Information Systems (accepted), 2008

Node and Edge Selectivity Estimation for Range Queries in Spatial Networks

E. Tiakas A.N. Papadopoulos A. Nanopoulos Y. Manolopoulos
Department of Informatics, Aristotle University
54124 Thessaloniki, Greece
{tiakas,papadopo,ananopou,manolopo}@csd.auth.gr

Abstract

Modern applications requiring spatial network processing pose several interesting query optimization challenges. Spatial networks are usually represented as graphs, and therefore, queries involving a spatial network can be executed by using the corresponding graph representation. This means that the cost for executing a query is determined by graph properties such as the graph order and size (i.e., number of nodes and edges) and other graph parameters. In this paper, we present novel methods to estimate the number of nodes and edges in regions of interest in spatial networks, towards predicting the space and time requirements for range queries. The methods are evaluated by using real-life and synthetic data sets. Experimental results show that the number of nodes and edges can be estimated efficiently and accurately, with relatively small space requirements, thus providing useful information to the query optimizer.

Keywords: *spatial networks, selectivity estimation, query optimization*

1 Introduction

Spatial networks can be represented as graphs, where road segments are represented by graph edges and crossroads (and other points of interest) are represented by graph vertices. Depending on the application, such a graph may be weighted, directed or undirected. This way, any spatial query on the original network can be executed on the underlying graph G. Evidently, the performance of such queries depends on the number of nodes and edges found in the region of interest, which defines a subgraph of G, as well as on the number of objects lying on the edges.

Several query processing techniques have been proposed for fundamental query types in spatial networks, such as range and k-nearest-neighbors [11, 15, 17]. However, when such queries are combined, appropriate query optimization techniques are necessary to increase efficiency. Therefore, estimations on factors affecting the performance of such queries are crucial for query optimization purposes. More specifically, the number of vertices and edges contained in a specific region, is an indication of the required computational time required to store the corresponding subgraph, as well as the time required to execute queries.

This paper is a first effort towards a comprehensive study in estimating the number of vertices and edges contained in a region of a spatial network. This region of interest is defined by a starting vertex v_0 and a network-based distance e. The goal is to estimate the number of nodes and edges contained in the region of interest as accurately as possible. Such an estimation is useful in several cases such as:

- In location-based services, it is important to predict the trajectory of a moving object [4], taking into account that the motion is not predefined (e.g., a bus). The estimated number of junctions and street segments that a vehicle may visit provides significant help towards this direction, since they provide an indication regarding the size of uncertain region.

- By using the estimated number of nodes and edges combined with information regarding the current positions of moving objects one can estimate the number of objects lying in a particular distance from a query object. This information can be used for query optimization purposes in spatiotemporal query processing, since it may be used for estimating the selectivity of range and k-NN queries as well as estimating the I/O cost of the query.

- In some spatiotemporal applications, there is a need for continuous evaluation of queries. As an example, consider a moving object for which we need to continuously monitor the set of objects residing within a specific distance from the query object. Our methods may be applied in such cases towards estimating the size of the result for each timestamp.

In this respect, several different directions are examined, each one with different requirements and estimation accuracy. The examined methods are: (i) a simple method based on multidimensional scaling, (ii) an estimation with global parameters, (iii) a local estimation using densities or kernels, and (iv) an estimation with binary encoding techniques.

The rest of this paper is organized as follows. In Section 2, we present the related work and specify our contributions. In Section 3, we formulate the problem and present the estimation methods in detail. Section 4 contains the performance evaluation and related experimental results. Section 5 presents a discussion for applications of the proposed methods, whereas Section 6 concludes our work.

2 Related Work and Contribution

Selectivity estimation has been examined in the past with respect to spatial or spatiotemporal queries. Below, we briefly present some fundamental work in the area.

In [26], the authors examine the performance of range queries in R-trees and variants. More specifically, estimation formulae have been proposed for the number of disk accesses using global parameters and local densities. Selectivity estimation for spatial joins has been studied in [5], where the authors provide efficient methods with relative errors below 30%. In [9] the authors propose two approaches for the selectivity estimation of spatiotemporal queries: a simple histogram approach and an index-based estimator. The Power-Method [25] provides accurate estimations for such queries by using a simple formula with minimal computational cost, small space requirements and average relative error rate below 20%. In [1] the authors propose a selectivity estimation method with low relative estimation error (about 10%) for spatial queries using specific global parameters formulae based on Hausdorff fractal dimension.

The concept of local density has been studied extensively in general and spatial data sets, but not in combination with spatial networks. In the bibliography, one can find a plethora of density estimation proposals. In this direction, an important method is the Kernel Density Estimation Method and its variants [10, 24]. Kernel Density Estimators are used in many application domains such as clustering [13, 18], outlier detection [23], and visualization [19]. Moreover, several variations of kernel density estimations and smoothing have been proposed in [12, 30].

The basic limitation of the previous approaches is that they are restricted to Euclidean spaces only. Our contribution is the presentation of efficient methods for spatial query estimation for

spatial networks assuming non-Euclidean spaces. More specifically, we introduce and evaluate three novel estimation methods:

Global Parameters Estimation Method: which is based on global parameters (see Section 3.3).

Local Densities Estimation Method: which extends the previous method by using local density factors (see Section 3.4). We present a new computational model of local node densities in Section 3.4.1, as well as an alternative approach by applying the well-known Gaussian Kernel Densities Estimators in Section 3.4.2.

Binary Encoding Estimation Method: which uses specific graph transformations, a specific binary encoding technique and a formula with only binary and basic register operations for calculations (Section 3.5).

In addition to the these three methods, a simple solution based on multidimensional scaling is also evaluated. The advantage of this approach is that it can exploit previous results for the selectivity estimation of multidimensional objects using the Euclidean distance.

A preliminary version of this work appears in [28] where we have presented the basic estimation methods. The current version is more complete and in summary the new material is described as follows:

- the multidimensional scaling method is included for comparison purposes,
- a more thorough theoretical analysis is performed and proofs of fundamental theoretical results are given,
- estimation of the number of edges contained in the region of interest is given (in addition to the estimation of the number of nodes),
- a discussion regarding the exploitation of the results by the query optimizer for range and k-NN processing is included, studying selectivity estimation issues and processing cost with respect to the number of I/O operations required,
- a more thorough experimental evaluation is carried out.

3 Estimation Approaches

In this section, we define the problem and present the proposed methods aiming at effective estimation solutions. Table 1 contains the basic symbols used.

3.1 Problem Definition

Let $G(V_G, E_G)$ be a connected weighted graph where V_G and E_G is the set of nodes and edges respectively. The distance measure $d(v, u)$ denotes the shortest path distance between nodes v and u. Given a specific starting node $v_0 \in V_G$, and a desired distance e, we are interested in determining two estimators, $\widetilde{N}(v_0, e)$ and $\widetilde{E}(v_0, e)$, for the total number of nodes and edges that can be reached from v_0 within a distance less than or equal to e. The starting vertex v_0 and the distance e define the region of interest. It is assumed that an edge is counted if and only if it is fully contained in the region of interest. The estimators are formally defined as follows:

Symbol	Description
G	an undirected graph
V_G	set of nodes of graph G
E_G	set of edges of graph G
$\|V_G\|$	number of nodes of graph G
$\|E_G\|$	number of edges of graph G
v_0	selected starting node
e	query distance
$d(v_i, v_j)$	network distance between nodes v_i, v_j
D_G	diameter of G, $D_G = max(d(v_i, v_j), \forall v_i, v_j \in V_G))$
$deg(v_i)$	degree of node v_i
$w(v_i, v_j)$	weight of the edge connecting nodes v_i and v_j
$\overline{w}$	average edge weight
$\overline{deg}$	average node degree
$N(v_0, e)$	exact number of nodes within distance e from v_0
$E(v_0, e)$	exact number of edges within distance e from v_0
$\widetilde{N}(v_0, e)$	estimation of $N(v_0, e)$
$\widetilde{E}(v_0, e)$	estimation of $E(v_0, e)$

Table 1: Frequently used symbols.

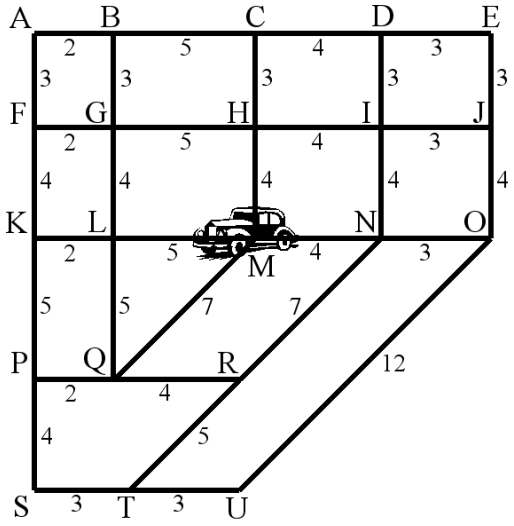

Figure 1: A simple road network example.

$$N(v_0, e) = |\{v \in V_G : d(v, v_0) \leq e\}|$$

$$E(v_0, e) = |\{(v_i, v_j) \in E_G : \min(d(v_i, v_0), d(v_j, v_0)) + w(v_i, v_j) \leq e\}|$$

Figure 1 depicts an example of a spatial network represented by a weighted graph. The starting position of the car is node $v_0 = M$. Table 2 presents the number of reachable nodes and edges from node M within the distances $e = 0, 1, 2, ..., 10$. We observe that as the value

e	$N(v_0, e)$	Nodes	$E(v_0, e)$	Edges
0,1,2,3	1	M	0	—
4	3	M,N,H	2	MH,MN
5,6	4	M,N,H,L	3	MH,MN,ML
7	8	M,N,H,L,Q,O,K,C	7	MH,MN,ML,MQ,HC,NO,KL
8	9	M,N,H,L,Q,O,K,C,I	9	MH,MN,ML,MQ,HC,NO,KL,HI,NI
9	11	M,N,H,L,Q,O,K,C,I,G,P	12	MH,MN,ML,MQ,HC,NO,KL,HI,NI,HG,LG,PQ
10	11	M,N,H,L,Q,O,K,C,I,G,P	13	MH,MN,ML,MQ,HC,NO,KL,HI,NI,HG,LG,PQ,LQ

Table 2: Reachable nodes and edges within distance e from node M.

of e increases, the number of nodes and edges contained in the region of interest also increases (as expected). The challenge is to derive estimators $\widetilde{N}(v_0, e)$ and $\widetilde{E}(v_0, e)$ to effectively estimate $N(v_0, e)$ and $E(v_0, e)$ by avoiding the execution of graph processing algorithms (e.g., network expansion or Dijkstra-like processing).

3.2 Estimation Based on Multidimensional Scaling

A simple approach that can be applied is to transform the corresponding graph in the Euclidean space, since several tools have been proposed in the literature for estimating the number of points contained in a region. Thus, as a first approach along this line, we use the Classical Multi-Dimensional Scaling (MDS) [2, 3, 20].

MDS comprises a set of tools that can transform data objects from a non-Euclidean space to an Euclidean space, allowing effective visualization. MDS can also produce representations of data objects in a low dimensional space (dimensionality reduction), which is frequently desirable. Due to dimension reduction, approximation quality may be reduced, and there are several adjustable proposed functions [2, 14, 31, 29], yielding the optimal arrangement towards minimum loss of distance preservation (loss functions).

The raw data of an MDS analysis are typically the dissimilarity values between the objects in the original space, which have been calculated by using a specific distance measure. The result of an MDS analysis is a spatial configuration in which all objects are represented as points into the n-dimensional Euclidean space ($\Re^n$), which are arranged in such a way, that their Euclidean distances correspond to the original dissimilarities and approximate their distance values in the original space.

Let p denote the number of objects for which the pairwise dissimilarities are available in the $p \times p$ matrix P. The result of MDS is contained in a coordinate matrix S of size $p \times n$ ($n < p$), which represents the corresponding points into the Euclidean space ($\Re^n$). The number of dimensions (n) in the final Euclidean space is defined by the positive eigenvalues of a produced matrix. Therefore, n is a specific constant number that defines the maximum number of required dimensions, which can reproduce the original space without significant loss of quality. The following steps involve some linear algebra and summarize the algorithm of classical MDS (more details can be found in [2]):

- Compute the matrix of squared proximities $P^{(2)} = [x_{ij}^2]$.
- Compute the matrix $B = -\frac{1}{2} J P^{(2)} J$ using the matrix $J = I - \frac{1}{p} U$, where I is the unitary matrix, p is the number of objects and U is the matrix which has all of its values equal to 1. This step is also called *double centering*.

