarsest and finest level and 0.10 when introducing network elements at all levels.

Conclusions

In this paper we introduced a multiscale planar graph generation framework and its implementation using Musketeer framework (Gutfraind et al. 2015) . Our evaluation suggest that multiscale planar graph generation method can generate realistic replicas of planar networks across domains with small loss of similarity. While there are clearly enough space for the improvement of this method, to the best of our knowledge, this is the first general purpose synthetic planar graph generation method that is able to produce realistic instances.

Several future research directions can be explored. First, we would like to introduce the algebraic distance edge weighting scheme (Chen and Safro 2011) in order to more accurately preserve the distances during the uncoarsening. We have successfully used this improvement for network sparsification (John and Safro 2016) and several combinatorial

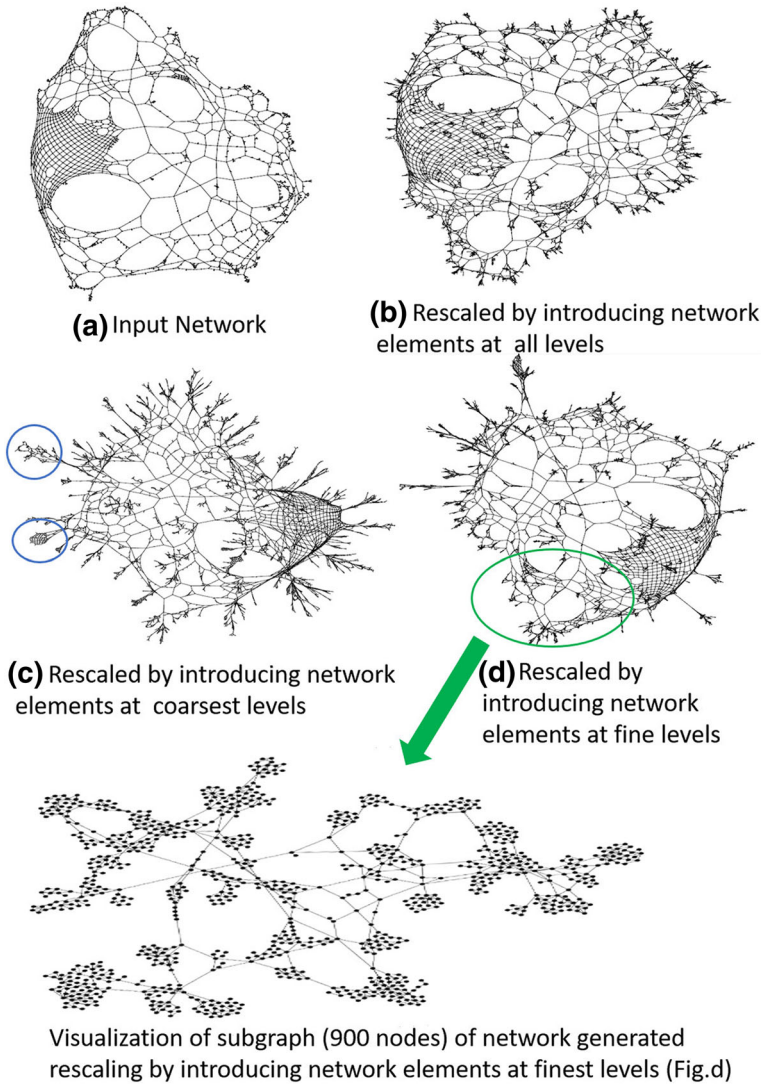

Fig. 16 Visualization of a road network from roadNet-TX with 2001 nodes and 2957 edges rescaled to at least 5900 nodes and 7000 edges. **a** Input network. **b** Rescaled by introducing network elements at all levels. **c** Rescaled by introducing network elements at coarsest levels. **d** Rescaled by introducing network elements at fine levels

optimization solvers (Safro et al. 2015; Ron et al. 2011). Second, it would be interesting to investigate whether one should control the size of aggregates to preserve uniform coarsening, a multilevel technique that is well known in graph partitioning (Safro et al. 2015; Buluç et al. 2016). The role of uniform coarsening is not well understood in multiscale graph generation.