Algorithm MDS-CreateDensityMatrix(G, V_G, k, c)

1. Apply the classical MDS transformation on G: $G \xrightarrow{T_{MDS}} S \subseteq \Re^n$.
2. Select k (where $2 \leq k \leq n$) from the n returned dimensions of S, based on the k largest eigenvalues.
3. Construct the estimation subset $\widetilde{S} \subseteq S$ on $\Re^k$, with the selected dimensions.
4. Construct the hyper-rectangle H that encloses $\widetilde{S}$ using the global minimum and maximum coordinates: $minx_i = \min\{vx_i, v \in V_G\}$ and $maxx_i = \max\{vx_i, v \in V_G\}$, where $i = 1, ..., k$.
5. Apply the hyper-grid C on the hyper-rectangle H.
6. Compute the grid cell sizes in each dimension: $dx_i = \frac{maxx_i - minx_i}{c}$, for c a desired number.
7. For each cell $C(i_1, ..., i_k)$, where $1 \leq i_1, ..., i_k \leq c$, store the number of contained points in $CD(i_1, ..., i_k)$.

Figure 2: Outline of MDS preprocessing algorithm.

- Extract the n largest positive eigenvalues $\lambda_1, ..., \lambda_n$ of B and the corresponding n eigenvectors $e_1, ..., e_n$.
- Compute and return the final coordinate matrix $S = E_n L_n^{(1/2)}$, where E_n is the matrix of the n eigenvectors and $L_n^{(1/2)}$ is the diagonal matrix of the square roots of the n eigenvalues of B, respectively.

Often, it is desirable to select a smaller number of dimensions $k < n$. Apparently, the fewer dimensions we select during dimensionality reduction, the less the distance preservation achieved. The set of dimensions is always selected according to the largest eigenvalues.

3.2.1 The MDS-based Method

In MDS-Grid method we apply the classical MDS transformation, where the raw data P are arranged by the coordinates of the nodes of the original network G. On the resulted coordinate matrix S, which has p points with n-coordinates, we perform dimensional reduction by taking the first $k < n$ coordinates (which correspond to the k-largest eigenvalues) into a new matrix $\widetilde{S}$. In the sequel, by T_{MDS} we denote the classical MDS transformation: $G \xrightarrow{T_{MDS}} S \subseteq \Re^n$.

After building the final point-set $\widetilde{S}$ in the Euclidean space $\Re^k$ with the appropriate number of dimensions k, we can apply any method for the Euclidean e-range estimations on $\widetilde{S}$. Here, we follow a simple histogram-based technique and apply a k-dimensional hyper-grid C, splitting the space S into c^k equal-sized hyper-rectangle cells $C(i_1, i_2, ..., i_k)$ (where $1 \leq i_1, i_2, ..., i_k \leq c$, where c is a desired number of cells in each dimension), and computing their corresponding densities by counting all contained node-points. Then, we keep all these cell densities in a k-dimensional matrix CD to subsequently compute the total number of existing nodes in the desired e-range regions. The preprocessing algorithm for building the cell densities matrix is depicted in Figure 2.

After the construction of the CD matrix, we can estimate the number of nodes $N(v_0, e)$ in an e-range region of a node v_0 of G, using the densities of the cells lying in the hyper-spherical e-range region of the point $v_0' = T_{MDS}(v_0)$ of $\widetilde{S}$ (see Figure 3). For this estimation, we apply the algorithm of Figure 4. Note that the under- and over-estimation steps scan only the cells of the hyper-grid H, where the minimum bounded hyper-rectangle (MBR) of the hyper-sphere A'

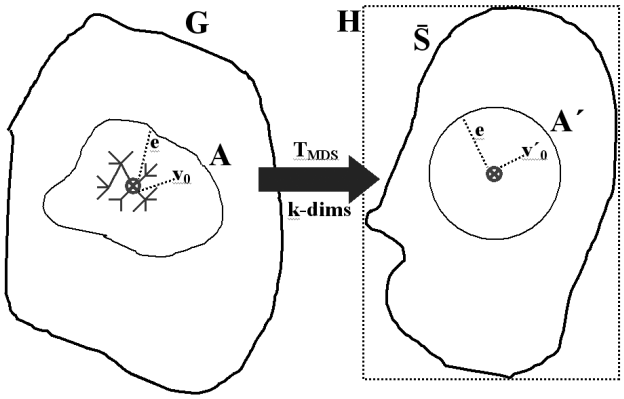

Figure 3: Estimation with the MDS-Grid method.

lies. Based on the algorithm of Figure 4, the node estimation formula has as follows:

$$\boxed{\widetilde{N}(v_0, e) = \frac{1}{2} \cdot \left(\sum_{C(i_1,i_2,...,i_k) \subset A'} CD(i_1, i_2, ..., i_k) + \sum_{C(i_1,i_2,...,i_k) \cap A' \neq \emptyset} CD(i_1, i_2, ..., i_k) \right)} \quad (1)$$

or equivalently:

$$\boxed{\widetilde{N}(v_0, e) = \sum_{C(i_1,i_2,...,i_k) \subset A'} CD(i_1, i_2, ..., i_k) + \frac{1}{2} \cdot \sum_{\substack{C(i_1,i_2,...,i_k) \cap A' \neq \emptyset \\ \wedge C(i_1,i_2,...,i_k) \not\subseteq A'}} CD(i_1, i_2, ..., i_k)} \quad (2)$$

Algorithm MDS-NodesEstimation(v_0, e)

---

1. Find the corresponding node-point of v_0 on $\widetilde{S}$: $v'_0 = T_{MDS}(v_0)$.
2. Define the hyper-sphere A' with center v'_0 and radius e (Figure 3).
3. Estimate the number of node-points $N(v'_0, e)$ in A' as follows:
4. (a) **underestimate** $N(v'_0, e)$ by summing the densities of the cells that lie in A': $\hat{N}(v'_0, e)$.
5. (b) **overestimate** $N(v'_0, e)$ by summing the densities of the cells that lie in or intersect A': $\check{N}(v'_0, e)$.
6. (c) compute the average value: $\widetilde{N}(v'_0, e) = \frac{\hat{N}(v'_0,e) + \check{N}(v'_0,e)}{2}$.
7. Return $\widetilde{N}(v'_0, e)$

Figure 4: Outline of MDS node estimation algorithm.

In Equation 2, the node estimation is performed by summing all cell densities of the cells that are included into the volume of the hyper-sphere A', and by summing the half of the cell densities of all the cells that intersect A'. Therefore, the intersection parts are counted as the 50% of the cells. If we calculate the intersection volume and we include their percentages into Equation 2, we could perform a better estimation. However, in that case, the calculation of all intersection parts will significantly increase the computational time of the algorithm in Figure 4, as the cost of this calculation is more expensive.

3.2.2 Time, Space and Preprocessing Requirements

The storage requirements of the MDS-Grid method comprise the required space for (a) the final k coordinates of all nodes after the MDS transformation, plus (b) the cell densities matrix.

Apparently, the required space is $O\left(k \cdot |V_G| + c^k\right)$. This amount depends on the values of k and c, which can be modified accordingly. All these data can be kept in main memory.

The required time for the computation of the estimator $\widetilde{N}(v_0, e)$ is the time for scanning the cells of the hyper-grid H, in which the minimum bounded hyper-rectangle (MBR) of the hyper-sphere A' lies. This time equals: $\lceil \frac{2e}{dx_1} \rceil \cdot \lceil \frac{2e}{dx_2} \rceil \cdot ... \cdot \lceil \frac{2e}{dx_k} \rceil$ according to Formulae (1-2). Therefore, the total time cost for the estimation equals to: $O\left(\prod_{i=1,...,k} \left\lceil \frac{2e}{dx_i} \right\rceil\right)$. A significant observation is that the estimation time depends on the selection of the distance e and the parameters c and k (since these parameters affect the cell dimensions dx_i).

The total preprocessing cost includes:

- the time for the MDS transformation, which is an $O(|V_G|^3)$ time, since MDS performs array multiplications of size $O(|V_G|^2)$ on the dissimilarity matrix of all nodes, as well as other less costly operations (e.g., finding eigenvalues, etc.), plus
- the time for the construction of $\widetilde{S}$ by selecting the k coordinates, which is an $O(k \cdot |V_G|)$ time, plus
- the time for the construction of the hyper-grid H by finding the global minimum and maximum coordinates, which is an $O(k \cdot |V_G|)$ time, plus
- the time for the initialization and construction of the cell densities matrix, which is an $O(c^k)$ time, plus finally
- the time to compute the cell densities by scanning all node-points on H, which is an $O(|V_G|)$ time.

Summing up, we have an $O(|V_G|^3 + k \cdot |V_G| + k \cdot |V_G| + c^k + |V_G|) = O(|V_G|^3 + c^k)$ preprocessing cost.

3.2.3 Disadvantages of the MDS-Grid Based Approach

The MDS-Grid method has several disadvantages, which can turn this approach inapplicable for estimation purposes in a spatial network:

- it can be used only for node estimations but not for edge estimations, due to the nature of the transformation process.
- it can not be used for very large graphs due to the high time and space and preprocessing requirements: $O(|V_G|^3 + c^k)$ time and $O(|V_G|^2)$ space (the dissimilarity matrix and all intermediate matrices need quadratic space).
- it requires a significant amount of space: $O\left(k \cdot |V_G| + c^k\right)$, i.e., the required space grows exponentially by increasing the number of dimensions k.
- the estimation time grows exponentially as this radius grows. Moreover, by increasing the number of dimensions k or the number of grid cells c, the estimation time grows significantly for large values of e.

Based on the previous observations, the MDS-Grid method may fail for the node estimation problem, as there is a necessity for good estimation accuracy with low time and space requirements. Evidently, there is a need to develop estimation methods with the following desirable properties:

- support of both node and edge estimations,
- reduced estimation time,
- reduced space and preprocessing requirements, and
- satisfactory estimation accuracy.

3.3 Global Parameters Estimation Method

In this method, the estimation formulae $\widetilde{N}(v_0, e)$ and $\widetilde{E}(v_0, e)$ are based on two global graph parameters: the average edge weight $\overline{w}$ and the average node degree $\overline{deg}$. These parameters are formally defined as follows:

$$\overline{w} = \frac{1}{|E_G|} \sum w(v_i, v_j)$$

$$\overline{deg} = \frac{1}{|V_G|} \sum_i deg(v_i) = \frac{2|E_G|}{|V_G|}$$

where $w(v_i, v_j)$ is the weight associated to the edge connecting nodes v_i and v_j and $deg(v_i)$ is the degree of node v_i.

The Global Parameters Estimation method is based on simple formulae for the estimated number of nodes and edges and requires only the global parameters $\overline{w}$ and $\overline{deg}$:

$$\boxed{\widetilde{N}(v_0, e) = \widetilde{N}(e) = \frac{\overline{deg}}{2} \cdot \frac{e}{\overline{w}} \cdot \left(\frac{e}{\overline{w}} + 1\right) + 1} \tag{3}$$

$$\boxed{\widetilde{E}(v_0, e) = \widetilde{E}(e) = \overline{deg} \cdot \frac{e^2}{\overline{w}^2}} \tag{4}$$

3.3.1 Derivation of the Proposed Formulae

In order for the Global method to be effective regarding estimation accuracy, it should be applied in special types of graphs, obeying some specific properties. These graphs are termed *regular uniform* graphs and they are formally defined as follows:

Definition 3.1. *A connected graph G is called regular uniform (RU graph) if the following properties hold: (i) all nodes have the same degree, (ii) all edge weights are identical, and (iii) If we select a random node v and gradually increase the distance around v by multiples of the constant edge weight, the number of new reachable nodes increases by following an arithmetic growth.*

An infinite RU graph is an RU graph with infinite number of nodes and edges. Although infinite graphs do not appear in real-life applications, they are used as mathematical tools to investigate some properties of the Global estimation method. Using the previous definition, an RU graph can be constructed by the following process:

- We add a starting central node v_c.
- We add n new nodes around v_c, where n is equal to the constant node degree of the RU graph, and we connect these nodes with v_c. These nodes are the level-1 nodes.

- We add $2 \cdot n$ new nodes (the level-2 nodes) around the level-1 nodes and we connect them with the level-1 nodes, such that all level-1 nodes to have degree n ("completed" nodes). We can also make connections between nodes of the same level.
- We repeat the previous step to construct the next level nodes (level-3 has $3 \cdot n$ nodes, level-4 has $4 \cdot n$ nodes etc.)
- We continue to infinity to create an *infinite* RU graph or we can stop at any level to create a *finite* RU graph.
- All generated edges are assigned the same weight.

Evidently, in a finite RU graph, the degree of the last level nodes may be less than n. Figure 5 presents the level-3 expanded finite RU graphs varying the node degrees from 2 to 10 using the above methodology. On several real spatial networks there are many graph parts that are isomorphic to finite RU graphs (expanded at low levels), and this the reason we have chosen this graph family for our estimations. For example, many real city road networks have a lot of square blocks, thus there are many graph parts that are isomorphic to "square-like" parts (upper-right graph of Figure 5).

Lemma 3.1. *Let H_∞ be an infinite RU graph, where all node degrees are equal to a positive integer number deg and all edge weights are equal to a positive real number w. Then, for any random selected node $v_0 \in H_\infty$, the exact number of nodes $N(v_0, e)$ lying in the range region of v_0 within a network distance $e \geq 0$ (which is a multiple of w), is:*

$$N(v_0, e) = N(e) = \frac{deg}{2} \cdot \frac{e}{w} \cdot \left(\frac{e}{w} + 1\right) + 1$$

Proof. Since H_∞ is an infinite RU graph, the exact number of nodes $N(v_0, e)$ does not depend on the selection of v_0, and depends only on the distance e, thus: $N(v_0, e) = N(e)$.

Since the distance e is a multiple of w, we have: $e = k \cdot w$ where k is a positive integer number or zero. Following the above mentioned construction procedure and using the property of the arithmetical progress increase (which is based on), we have for the number of nodes $N(e)$ the following results:

- for $e = 0w = 0$: $N(e) = 1$ (the node v_0 only).
- for $e = 1 \cdot w = w$: $N(e) = 1 + deg$ (the node v_0 plus the deg nodes of Level-1).
- for $e = 2 \cdot w$: $N(e) = 1 + deg + 2 \cdot deg$ (the node v_0 plus the deg nodes of Level-1 and the $2 \cdot deg$ nodes of level-2), etc.

Therefore, using induction we derive the following result for $e = kw$:

$$N(e) = 1 + deg + 2 \cdot deg + 3 \cdot deg + ... + k \cdot deg$$

$$= deg \cdot (1 + 2 + 3 + ... + k) + 1 = deg \cdot \frac{k(k+1)}{2} + 1 = \frac{deg}{2} \cdot k(k+1) + 1$$

As $k = \frac{e}{w}$, we finally have:

$$N(e) = \frac{deg}{2} \cdot \frac{e}{w} \cdot \left(\frac{e}{w} + 1\right) + 1$$

□

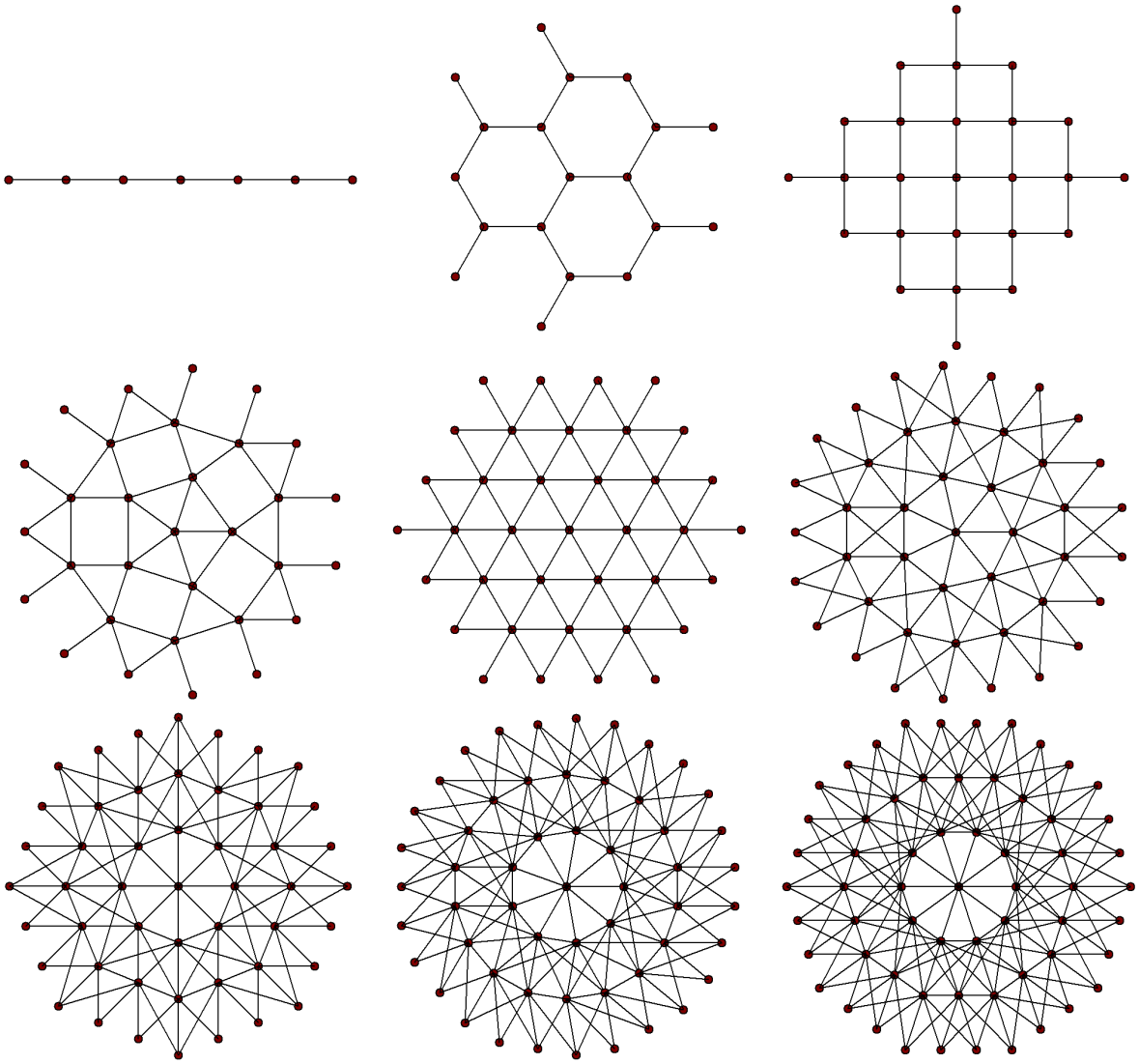

Figure 5: RU graphs with degrees 2-10, expanded to level-3.

Regarding edge estimation, again the number of edges contained in the region of interest does not depend on the starting vertex v_0, and depends only on the selected distance e. Therefore, $E(v_0, e) = E(e)$. The problem with edge estimation is that the number of estimated edges coincides with the true number of edges in the region of interest only when $deg = 4$. In other words, the arithmetic growth in the number of edges is not generally satisfied. More specifically, for $e = 0 \cdot w, 1 \cdot w, 2 \cdot w, 3 \cdot w..., k \cdot w$ we have $0, deg, 3 \cdot deg, 5 \cdot deg, ..., (2k-1) \cdot deg$ new edges respectively from each level, thus for $e = kw$ we have:

$$E(e) = 0 + deg + 3 \cdot deg + 5 \cdot deg + ... + (2k-1)deg =$$

$$= deg \cdot [1 + 3 + 5 + ... + (2k-1)] = deg \cdot k^2 = deg \cdot \frac{e^2}{w^2}$$

The above formula can be used for edge estimation as long as the degree of the RU graph is near 4. For arbitrary values the edge estimator will not provide accurate results.

To apply the derived formulae in arbitrary graphs, we must relax two basic assumptions used so far: (i) the infinity assumption, and (ii) the global equality of edge weights and node degrees. The following lemma describes the case of a finite RU graph. As it has been noted before, this type of graph is an RU graph, except from the fact that nodes of the last level does not satisfy the assumption that all node degrees are equal.

Lemma 3.2. *Let H be a finite RU graph with node degrees equal to a positive integer deg (except probably the last level nodes) and edge weights equal to a positive real w. Moreover, let H_C denote the central region of H, which is defined by its central node v_c (the starting central node of its construction procedure) and a network distance equal to $D_H/4$ (where D_H is the diameter of the graph H). Then, for any region R which is defined by a randomly selected starting node v_0 from the region H_C and a network distance $0 \leq e \leq D_H/4$, the numbers of nodes $N(v_0, e)$ and edges $E(v_0, e)$ contained in the region R, are independent of the selection of v_0, but depend only on the selection of e. Thus:*

$$N(v_0, e) = N(e), \quad E(v_0, e) = E(e), \quad \forall v_0 \in H_C, \forall e : 0 \leq e \leq D_H/4$$

Proof. H is a finite RU graph, thus it is isomorphic to a symmetrical graph constructed with the above presented methodology, starting from the central node v_c and expanded till a specific level. Let n be the last expanded level of H which contains its "border" nodes. Then, assuming the network distance d(), all shortest paths from v_c to any node of the last level have length equal to: $R_H = n \cdot w$. This constant distance R_H is the radius of the graph H. Therefore, the diameter D_H of the graph H is the constant distance: $D_H = 2n \cdot w$.

The central region H_C of H contains all reachable nodes and edges from v_c within a network distance: $D_H/4 = \frac{2n \cdot w}{4} = \frac{n}{2} \cdot w$. Therefore it is also a finite RU graph with the same central node v_c and expanded till the $\lfloor \frac{n}{2} \rfloor$ level. Thus, if v_0 is a randomly selected starting node from the region H_C, it lies into a specific expansion level k of H_C with: $0 \leq k \leq \lfloor \frac{n}{2} \rfloor$. Moreover, the network distance between the nodes v_0 and v_c satisfies the inequality:

$$0 \leq d(v_0, v_c) \leq \lfloor \frac{n}{2} \rfloor \cdot w$$

Now, the region R contains all reachable nodes and edges from v_0 within a network distance: $D_H/4 = \frac{n}{2} \cdot w$. Due to the "uniformity" attribute of the whole graph, R is also a finite RU graph with central node v_0 and expanded till its $\lfloor \frac{n}{2} \rfloor$ level. Thus, if v is a randomly selected node from the region R, it must lies into a specific expansion level m of R where: $0 \leq m \leq \lfloor \frac{n}{2} \rfloor$. Moreover, the network distance between the nodes v and v_0 must satisfy the inequality:

$$0 \leq d(v, v_0) \leq \lfloor \frac{n}{2} \rfloor \cdot w$$

Therefore, using the triangular inequality which holds for the network distance metric and the previous inequalities we take:

$$0 \leq d(v_c, v) \leq d(v_c, v_0) + d(v_0, v) \leq \lfloor \frac{n}{2} \rfloor \cdot w + \lfloor \frac{n}{2} \rfloor \cdot w \leq 2 \cdot \frac{n}{2} \cdot w \leq R_H$$

As the node v can be any node of the region R, the whole region R lies into the graph H so it keeps the "uniformity" attribute, thus: $N(v_0, e) = N(e)$, $E(v_0, e) = E(e)$, and the lemma has been proved. □

Therefore, by combining Lemmas 3.1-3.2, for any finite RU graph H with diameter D_H and central region H_C (defined by its central node v_c and a network distance equal to $D_H/4$), the

number of reachable nodes $N(v_0, e)$ and edges $E(v_0, e)$ from a node $v_0 \in H_C$ within a network distance $0 \leq e \leq D_G/4$, remain the same. Thus, the same derived formulae can be used for estimations in finite RU graphs if we follow the mentioned restrictions for the parameters v_0 and e.

In the previous, we have examined the cases of infinite RU graphs and finite RU graphs with equal node degrees and edge weights. Now, if we relax this "regularity" attribute, we can still use the same formulae for estimations, after substituting the constants deg and w with the average degree $\overline{deg}$ and the average weight $\overline{w}$, respectively. Thus, we can use the formulae (3-4) for estimations on general graphs. The usage of these average values into the formulae can lead to efficient estimation results, if the node degrees and edge weights distributions do not significantly deviate from these averages. However, in case of large deviations for a large number of nodes and edges, these formulae may produce significant errors, as it is confirmed in the experimental results.