There exist several further research directions that are application dependent. For example, approaches for assessing how much variation is desirable in the generation and how to decide whether enough changes have been introduced can vary from application to application. However, a unified strategy to address this issue would be very helpful.

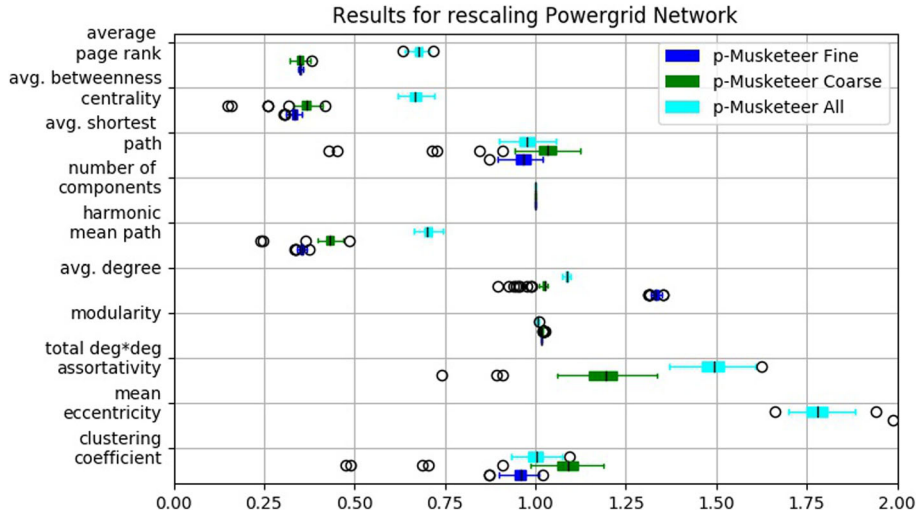

Fig. 17 Computational results for road network from roadNet-TX with 2001 nodes and 2957 edges rescaled to at least 6000 nodes and 6500 edges

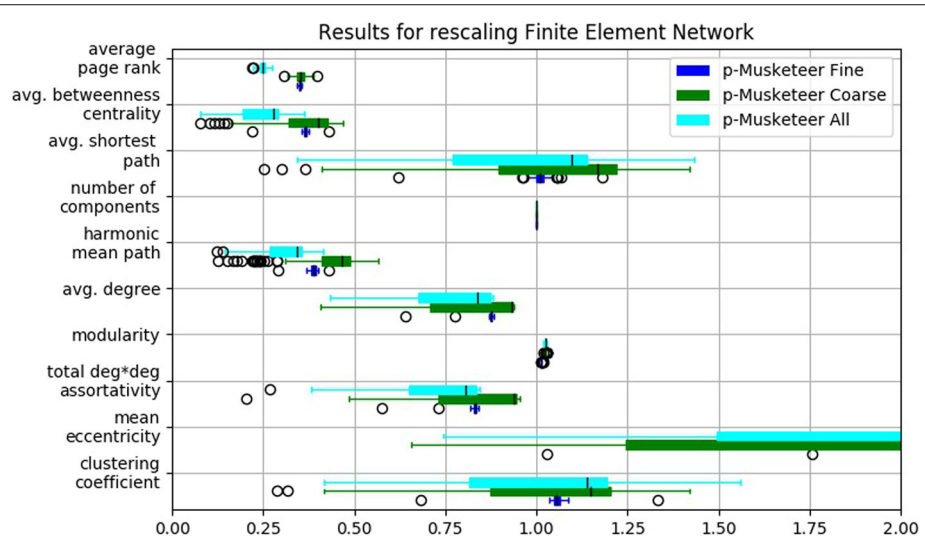

Fig. 18 Computational results for road network real water network with 407 nodes and 459 edges rescaled to at least 1098 nodes and 1500 edges

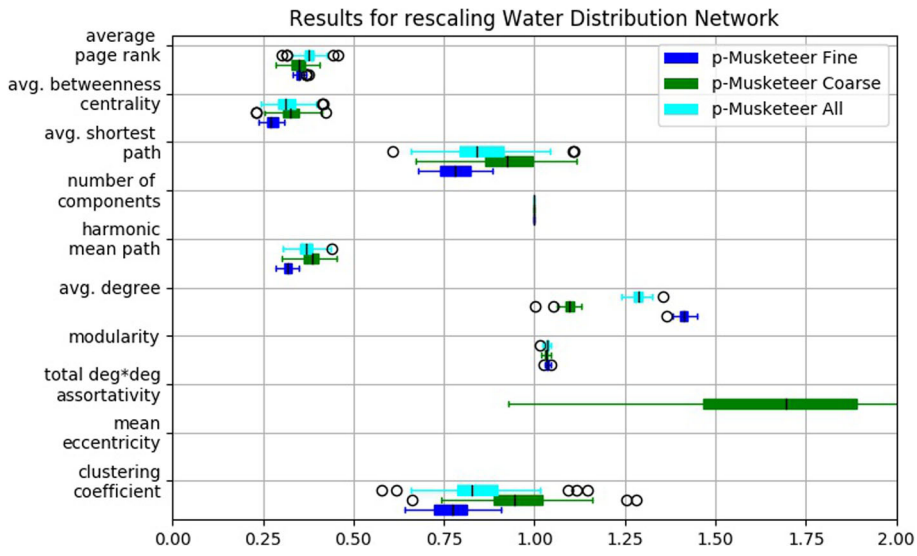

Fig. 19 Computational results for finite-element graph with 4704 nodes and 13427 edges rescaled to at least 12700 nodes and 36000 edges

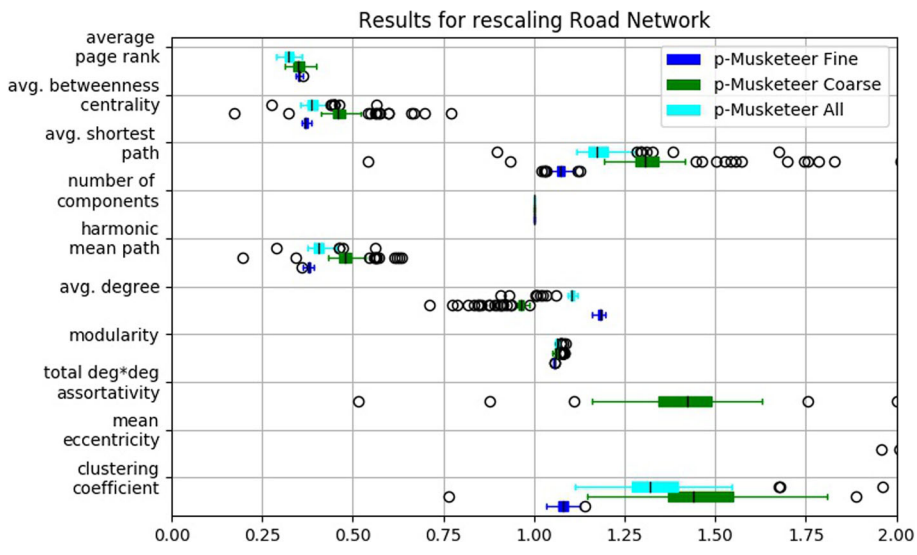

Fig. 20 Computational results for power grid graph opsahl-powergrid with 4941 nodes and 6211 edges rescaled to at least 16500 nodes and 27000 edges