Definition 3.2. *A connected graph G is called almost regular uniform (ARU graph) if the following conditions are satisfied: (i) all graph edge weights are lying in the interval $[(1-a)\overline{w}, (1+a)\overline{w}]$, where a is a small positive real (e.g., $0 < a < 0.2$), (ii) all graph node degrees are integers in the interval: $[\lfloor\overline{deg}\rfloor - 1, \lceil\overline{deg}\rceil + 1]$, and (iii) the same as in the previous definition.*

Theorem 3.3. *Let G be an almost regular uniform spatial network, and also let:*

1. *the selected starting node v_0 to belong in a region of G defined by its central node v_c (at this point, as central node we define the median node of the shortest path with length D_G, where D_G is the diameter of the graph), and a network distance equal to $\frac{D_G}{4}$, and*
2. *the desired range distance e to lie in the interval $[0, \frac{D_G}{4}]$.*

Then, the estimator for the number of nodes $N(v_0, e)$:

$$\widetilde{N}(v_0, e) = \widetilde{N}(e) = \frac{\overline{deg}}{2} \cdot \frac{e}{\overline{w}} \cdot \left(\frac{e}{\overline{w}} + 1\right) + 1$$

have estimation error $Error[\widetilde{N}]$ which satisfy the following upper bound:

$$Error[\widetilde{N}] \leq \max\left\{ \left| \frac{\lceil\overline{deg}\rceil + 1}{\overline{deg}(1+a)} - 1 \right|, \left| \frac{\lfloor\overline{deg}\rfloor - 1}{\overline{deg}(1-a)} - 1 \right| \right\}$$

In addition, if the average degree $\overline{deg}$ is near to 4 (with the same deviation a of the almost regular uniform definition of G),then the estimator for the number of edges $E(v_0, e)$:

$$\widetilde{E}(v_0, e) = \widetilde{E}(e) = \overline{deg} \cdot \frac{e^2}{\overline{w}^2}$$

have estimation error $Error[\widetilde{E}]$ which satisfy the following upper bound:

$$Error[\widetilde{E}] \leq \max\left\{ \left| \frac{\lceil\overline{deg}\rceil + 1}{\overline{deg}(1+a)^2} - 1 \right|, \left| \frac{\lfloor\overline{deg}\rfloor - 1}{\overline{deg}(1-a)^2} - 1 \right| \right\}$$

The proof is included in the Appendix. From the latter theorem it is observed that these estimation error bounds are independent of the average weight $\overline{w}$ and the selected distance e.

3.3.2 Time, Space and Preprocessing Requirements

A significant advantage of the Global Parameters Estimation method is that there is no need for bookkeeping. Another advantage is that the estimators $\widetilde{N}(v_0, e)$ and $\widetilde{E}(v_0, e)$, can be computed instantly. The preprocessing steps require only the computation of the global parameters: $\overline{deg} = \frac{2|E_G|}{|V_G|}$ and $\overline{w} = \frac{\sum w(v_i,v_j)}{|E_G|}$ which can be computed in $O(V_G + E_G)$ time.

3.4 Local Densities Estimation Method

The next method uses the concept of *local densities* to take into consideration the influence of sparse and dense regions to the estimation process. The local densities are computed in a preprocessing step and maintained in main memory. The more representative local density factors are used, the better estimation results are obtained. In the sequel, we present two methods for the calculation of the local densities: (a) the Local Counting Density Estimators and (b) Kernel Density Estimators.

3.4.1 Local Counting Density Estimators

The main idea behind the Local Counting Density Estimators, is to count the number of nodes and edges on every node $v \in G$, within a small local distance e_c around node v. To derive the new expressions we first need to define the following concepts:
The **Local Node Density** of a node $v \in V_G$ is an integer number defined as follows:

$$LND_v = |\{v_i \in V_G : d(v, v_i) \leq e_c\}|$$

The **Local Edge Density** of a node $v \in V_G$ is an integer number defined as follows:

$$LED_v = |\{(v_i, v_j) \in E_G : \min(d(v, v_i), d(v, v_j)) + w(v_i, v_j) \leq e_c\}|$$

The **Normalized Local Node Density** of a node $v \in V_G$ is a real number in the interval [0,1], defined as follows:

$$NLND_v = \frac{LND_v}{\max\{LND_{v_i}, v_i \in V_G\}} \tag{5}$$

The **Normalized Local Edge Density** of a node $v \in V_G$ is a real number in the interval [0,1], defined as follows:

$$NLED_v = \frac{LED_v}{\max\{LED_{v_i}, v_i \in V_G\}} \tag{6}$$

For a better understanding of these concepts, we present an example based on the spatial network of Figure 1. By setting $e_c = 7$, we count for every node all reachable nodes and edges with shortest path distance less than or equal to 7. Table 3 depicts all calculations. Nodes with normalized local densities close to 1, lie on dense regions, whereas nodes with normalized local densities close to 0, lie in sparse regions. Moreover, the most dense region will have both normalized local densities set to 1.

Using these definitions new estimation formulae can be derived by extending the global parameter estimation formulae with the normalized local node and edge densities:

$$\boxed{\widetilde{N}(v_0, e) = NLND_{v_0} \cdot \left(\frac{\overline{deg}}{2} \cdot \frac{e}{\overline{w}} \cdot \left(\frac{e}{\overline{w}} + 1\right) + 1\right)} \tag{7}$$

$$\boxed{\widetilde{E}(v_0, e) = NLED_{v_0} \cdot \overline{deg} \cdot \frac{e^2}{\overline{w}^2}} \tag{8}$$

v_0	LND_{v_0}	LED_{v_0}	$NLND_{v_0}$	$NLED_{v_0}$
A	6	6	0.75	0.6
B	6	6	0.75	0.6
C	8	8	1	0.8
D	7	8	0.875	0.8
E	6	6	0.75	0.6
F	7	8	0.875	0.8
G	7	8	0.875	0.8
H	8	8	1	0.8
I	8	10	1	1
J	7	8	0.875	0.8
K	8	9	1	0.9
L	8	9	1	0.9
M	8	7	1	0.7
N	7	7	0.875	0.7
O	6	6	0.75	0.6
P	7	7	0.875	0.7
Q	7	7	0.875	0.7
R	5	4	0.625	0.4
S	5	4	0.625	0.4
T	5	4	0.625	0.4
U	3	2	0.375	0.2
MAX	8	10		

Table 3: Local Counting Density Estimators for the graph of Figure 1 with $e_c = 7$.

The usage of normalized local node and edge densities as global factors into Equations 7 and 8, ensures that: (i) the more dense regions we have around v_0, the more similar the results are to those of Global Method, (ii) the more sparse regions we have around v_0, the more the deviation from the values returned by the Global Method. Therefore, we parameterize the influence of sparse and dense regions to the estimation procedure relatively to the their corresponding sparsity or density.

This simple approach offers satisfactory results if the distance e_c is set correctly. We use e_c values that are small multiples of the average weight $\overline{w}$. The accuracy of the above estimators depends on this constant. In particular, there is a multiple k of $\overline{w}$, where the e_c parameter minimizes the estimation error. As e_c takes values smaller than $k \cdot \overline{w}$, the local densities are underestimated and the estimation error increases. On the other hand, as e_c takes values larger than $k \cdot \overline{w}$, the local densities are overestimated and the estimation error again increases. Therefore, we can tune the e_c parameter by means of the k value, to minimize the estimation error.

3.4.2 Kernel Density Estimators

Kernel Density Estimators belong to a class called non-parametric density estimators, because they do not have a fixed functional form with constant global parameters. In particular, they smooth out the contribution of each observed object over a local neighborhood of a specific object.

We apply the Kernel Density Estimators to our problem, where the observed objects are the nodes of the spatial network graph G, and their contribution is their network distance

(i.e., shortest path distance). Any known kernel function $K(x)$ can be used such as: Uniform, Triangle, Epanechnikov, Quartic, Triweight, Cosinus, etc. In Section 4.3 we examine all these functions in node and edge estimations. For example, the normal (Gaussian) kernel has the following form:

$$K(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}$$

The **Local Node Density** of a node $v \in V_G$ is a positive real number defined as follows:

$$LND_v = \sum_{v_i \in V_G} K\left(\frac{d(v, v_i)}{h}\right)$$

The **Local Edge Density** of a node $v \in V_G$ is a positive real number defined as follows:

$$LED_v = \sum_{(v_i, v_j) \in E_G} \left(K\left(\frac{d(v, v_i)}{h}\right) + K\left(\frac{d(v, v_j)}{h}\right)\right)$$

The normalized local node and edge densities are defined as in Equations (5) and (6), whereas the final estimation is performed again by Equations (7) and (8). The parameter $h > 0$ is the bandwidth of the kernel function, which is a constant used for smoothing.

Next, we present an example based on the spatial network of Figure 1. Assuming a bandwidth $h = 7$, Table 4 depicts the results of local densities using the Gaussian kernel. The calculations of LND for node A is based on the calculation of all shortest paths from A which are: {0, 2, 7, 11, 14, 3, 5, 10, 14, 17, 7, 9, 14, 18, 21, 12, 14, 18, 16, 19, 22}, respectively. Thus:

$$LND_A = K\left(\frac{0}{7}\right) + K\left(\frac{2}{7}\right) + K\left(\frac{7}{7}\right) + K\left(\frac{11}{7}\right) + K\left(\frac{14}{7}\right) + \ldots + K\left(\frac{22}{7}\right)$$

$$= \frac{1}{\sqrt{2\pi}}\left(e^{-\frac{1 \cdot 0^2}{2 \cdot 7^2}} + e^{-\frac{1 \cdot 2^2}{2 \cdot 7^2}} + e^{-\frac{1 \cdot 7^2}{2 \cdot 7^2}} + e^{-\frac{1 \cdot 11^2}{2 \cdot 7^2}} + e^{-\frac{1 \cdot 14^2}{2 \cdot 7^2}} + \ldots + e^{-\frac{1 \cdot 22^2}{2 \cdot 7^2}}\right) =$$

$$= \frac{1}{\sqrt{2\pi}}\left(1 + 0.960005441 + 0.60653066 + 0.290923807 + 0.135335283 + \ldots\ldots + 0.007163364\right) =$$

$$= \ldots = 2.778$$

The accuracy of the estimators $\widetilde{N}(v_0, e)$ and $\widetilde{E}(v_0, e)$ depend on h. More specifically, there is a real value h_m where the estimation error is minimized. As h take values smaller than h_m, the distances are elongated into the kernel $\left(\frac{d(v,v_i)}{h}\right)$, thus the local densities are underestimated and the estimation error increases. Also, as h takes values larger than h_m, the distances shrink into the kernel, the local densities are overestimated and, thus, the estimation error again increases. Therefore, we can tune the h parameter to minimize the estimation error.

3.4.3 Time, Space and Preprocessing Requirements

In both variations of the Local Densities estimation method, the required space for all pre-computed data is $O(|V_G|)$. These data can be kept in memory and, thus, the estimation values $\widetilde{N}(v_0, e)$ and $\widetilde{E}(v_0, e)$ can be computed in constant time. The total preprocessing cost depends on the specific method used:

v_0	LND_{v_0}	LED_{v_0}	$NLND_{v_0}$	$NLED_{v_0}$
A	2.778	17.305	0.723	0.661
B	3.112	19.932	0.810	0.761
C	3.297	21.432	0.858	0.818
D	3.057	19.571	0.796	0.747
E	2.491	15.361	0.648	0.586
F	3.313	21.525	0.862	0.822
G	3.681	24.582	0.958	0.938
H	3.842	26.199	1	1
I	3.547	23.719	0.923	0.905
J	2.854	18.489	0.743	0.706
K	3.443	22.568	0.896	0.861
L	3.712	25.180	0.966	0.961
M	3.685	25.750	0.959	0.983
N	3.439	23.449	0.895	0.895
O	2.719	17.888	0.708	0.683
P	3.182	20.096	0.828	0.767
Q	3.307	21.782	0.861	0.831
R	2.986	19.211	0.777	0.733
S	2.478	14.605	0.645	0.557
T	2.279	12.990	0.593	0.496
U	1.759	9.531	0.458	0.364
MAX	3.842	26.199		

Table 4: Kernel Density Estimators for the graph of Figure 1 with $h = 7$.