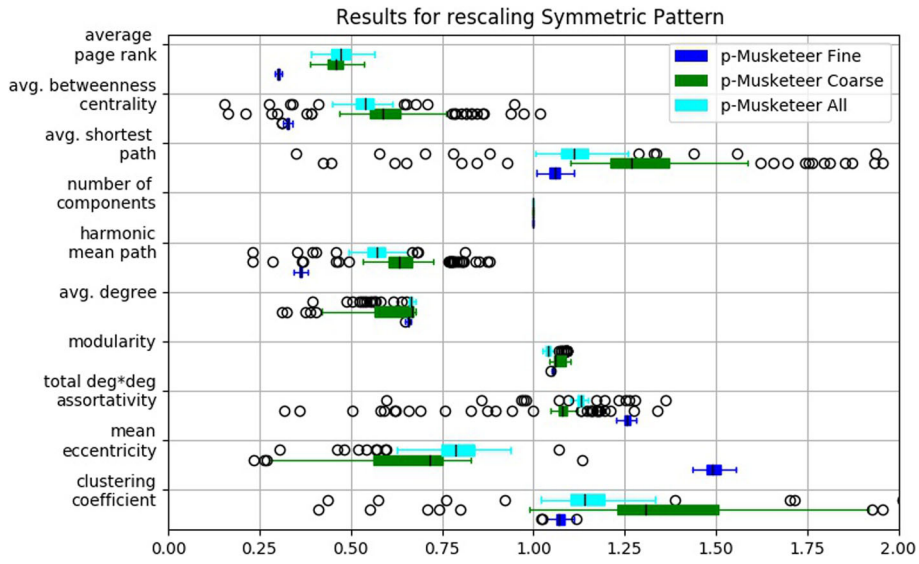

Fig. 21 Computational results on performance of planar Musketeer on symmetric pattern with 1141 nodes and 3162 edges to at least 2200 nodes and 6000 edges

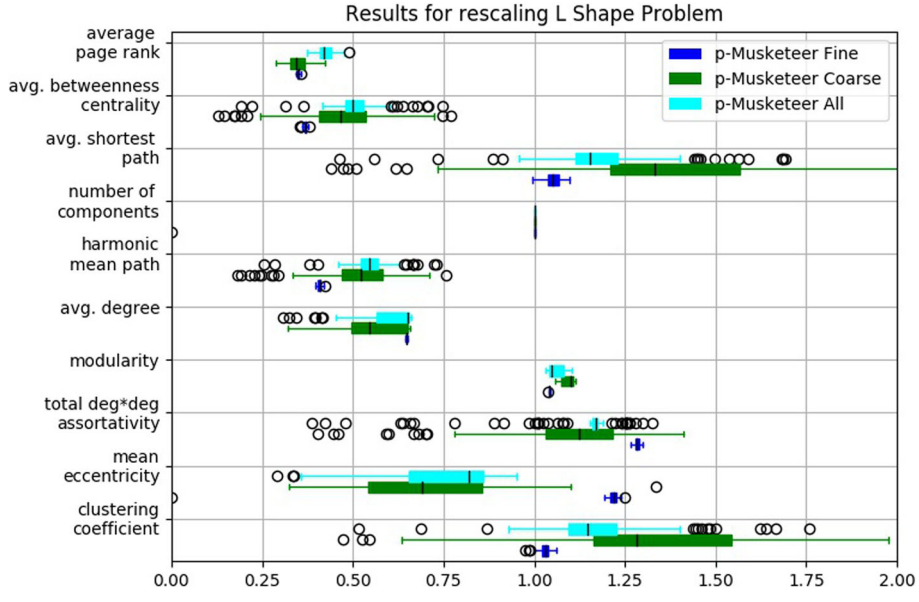

Fig. 22 Computational results on performance of planar Musketeer on graph for thermal L-Shape Problem with 3025 nodes and 8904 edges to at least 9000 nodes and 12000 edges

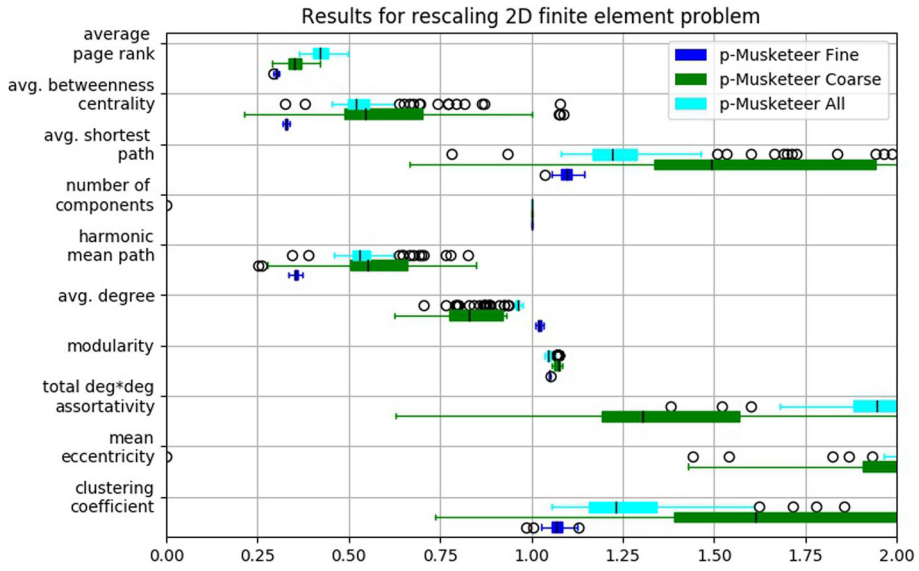

Fig. 23 Computational results on performance of planar Musketeer 2D finite element graph with 1866 nodes and 3538 edges rescaled to at least 4000 nodes and 7000 edges

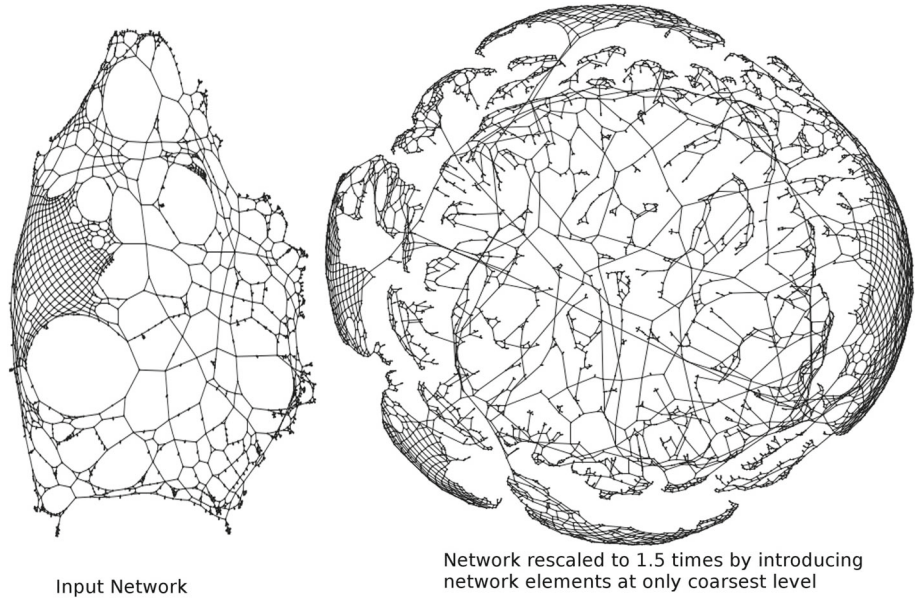

Fig. 24 Visualization of a road network from roadNet-TX with 2001 nodes and 2957 edges rescaled to at least 5028 nodes and 7375 edges

Acknowledgements
This material is based upon work supported by the National Science Foundation under grants #1522751 and #1745300.

Funding
This material is based upon work supported by the National Science Foundation under grants #1522751 and #1745300.

Availability of data and materials
All datasets and algorithm implementation presented in this work are available at https://bit.ly/2CjOUAS

Authors' contributions
We use the following notation for different types of contribution: AD - algorithm design and discussions, I - implementation, P - paper writing, E - experimental evaluation. The authors contributed as follows: VC (AD, I, P, E), AG (AD, P), IS (AD, I, P, E). All authors read and approved the final manuscript.

Competing interests
The authors declare that they have no competing interests.