- Local Counting Density Estimators need to compute all shortest path distances in the e_c-range region. Therefore, for a starting node v_0, we need $O\left(\frac{e_c}{D_G} \cdot |V_G| \cdot \log |V_G|\right)$ time (e.g., by implementing the Dijkstra algorithm with binary heaps). Finally, for all graph nodes we need $O\left(\frac{e_c}{D_G} \cdot |V_G|^2 \cdot \log |V_G|\right)$ time.

- Kernel Density Estimators need to compute all shortest path distances between all nodes. By using a common "all to all" shortest path algorithm (i.e. Dijkstra), we reach an $O\left(|V_G|^2 \cdot \log |V_G|\right)$ time. However, in large graphs, large distances from a node can not affect significantly its kernel densities. Thus, we can prune some terms of the summations on kernel densities without significant changes to their values, if we select only the nodes with shortest paths less than or equal to a constant multiple of h $(k \cdot h)$ [1]. Then, the total preprocessing cost can be reduced to $O\left(\frac{k \cdot h}{D_G} \cdot |V_G|^2 \cdot \log |V_G|\right)$.

3.5 Binary Encoding Estimation Method

The next method is applicable in any graph type and its efficiency is independent of the parameters e and v_0. Its rationale is based on the following observations. As mentioned, all local densities pre-computations require some or all of the shortest path distances between the graph

[1] In the above calculations the shortest path from node A to U is 22, whereas the contribution of node U to kernel densities of node A is only 0.007163364. Thus, in the above example we can choose the nodes which have shortest path distance $\leq 3h$ $(3h = 3 \cdot 7 = 21)$, and then the kernel densities calculations will remain efficient enough.

nodes. Therefore, a process that could encode all shortest path distances into node labels in a preprocessing step would avoid all computations of network distances.

Along this lines, in [8] the authors present an efficient encoding technique based on hypercube embedding for assigning labels to graph nodes so that the network distance between any two nodes can be calculated directly by using their labels. More specifically, after the encoding process, the node labels are binary numbers and the network distance of any two nodes can be approximated by the Hamming distance of the respective binary codes. This method requires a unitary weighted graph (i.e., all edge weights are equal to 1). Therefore, to apply this technique, we first have to transform the spatial network graph During this transformation all network distances on the final graph must closely approximate the corresponding original distances. For this purpose, we propose the following preprocessing steps:

1. Select a real value w_u for unitary distance from the interval $(0, \overline{w}]$. This constant affects the precision of distance approximations. If it takes values close to $\overline{w}$ then we will have low precision and possibly a high estimation error. If instead it takes values close to 0, then we will have high precision and an almost zero estimation error.

2. Transform the graph G into a new graph $G' = T(G)$, by dividing all weights by w_u and by adding intermediate nodes. In particular, in every edge (v_i, v_j) of G we add a number of intermediate nodes. This number is equal to the closest integer of $\frac{w(v_i,v_j)}{w_u}$ minus 1 (i.e. $\lceil \frac{w(v_i,v_j)}{w_u} \rceil - 1$ or $\lfloor \frac{w(v_i,v_j)}{w_u} \rfloor - 1$). If A_G is the set of all new nodes, then graph G' will have $|V_{G'}| = |V_G| + |A_G|$ and $|E_{G'}| = |E_G| + |A_G|$, whereas all edge weights of graph G' will be equal to 1.

Given G', we can execute the encoding algorithms of [8], and construct these binary codes of length k bits, where k equals the number of edges contained in the perimeter of graph G'. According to [8], $k = O(\sqrt{n})$, where n is the number of nodes of G', thus: $k = O(\sqrt{|V_{G'}|}) = O(\sqrt{|V_G| + |A_G|})$.

Given the binary encoding for all nodes of graph G after the preprocessing phase, we can proceed with the derivation of estimations. Some definitions are first necessary:

- C_v is the binary code of a node $v \in V_G$.
- $H(c_{v_i}, c_{v_j})$ is the Hamming distance between the binary codes c_{v_i}, c_{v_j}, and equals:

$$H(c_{v_i}, c_{v_j}) = \sum_{bits=1} \left(c_{v_i} \oplus c_{v_j} \right)$$

  where $\oplus$ stands for the XOR binary operator.

- $u(x)$ is the simple signal function that returns 1 if $x \geq 0$ and 0 otherwise.

As mentioned, we can estimate all shortest path distances on graph G by computing the Hamming distance of the corresponding binary codes. It can be proven that an estimate for the shortest path distance of two nodes v_i, v_j on the original graph G is:

$$d(v_i, v_j) \approx \frac{H(c_{v_i}, c_{v_j})}{2} \cdot w_u$$

Therefore, we derive the following estimation formulae for the binary encoding method:

$$\boxed{\widetilde{N}(v_0, e) = \sum_{v_i \in V_G} u\left(e - \frac{H(c_{v_0}, c_{v_i}) \cdot w_u}{2}\right)} \tag{9}$$

$$\boxed{\widetilde{E}(v_0, e) = \sum_{(v_i,v_j)\in E_G} u\left(e - w_u \cdot \frac{\min\{H(c_{v_0}, c_{v_i}), H(c_{v_0}, c_{v_j})\}}{2} - w(v_i, v_j)\right)} \tag{10}$$

Next, we will first prove a proposition that will be helpful for the subsequent space and time complexity calculations.

Proposition 3.1. *The number of added nodes and edges $|A_G|$ on graph G to produce graph G' is:*

$$|A_G| \approx |E_G| \cdot \frac{\overline{w} - w_u}{w_u} \tag{11}$$

Proof. In every edge (v_i, v_j) of G we have added $\left(\lceil \frac{w(v_i,v_j)}{w_u} \rceil - 1\right)$ or $\left(\left\lfloor \frac{w(v_i,v_j)}{w_u} \right\rfloor - 1\right)$ intermediate nodes, thus the total number of added nodes is:

$$|A_G| \approx \sum_{(v_i,v_j)\in E_G} \left(\frac{w(v_i, v_j)}{w_u} - 1\right) = \frac{\sum_{(v_i,v_j)\in E_G} w(v_i, v_j)}{w_u} - |E_G|$$

However by the definition of $\overline{w}$ we have:

$$\overline{w} = \frac{\sum_{(v_i,v_j)\in E_G} w(v_i, v_j)}{|E_G|} \Leftrightarrow \sum_{(v_i,v_j)\in E_G} w(v_i, v_j) = \overline{w} \cdot |E_G|$$

Therefore:

$$|A_G| \approx \frac{\overline{w} \cdot |E_G|}{w_u} - |E_G| = |E_G| \cdot \frac{\overline{w} - w_u}{w_u}$$

□

Based on this proposition we can calculate the total number of nodes and edges of graph G' using only basic parameters from the original graph G and Equation (11):

$$n = |V_{G'}| = |V_G| + |A_G| \approx |V_G| + |E_G| \cdot \frac{\overline{w} - w_u}{w_u} \tag{12}$$

$$|E_{G'}| = |E_G| + |A_G| \approx |E_G| + |E_G| \cdot \left(\frac{\overline{w}}{w_u} - 1\right) = |E_G| \cdot \frac{\overline{w}}{w_u} \tag{13}$$

Thus, the length of each binary code is:

$$k = O(\sqrt{n}) = O(\sqrt{|V_{G'}|}) = O\left(\sqrt{|V_G| + |E_G| \cdot \frac{\overline{w} - w_u}{w_u}}\right) \tag{14}$$

3.5.1 Time, Space and Preprocessing Requirements

From Equations (12-14), we observe that the parameter w_u affects the number of nodes and edges on graph G', and subsequently the length of the binary encoding k. As parameter w_u takes values close to 0, k becomes larger, and thus more space is needed for the encoding. Therefore, there is a trade-off between the required space and the estimation accuracy of this method.

According to Equation (14), we need $O\left(|V_G| \cdot \sqrt{|V_G| + |E_G| \cdot \frac{\overline{w} - w_u}{w_u}}\right)$ space for the binary codes. Thus, in shortage of space, we can calibrate the w_u value to the available space and keep these data in memory.

The required time for computations of estimators $\widetilde{N}(v_0, e)$ and $\widetilde{E}(v_0, e)$, will be only the time required for the calculations of their formulae, which are $O(|V_G|)$ and $O(|E_G|)$ respectively. However, this time cost is negligible, since these calculations have only binary and basic register operations.

The preprocessing cost is composed of the time required for the construction of G' from G:

$$O(|V'_G| + |E'_G|) = |V_G| + |E_G| \cdot \frac{\overline{w} - w_u}{w_u} + |E_G| \cdot \frac{\overline{w}}{w_u} = |V_G| + |E_G| \cdot \frac{2\overline{w} - w_u}{w_u}$$

plus the time required for the encoding of G', which according to [8] and Equation (14) is:

$$O(n\sqrt{n}) = O\left(\left(|V_G| + |E_G| \cdot \frac{\overline{w} - w_u}{w_u}\right) \cdot \sqrt{|V_G| + |E_G| \cdot \frac{\overline{w} - w_u}{w_u}}\right)$$

3.6 Comparison of Methods

Table 5 summarizes the theoretical results regarding the performance of the estimation methods. Note that, for the MDS method only node estimation time is given since edge estimation is not supported. These results are discussed below:

Method	**Estimation Time** (node/edge)	**Space Required**	**Preprocessing Time**
MDS-Grid	$O\left(\prod_{i=1,\ldots,k} \lceil \frac{2e}{dx_i} \rceil\right)$	$O\left(k \cdot \lvert V_G \rvert + c^k\right)$	$O\left(\lvert V_G \rvert^3 + c^k\right)$
Global	$O(1)/O(1)$	$O(1)$	$O(V_G + E_G)$
Local	$O(1)/O(1)$	$O(\lvert V_G \rvert)$	$O\left(\frac{e_c}{D_G} \cdot \lvert V_G \rvert^2 \cdot \log \lvert V_G \rvert\right)$
Kernel	$O(1)/O(1)$	$O(\lvert V_G \rvert)$	$O\left(\frac{k \cdot h}{D_G} \lvert V_G \rvert^2 \cdot \log \lvert V_G \rvert\right)$
Binary	$O(\lvert V_G \rvert)/O(\lvert E_G \rvert)$	$O\left(\lvert V_G \rvert \cdot \sqrt{\lvert V_G \rvert + \lvert E_G \rvert \cdot \frac{\overline{w} - w_u}{w_u}}\right)$	$O(n\sqrt{n}),\ \ n = \left(\lvert V_G \rvert + \lvert E_G \rvert \cdot \frac{\overline{w} - w_u}{w_u}\right)$

Table 5: Complexities for estimation, space requirements and preprocessing.

Estimation time. The MDS-Grid method is not efficient due to high time, space and preprocessing requirements. On the contrary, the Global, Local and Kernel methods perform fast calculations for both node and edge estimations. In addition, as desired, the estimation time does not depend on the parameters v_0, e, e_c, h, $|V_G|$, $|E_G|$.

The Binary method requires $|V_G|$ and $|E_G|$ binary and register operations for the node and edge estimation, respectively. This cost is independent of the parameters v_0 and e, but depends on the graph parameters $|V_G|$ and $|E_G|$) and the unitary distance w_u, which affects the code length. However, due to the usage of only binary and register operations, the estimation time is expected to be low.

Required space. The Global method does not have any additional space requirements. Local and Kernel methods require $O(|V_G|)$ space, which is independent of the parameters v_0, e, e_c, h.

The Binary method requires an $O\left(|V_G| \cdot \sqrt{|V_G| + |E_G| \cdot \frac{\overline{w} - w_u}{w_u}}\right)$ space. By decreasing the value of w_u, the required space increases significantly, due to the increase of the required number of bits for the encoding (k), thus, achieving more accurate estimations.

Preprocessing time. The Global method has the shortest preprocessing time, which is linear on the number of edges. The Local and Kernel methods have quadratic preprocessing time due to the shortest path computations. There is a significant gain in the Local method (e.g., the coefficient $\frac{e_c}{D_G}$), because all shortest path computations end when the e_c-range is reached. Also, there is a significant gain in the Kernel method (the coefficient $\frac{k \cdot h}{D_G}$), if we end all shortest path computations when the $(k \cdot h)$-range is reached.

The Binary method has a sub-quadratic preprocessing time, which depends on both the graph size and the w_u parameter. It is evident, that by decreasing the value of w_u, the preprocessing time is expected to increase significantly.

4 Experiments and Results

In this section, we present experimental results for all the presented estimation methods, which have been implemented on a Pentium IV 3GHz CPU, with 1GB RAM. We have used both real-life and synthetic spatial networks and we have tested the proposed methods for several parameter values. For brevity, we present only a small set of representative results, which depict the most significant performance issues and trade-offs.