Publisher's Note

Author details
[1]School of Computing, Clemson University, Clemson, SC, USA. [2]Loyola University Medical Center, Maywood, IL, USA.

Received: 3 July 2018 Accepted: 10 May 2019
Published online: 16 July 2019

References

Aiello W, Chung F, Lu L (2000) A random graph model for massive graphs. In: Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing. ACM. pp 171–180
Aiello W, Chung F, Lu L (2001) A random graph model for power law graphs. Exp Math 10(1):53–66
Barthélemy M (2011) Spatial networks. Phys Rep 499(1-3):1–101
Brandt A, Ron D (2003) Chapter 1 : Multigrid solvers and multilevel optimization strategies. In: Cong J, Shinnerl JR (eds). Multilevel Optimization and VLSICAD. Springer
Brinkmann G. (2011) Program fullgen-a program for generating nonisomorphic fullerenes. see http://cs.anu.edu.au/bdm/plantri
Brinkmann G, McKay BD, et al (2007) Fast generation of planar graphs. MATCH Commun Math Comput Chem 58(2):323–357
Buluç A, Meyerhenke H, Safro I, Sanders P, Schulz C (2016) Recent advances in graph partitioning. In: Algorithm Engineering: Selected Results and Surveys. Springer. pp 117–158
Chakrabarti D, Zhan Y, Faloutsos C (2004) R-mat: A recursive model for graph mining. In: Proceedings of the 2004 SIAM International Conference on Data Mining. Springer. pp 442–446
Chen J, Safro I (2011) Algebraic distance on graphs. SIAM J Sci Comput 33(6):3468–3490
Chimani M, Gutwenger C, Jünger M, Klau GW, Klein K, Mutzel P (2013) The open graph drawing framework (ogdf). Handb Graph Drawing Vis 2011:543–569
Cura R, Perret J, Paparoditis N (2015) Streetgen: In-base procedural-based road generation. ISPRS Ann Photogramm Remote Sens Spat Inf Sci 2:409
Davis T (1997) University of Florida Sparse Matrix Collection. NA Dig 97(23)
Denise A, Vasconcellos M, Welsh DJ (1996) The random planar graph. Congressus numerantium:61–80
Erdös P, Rényi A (1959) On random graphs, I. Publ Math (Debrecen) 6:290–297
Fronczak P, Fronczak A, Bujok M (2013) Exponential random graph models for networks with community structure. Phys Rev E 88(3):032810
Gerke S, McDiarmid C (2004) On the number of edges in random planar graphs. Comb Probab Comput 13(2):165–183
Gilbert EN (1961) Random plane networks. J Soc Ind Appl Math 9(4):533–543
Gutfraind A, Meyers LA, Safro I (2012) Multiscale network generation. CoRR abs/1207.4266. 1207.4266
Gutfraind A, Safro I, Meyers LA (2015) Multiscale network generation. In: 18th IEEE International Conference on Information Fusion (Fusion). Springer. pp 158–165
Hager WW, Hungerford JT, Safro I (2018) A multilevel bilinear programming algorithm for the vertex separator problem. Comput Optim Appl 69(1):189–223
Hunter DR, Handcock MS, Butts CT, Goodreau SM, Morris M (2008) ergm: A package to fit, simulate and diagnose exponential-family models for networks. J Stat Softw 24(3):54860
John E, Safro I (2016) Single-and multi-level network sparsification by algebraic distance. J Complex Netw 5(3):352–388
Karrer B, Newman ME (2011) Stochastic blockmodels and community structure in networks. Phys Rev E 83(1):016107
Leskovec J, Krevl A (2014) SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data
Leskovec J, Lang KJ, Dasgupta A, Mahoney MW (2009) Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. Internet Math 6(1):29–123
Mahdian M, Xu Y (2007) Stochastic kronecker graphs. In: International Workshop on Algorithms and Models for the Web-Graph. Springer. pp 179–186
McDiarmid C, Steger A, Welsh DJ (2005) Random planar graphs. J Comb Theory Ser B 93(2):187–205
Meinert S, Wagner D (2011) An experimental study on generating planar graphs. In: Frontiers in Algorithmics and Algorithmic Aspects in Information and Management. Springer. pp 375–387