(a) Oldenburg (OL) (b) California (CA) (c) San Francisco (SF)

Figure 6: Data sets.

The three real-life networks have been downloaded from [16] and correspond to the road networks of Oldenburg (OL), California (CA) and San Francisco (SF). The synthetic network (UN) is a grid-based graph constructed as in [6]. The grid is a square of 500×500 nodes corresponding to points with coordinates (i, j), for $1 \leq i, j \leq 500$. Each node has edges to its neighbors (left, right, up, down), if present. These edges have random real weights chosen uniformly from the interval $[12, 18]$. Table 6 depicts the graph parameters for the four spatial networks, whereas Figure 6 depicts the three real-life networks used.

Global Parameters	**Oldenburg**	**California**	**San Francisco**	**Synthetic**
$\|V_G\|$	6,105	21,048	174,956	250,000
$\|E_G\|$	7,035	21,693	223,001	499,000
$\overline{deg}$	2.304668	2.061289	2.549224	3.992
$\overline{w}$	73.67902	0.01618624	8.782676	15.00156
D_G	12,985.97	16.4288	16,828.54	13,366.2

Table 6: Graph parameters values for the four used spatial networks.

Parameter	Oldenburg	California	San Francisco	Synthetic
e	0-3250 step 10	0-4.1 step 0.01	0-4200 step 10	0-3340 step 10
e_c (Local)	75-3000 step 75	0.015-0.600 step 0.015	9-360 step 9	15-600 step 15
h (Kernel-Hi)	75-3000 step 75	0.015-0.600 step 0.015	9-360 step 9	15-600 step 15
h (Kernel-Lo)	5-150 step 5	0.001-0.030 step 0.001	0.5-15 step 0.5	1-30 step 1
w_u (Binary)	5-150 step 5	0.001-0.030 step 0.001	0.5-15 step 0.5	1-30 step 1

Table 7: Adjustable parameters ranges table.

In all experimental results, we have randomly selected a set S of starting nodes v_0, where $|S| = 5\%|V_G|$. We also performed e-range queries with e varying from 0 to $\approx \frac{D_G}{4}$ (which is half of the graph radius), with a small increasing step (0.01 in CA and 10 in OL, SF, UN). We calculated the $N(v_0, e), E(v_0, e)$ values and the corresponding estimations $\widetilde{N}(v_0, e)$ and $\widetilde{E}(v_0, e)$ for all methods with several parameters setups. For the Local, Kernel and Binary methods, for tuning reasons we varied their parameters in high (using values near multiples 1 to 40 of $\overline{w}$) and low ranges (using values near subdivisions $\frac{1}{15}$ to $\frac{30}{15}$ of $\overline{w}$). Table 7 depicts the final selected ranges for all parameters. Note that in the Kernel method we tested both high and low bandwidth ranges, because h can give efficient estimation results in both cases.

Next, we have calculated all average real values $N_{avg}(e)$ and $E_{avg}(e)$ and estimates $\widetilde{N}_{avg}(e)$ and $\widetilde{E}_{avg}(e)$, as follows:

$$N_{avg}(e) = \frac{1}{|S|} \sum_{v_0 \in S} N(v_0, e) \qquad E_{avg}(e) = \frac{1}{|S|} \sum_{v_0 \in S} E(v_0, e)$$

$$\widetilde{N}_{avg}(e) = \frac{1}{|S|} \sum_{v_0 \in S} \widetilde{N}(v_0, e) \qquad \widetilde{E}_{avg}(e) = \frac{1}{|S|} \sum_{v_0 \in S} \widetilde{E}(v_0, e)$$

In all methods we have used the following formulae as estimation error functions:

$$Error[N](e) = \frac{\left| N_{avg}(e) - \widetilde{N}_{avg}(e) \right|}{N_{avg}(e)} \qquad Error[E](e) = \frac{\left| E_{avg}(e) - \widetilde{E}_{avg}(e) \right|}{E_{avg}(e)}$$

4.1 Experimental Results for the MDS-Grid Method

Due to the significant preprocessing requirements, we have applied this method only to the OL data set, which has a small number of nodes (6,105). For this setting, a dissimilarity matrix with $6,105^2 = 37,271,025$ shortest path distances has been constructed. The applied MDS transformation returns a 6,105 $\times$ 3,221 matrix, thus all node-points may be considered as objects in a 3221-dimensional Euclidean space. Evidently, we have 3221 positive eigenvalues, where we can select some $2 \leq k < 3221$ dimensions for dimension reduction purposes. Table 8 depicts the 20 greatest eigenvalues sorted in descending order and normalized to the interval [0,1].

We observe that the first four eigenvalues are much greater than the remaining ones. Therefore, a reduction to 4 dimensions is fair, whereas a reduction to 2 or 3 dimensions is rather poor. Figure 7 depicts the OL node-points after the MDS transformation with a reduction to 2 dimensions, where the general shape of the graph has been kept only roughly. All eigenvalues after the fourth one are much smaller, thus selecting more than 4 dimensions will not result in any significant increase of distance preservation quality.

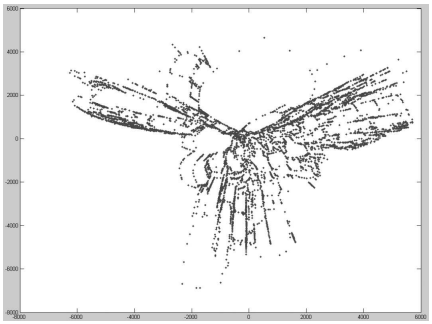

Figure 7: OL node-points set after MDS transformation in 2 dimensions.

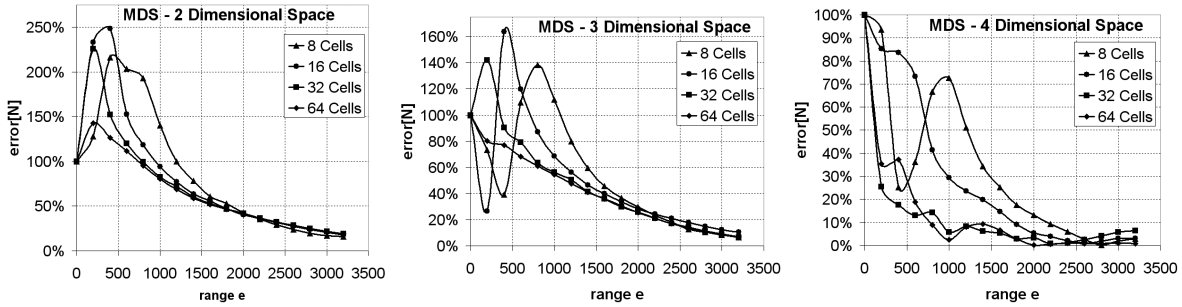

Figure 8: Estimation error with MDS-Grid Method.

Figure 8 depicts the estimation error for the MDS-Grid Method. As we increase the number of cells c or the number of dimensions k, we observe smaller peaks in the estimation error. However, the estimation error in general remains high. For $k = 2$ dimensions we have a maximum estimation error peak of 220%, 250%, 225%, 150% when $c = 8, 16, 32, 64$, respectively, which falls under 50% for large ranges. For $k = 4$ dimensions we have a maximum estimation error peak of 78%, 87%, 65%, 46%, when $c = 8, 16, 32, 64$, respectively, which falls under 10% for large ranges.

Table 9 presents the space requirements for the MDS-Grid Method (assuming 8 bytes per real number). We observe that the required space grows exponentially as we increase the number of the selected dimensions k.

In conclusion, the MDS transformation of the OL data set returned a large number of positive eigenvalues, thus we must use many of the k dimensions to achieve efficient estimation results. Also, even if we use many dimensions, we must increase the number of cells c to decrease the error peaks. Moreover, even if we increase both parameters c and k, this method gives efficient estimation results for large e values (e.g., > 2000). However, as we increase k, c and e, we result

1	0.47461	0.17330	0.11724	0.05526
0.02900	0.02306	0.01964	0.01564	0.01340
0.01160	0.01095	0.00967	0.00841	0.00722
0.00627	0.00569	0.00507	0.00484	0.00464

Table 8: Top-20 normalized eigenvalues of MDS transformation in OL data set.

in significant time/space complexities and deteriorated estimation performance.

4.2 Kernel Density Estimator Functions Comparison

We have examined the accuracy of the following kernel functions embedded in the Kernel Density Estimation Method:

$$\begin{aligned} Triangle &: (1-|x|)I(|x|\leq 1) \\ Epanechnikov &: \frac{3}{4}(1-x^2)I(|x|\leq 1) \\ Quartic &: \frac{15}{16}(1-x^2)^2 I(|x|\leq 1) \\ Triweight &: \frac{35}{32}(1-x^2)^3 I(|x|\leq 1) \\ Gaussian &: \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} \\ Cosinus &: \frac{\pi}{4} cos(\frac{\pi}{2}x) I(|x|\leq 1) \end{aligned}$$

where the function $I(|x| \leq 1)$ is defined as:

$$I(x) = 1, \quad \text{if} \quad |x| \leq 1 \quad \text{and} \quad I(x) = 0, \text{otherwise}$$

The following proposition unifies the two Local Densities Estimation Methods, i.e. the Local Counting Density estimators and the Kernel Density estimators. The Local method is a special case of the Kernel method when used with uniform kernels.

Proposition 4.1. *The Kernel Density Estimator Method with Uniform Kernels provides the same results with the Local Counting Method, when $h = e_c$.*

Proof. We will prove that in this case the estimation formulae of the two methods are identical. We present the proof for node estimation only, since the proof for edge estimation is similar.

Let us denote the number $|\{v_i \in V_G : d(v, v_i) \leq e_c\}|$ as L_v. Then, for the Local Counting Method it holds that $LND_v = L_v$, whereas for the Kernel Density Estimators Method we have:

$$LND_v = \sum_{v_i \in V_G} K\left(\frac{d(v,v_i)}{h}\right) = \sum_{v_i \in V_G : \frac{d(v,v_i)}{h} \leq 1} K\left(\frac{1}{2}\right) = \sum_{v_i \in V_G : d(v,v_i) \leq h} K\left(\frac{1}{2}\right)$$

$$= \frac{1}{2} \cdot \Big|\{v_i \in V_G : d(v, v_i) \leq h\}| = \frac{1}{2} \cdot |\{v_i \in V_G : d(v, v_i) \leq e_c\}\Big| = \frac{1}{2} \cdot L_v$$

Therefore, the corresponding normalized local node density for the Local Counting Method is: $NLND_v = \frac{L_v}{\max L_v}$, whereas for the Kernel Method is: $NLND_v = \frac{\frac{1}{2} \cdot L_v}{\frac{1}{2} \cdot \max L_v} = \frac{L_v}{\max L_v}$. □

	$k=2$	$k=3$	$k=4$
$c=8$	97,936	148,568	211,744
$c=16$	98,704	162,904	457,504
$c=32$	101,776	277,592	4,389,664
$c=64$	114,064	1,195,096	67,304,224

Table 9: Space requirements for the MDS-Grid method (in bytes).

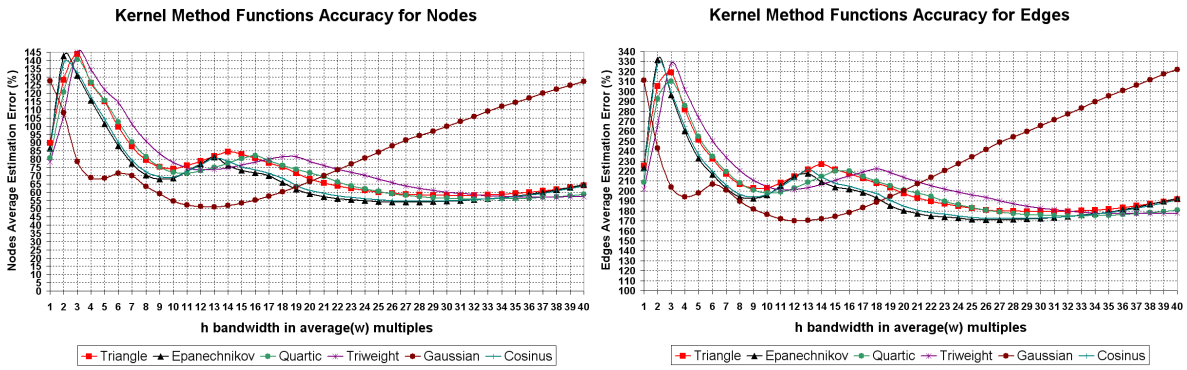

Figure 9: Kernel Method functions accuracy comparison in California network.

Next we present comparative results for the Kernel Method using all the aforementioned kernel functions except the uniform one because of Proposition 4.1 and the fact that the Local Counting Method has been studied separately. For this purpose, we vary the bandwidth of the Kernel Method for any selected kernel function in high and low ranges and record the average node and edge estimation errors for the used e values. Figure 9 depicts the final results for the California network (CA) for large e values, since similar results are obtained for the other spatial networks. We observe that:

- The Gaussian function appears to have the best average estimation error for both node and edge estimations, for $h \approx 12 \cdot \overline{w}$. In particular, it produces a minimum average estimation error about 51% in node estimations and about 170% in edge estimations.

- All other kernel functions have similar variations in node and edge estimations, where their minimum average estimation error is reached for higher multiples of $\overline{w}$. More specifically, for the Triangle, Epanechnikov, Quartic, Triweight, Cosinus kernel functions, the global minimum is reached in $h \approx 30 \cdot \overline{w}, 27 \cdot \overline{w}, 33 \cdot \overline{w}, 37 \cdot \overline{w}, 27 \cdot \overline{w}$, respectively. For those bandwidths, the node average estimation errors are: 58%, 54%, 56%, 57%, 54%, respectively, and the edge average estimation errors are: 180%, 171%, 175%, 177%, 172%, respectively.

For the rest of the performance evaluation we adopt the Gaussian kernel function for Kernel Method estimations since it provides better estimation accuracy.

4.3 Tuning the Proposed Methods

It has been mentioned that by tuning the parameters of the Local, Kernel and Binary methods the estimation error can be reduced significantly. Figure 10 depicts the results for all methods where we observe that:

- The Local method has been examined in high ranges of the distance e_c, because if $e_c < \overline{w}$ then we underestimate many local densities and the estimation error increases.

- The Kernel method has been tested in both high and low bandwidth ranges, because the h_m may appear in both ranges.

- The Binary method has been tested only in low ranges, because if $w_u > \overline{w}$, then the node and edge estimation errors grow significantly.

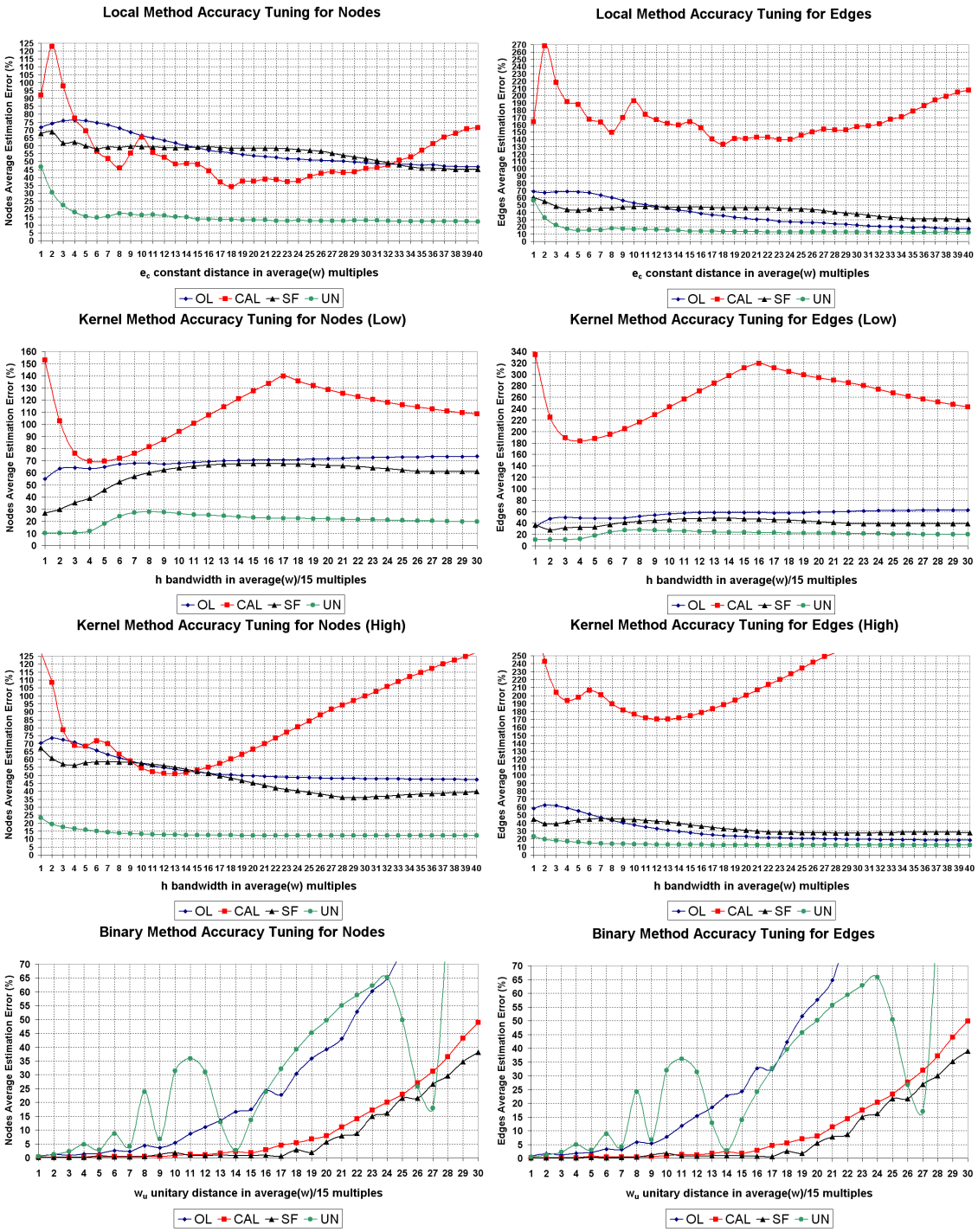

Figure 10: Tuning parameters in proposed methods.

- In all cases, the values that minimize the node average estimation error, minimize also the edge average estimation error.
- In many cases, local minimum values may appear, but there is only one global minimum value in the whole parameter range in which we must attune the methods.

- In Local methods, often the global minimum is reached in high parameter values but the gain in estimation error is not significant. Therefore, we must wisely choose the lower parameter value that produces an estimation error near the global minimum's to minimize the average estimation error and the corresponding preprocessing cost.

- In the Binary method, the average node and edge estimation errors decrease by decreasing the w_u value, but space requirements increase. Therefore, we must wisely choose the maximum parameter value that produces node and edge average estimation error under a small threshold (e.g., 3%), to minimize the average estimation error and the corresponding space requirements.

Table 10 summarizes the final selected parameter values for the Local, Kernel and Binary methods. These values achieve the best estimation accuracy with the shortest preprocessing time.

Network	**Local**	**Kernel**	**Binary**
Oldenburg	$e_c = 39\overline{w}$	$h = 32\overline{w}$	$w_u = \frac{7\overline{w}}{15}$
California	$e_c = 18\overline{w}$	$h = 12\overline{w}$	$w_u = \overline{w}$
San Francisco	$e_c = 35\overline{w}$	$h = \frac{2\overline{w}}{15}$	$w_u = \overline{w}$
Synthetic	$e_c = 15\overline{w}$	$h = \frac{2\overline{w}}{15}$	$w_u = \frac{14\overline{w}}{15}$

Table 10: Final selected parameter values after tuning.

4.4 Estimation Results after Tuning

In this section, we present comparative results regarding the accuracy of the four methods tuned as above for the whole spectrum of e values. Figure 11 depicts the respective node and edge estimation errors.

With respect to the **OL network** we observe that:

- The Global method achieves a node estimation error below 15% and an edge estimation error near 40%. The node estimation error is significantly better because the nodes in OL graph are almost uniformly distributed in space, but edge weights differ significantly.

- The Local and Kernel methods achieve node a estimation error near 50% and an edge estimation error under 20%. The edge estimation error is significantly better because Local methods can detect easier the edge weights distribution through the used local densities.

- The Binary method returns node and edge estimation errors under 5%, being the most accurate estimation method for the OL data set.

In conclusion, in graphs with an almost uniform distribution of nodes in space but non-uniform distribution of edges (like the OL data set), the Global method returns good node estimation results, the Local methods return good edge estimation results, whereas the Binary method returns the most accurate results for both node and edge estimations.

With respect to the **CA network** we observe that:

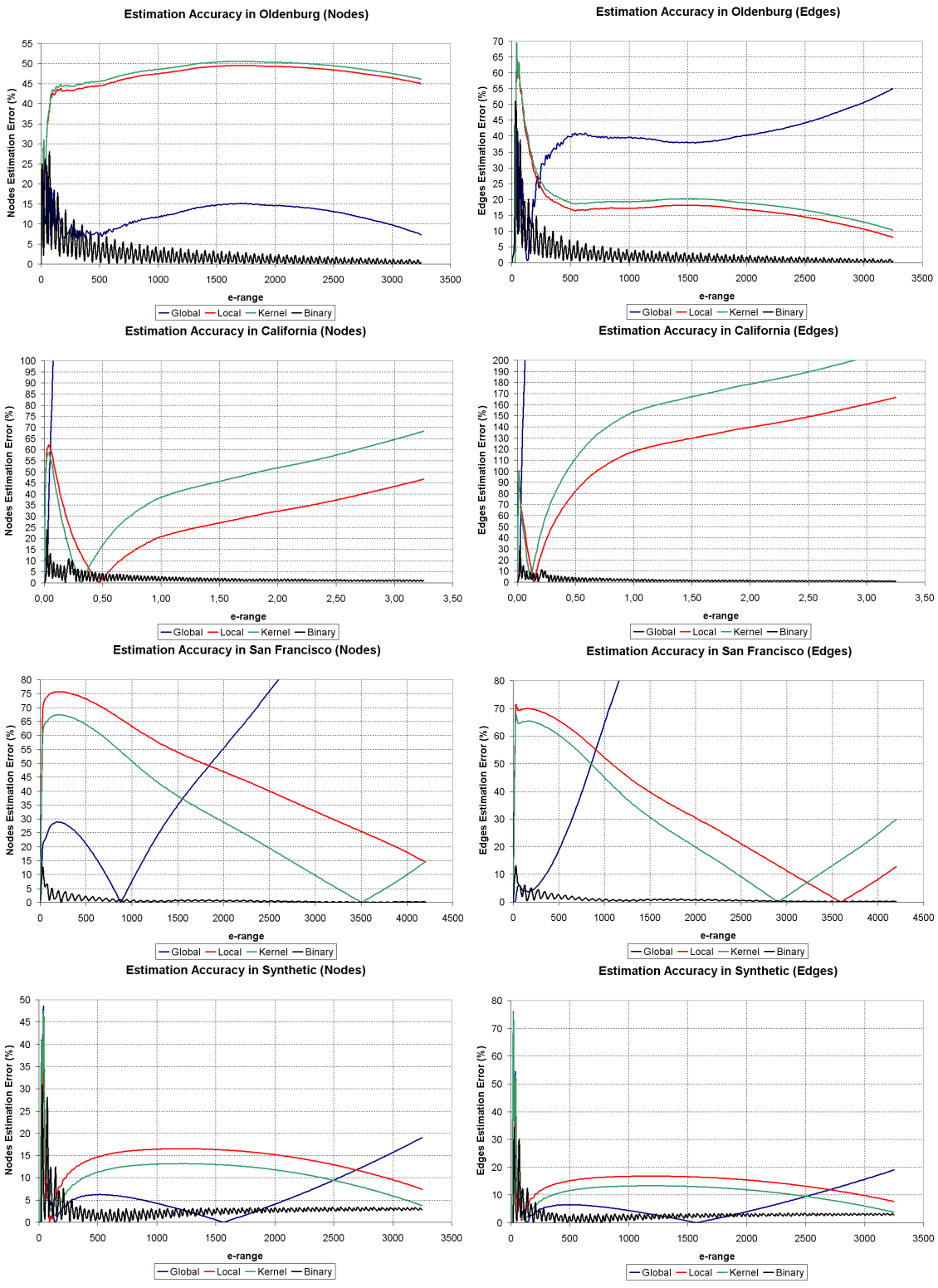

Figure 11: Estimation results after tuning.