Muranho J, Ferreira A, Sousa J, Gomes A, Marques AS (2012) Waternetgen: an epanet extension for automatic water distribution network models generation and pipe sizing. Water Sci Technol Water Supply 12(1):117–123
Newman M (2010) Networks: An Introduction. Oxford University Press, Inc., New York
Newman M (2018) Networks. Springer
Ostfeld A, Uber JG, Salomons E, Berry JW, Hart WE, Phillips CA, Watson J-P, Dorini G, Jonkergouw P, Kapelan Z, et al (2008) The battle of the water sensor networks (bwsn): A design challenge for engineers and algorithms. J Water Resour Plan Manag 134(6):556–568
Palla G, Lovász L, Vicsek T (2010) Multifractal network generator. Proc Natl Acad Sci 107(17):7640
Rao AR, Jana R, Bandyopadhyay S (1996) A markov chain monte carlo method for generating random (0, 1)-matrices with given marginals. Sankhyā: Indian J Stat Ser A:225–242
Ron D, Safro I, Brandt A (2011) Relaxation-based coarsening and multiscale graph organization. Multiscale Modeling Simul 9(1):407–423
Rossman LA (1994) EPANET Users Manual, Cincinnati, OH: US Environmental Protection Agency
Ruppert J (1995) A delaunay refinement algorithm for quality 2-dimensional mesh generation. J Algoritm 18(3):548–585
Safro I, Temkin B (2011) Multiscale approach for the network compression-friendly ordering. J Discret Algorithm 9(2):190–202
Safro I, Ron D, Brandt A (2006) Graph minimum linear arrangement by multilevel weighted edge contractions. J Algorithm 60(1):24–41
Safro I, Ron D, Brandt A (2008) Multilevel algorithms for linear ordering problems. ACM J Exp Algorithmic 13:4
Safro I, Sanders P, Schulz C (2015) Advanced coarsening schemes for graph partitioning. ACM J Exp Algorithmics (JEA) 19:2–2
Seshadhri C, Kolda TG, Pinar A (2012) Community structure and scale-free collections of erdős-rényi graphs. Phys Rev E 85(5):056109
Shewchuk JR (1996) Triangle: Engineering a 2d quality mesh generator and delaunay triangulator. In: Applied Computational Geometry Towards Geometric Engineering. Springer. pp 203–222
Sitzenfrei R, Möderl M, Rauch W (2013) Automatic generation of water distribution systems based on gis data. Environ Model Softw 47:138–147
Staudt C, Sazonovs A, Meyerhenke H (2014) Networkit: An interactive tool suite for high-performance network analysis. CoRR, abs/1403.3005:41
Staudt CL, Hamann M, Safro I, Gutfraind A, Meyerhenke H (2016) Generating scaled replicas of real-world complex networks. In: International Workshop on Complex Networks and Their Applications. Springer. pp 17–28
Staudt CL, Hamann M, Gutfraind A, Safro I, Meyerhenke H (2017) Generating realistic scaled complex networks. Appl Netw Sci 2(1):36. https://doi.org/10.1007/s41109-017-0054-z
Tabourier L, Roth C, Cointet J-P (2011) Generating constrained random graphs using multiple edge switches. J Exp Algorithmics 16:1–71117115. https://doi.org/10.1145/1963190.2063515
Thomassen C (1981) Kuratowski's theorem. J Graph Theory 5(3):225–241
Tutte WT (1963) A census of planar maps. Canad J Math 15(2):249–271
van Lidth de Jeude J, Di Clemente R, Caldarelli G, Saracco F, Squartini T (2019) Reconstructing mesoscale network structures. Complexity 2019:4
Wang Z, Thomas RJ, Scaglione A (2008) Generating random topology power grids. In: Hawaii International Conference on System Sciences, Proceedings of the 41st Annual. Springer. pp 183–183