- The Global method does not perform well, returning node and edge estimation error above 100% even for small e values. This happens because the CA graph contains mostly highways (i.e. long paths with many nodes) and, therefore, most node degrees are equal to 2. The Global method fails to estimate the numbers of nodes and edges in such cases.

- The Kernel method achieves good node estimation error for low ranges, but the error increases above 40% for high e values. In addition, it achieves high edge estimation error (above 100%). Thus, the Kernel method is better than Global method, but not effective enough.

- The Local method returns better results than the Kernel method, with an improvement of 20% in node estimation error and 40% in edge estimation error, but again these results are not satisfactory.

- The Binary method returns node and edge estimation errors under 5%, being the most accurate estimation method for the CA data set.

In conclusion, in graphs with an average degree around 2 (like the CA data set, where $\overline{deg} = 2.06$), the Global method fails, the Local and Kernel methods give better but not efficient estimations, whereas the Binary method prevails in accuracy and efficiency.

With respect to the **SF network** we observe that:

- The Global method achieves a node estimation error under 30% for small e values, whereas the error increases above 100% for large values. In addition, the edge estimation error in several cases raises above 100%. As the SF graph is a non-regular uniform graph with diverse node and edge densities, the Global method fails to achieve satisfactory results.

- The Local method returns a node estimation error near 75% for small e values, which falls under 30% for large e values. Also, the edge estimation error is near 70% for small e values and under 20% for large e values. Therefore, the Local method gives better estimation results than the Global method, but with large error rate variations.

- The Kernel method returns better results than the Local method, showing an 20% improvement in node and 10% in edge estimation error. However, the Kernel method is not stable and efficient enough.

- The Binary method returns node and edge estimation error under 5%, being the most accurate estimation method for the SF network.

In conclusion, in non-regular uniform graphs (like the SF data set), the Global method returns inaccurate estimations, the Local and Kernel methods return better but not stable estimations as they are sensitive to the parameters h and e_c, whereas the Binary method outperforms the other methods.

Finally, regarding the **Synthetic (UN) network** we observe that:

- The Global method gives excellent node and edge estimation results with an error rate under 7%. This happens because the UN graph is almost regular uniform.

- The Local and Kernel methods give good node and edge estimation results with error rates under 17% and 13%, respectively. Therefore, local node and edge densities are inferior in comparison to the Global method.

- The Binary method returns node and edge estimation errors under 5%, being the most accurate method for the UN network.

In regular uniform and almost regular uniform graphs (like the UN data set), all methods return good estimation results (error under 20%). Moreover, the Global method offers better estimation results than the Local methods, whereas the Binary method again achieves the most accurate results in comparison to the other approaches.

4.5 Space Requirements

We applied the proposed methods on the four presented networks, with parameters tuned as in Section 4.2, and we measured the required space. For fairness, we measured also the required space to store only the original graph data (node ID's, edges and weights) as a table of linked lists in main memory. Table 11 depicts the corresponding results.

Network	Graph size	Local-Kernel	Binary
OL	0,254	0,047	0,069
CA	0,818	0,161	0,291
SF	7,774	1,335	6,979
UN	15,236	1,907	12,742

Table 11: Space requirements for all methods and networks (in MB).

We observe that the results comply with the observations in Sections 3.3-3.6. Moreover, all methods require less space than the graph data. More specifically:

- The Global method needs no extra space, and therefore it does not appear in the results.
- The Local and Kernel methods require much less space than the graph data.
- The tuned Binary method requires more space than the Local/Kernel methods but still less than the graph data.

4.6 Estimation Time Comparison and Experiments

In this section, we further study the estimation times of the proposed methods, which are also significant for their efficiency. As presented in Section 3.6, the Global, Local and Kernel methods perform estimations in constant time and therefore the estimation cost does not depend on the values of the parameters.

As we have already presented in Section 3.6, Binary method's estimation times are independent of the parameters v_0, e, but depend on the graph parameters ($|V_G|$ and $|E_G|$) and the unitary distance w_u. Therefore, we expect the estimation time to grow in larger graphs or small unitary distances. However, the Binary method's formulae involve binary and register operations, thus the expected estimation times are still satisfactory.

To evaluate these observations, we measure the node and edge estimation times for all proposed methods after tuning and for all spatial networks, by varying the e parameter as shown in Table 7. For fairness, we also measure the time required by the Dijkstra algorithm (i.e., an implementation with priority queues similar to [27]), which has an $(O(|E_G| + |V_G| \log \log |V_G|))$ performance, applied in those networks for the same e values. The corresponding results are given in Figure 12.

We observe that all the proposed methods are time efficient. Moreover, we confirm that:

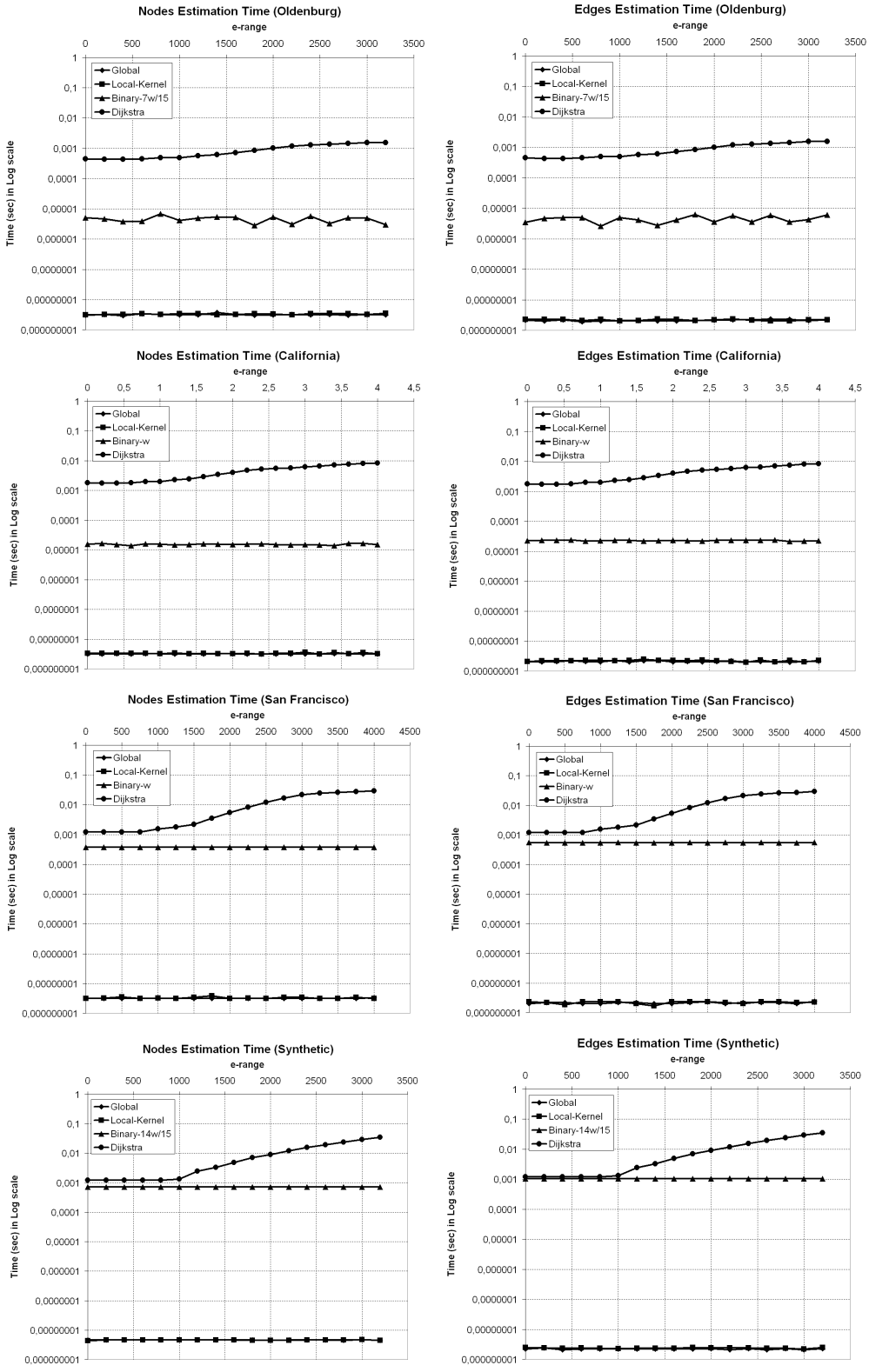

Figure 12: Estimation costs.

- The Global, Local and Kernel methods give almost constant node and edge estimation (in the level of nsecs).
- The Binary method gives almost constant node and edge estimation times (i.e., less than 1 msec), which grows slowly and almost linearly by increasing the distance e.
- The Dijkstra algorithm implementation returns scanning times at the level of msecs, which increase by increasing the distance e. In addition, this time cost is always greater than that of the other methods.

5 Discussion

In this section, we present some interesting aspects regarding the usability of the estimation methods.

5.1 Avoid Tuning

The proposed methods (except from Global) require fine-tuning of their basic parameters to minimize the estimation errors. However, the exhaustive tuning leads to high preprocessing calculations and time, especially for large graphs. Therefore, an important question is: "can we decide the parameter values without exhaustive tuning?".

By observing the tuning results of Figure 10, we can see that in many cases the average estimation error gain from value to value is not important. Thus, we can avoid to select the parameter values that produce the global minimum average estimation errors, and to select different values that produce average estimation errors close to the global minimums. Therefore, we performed clustering on average estimation error results for several real and synthetic data sets and for the same parameter ranges, and we observed the following:

- The Local Method with global constant radius $e_c = \sqrt{\frac{D_G}{4\overline{w}}}$ returns average estimation errors close to the global minimums for the most graphs.
- The Kernel Method with bandwidth $h = \frac{\overline{w}}{3}$ again returns average estimation errors close to the global minimums for the most graphs.
- The Binary Method with unitary distance $w_u = \frac{\overline{w}}{2}$ returns excellent average estimation errors for most graphs. Moreover, with the choice $w_u = \frac{\overline{w}}{3}$ the estimations are very accurate with average errors under 5% for almost all graphs. At the same time, both choices keep the required space for the encoding low.

5.2 Networks with Spatial Objects

In this section, we give some useful directions on how the proposed estimation methods can be applied on data sets with spatial objects that are uniformly distributed on the spatial network, in order to estimate: (i) the selectivity of range and k-NN queries, and (ii) the corresponding I/O cost of these queries.

5.2.1 Selectivity Estimation of Range Queries

Let G be a spatial network and S a data set with spatial objects that are uniformly distributed on the edges of G. Then, the total number of edges in the graph G is $|E_G|$ and the total number of spatial objects is $|S|$. The average number of spatial objects that lie on an edge (which we denote as $\overline{s}$) is the total number of the objects multiplied by the average distance of an edge and divided by the sum of distances of all edges of the graph (due to the uniform distribution of objects on the edges). Thus, if we assume that the weights of all edges are their corresponding distances, we have:

$$\overline{s} = |S| \cdot \frac{\overline{w}}{\sum w(v_i, v_j)} = \frac{|S| \cdot \overline{w}}{|E_G| \cdot \overline{w}} = \frac{|S|}{|E_G|}$$

A range query initiated at a node $v_0 \in G$ with a positive range distance e returns $\widetilde{E}(v_0, e)$ total edges. Therefore, we can estimate the selectivity of such a range query on the spatial objects data set (which we denote $\widetilde{S}(v_0, e)$) with the following formula:

$$\widetilde{S}(v_0, e) = \overline{s} \cdot \widetilde{E}(v_0, e) = \frac{|S| \cdot \widetilde{E}(v_0, e)}{|E_G|}$$

The estimator $\widetilde{E}(v_0, e)$ in the last formula can be computed with any of the previously discussed methods.

5.2.2 Selectivity Estimation of k-NN Queries

Nearest-Neighbor queries on spatial objects can be efficiently estimated using the proposed methods. For a k-NN query initiated at node $v_0 \in G$, we can estimate the selectivity (which we denote $\widetilde{S}(v_0, k)$) by finding the minimum proper range distance radius $e_{min} > 0$ such that the number of returned spatial objects is at least k, and by using this radius in the previous range query estimators. Ideally, the selectivity of a k-NN query must be equal to k, but using the range queries estimators the returned objects may be more than k. Thus, we must have:

$$\widetilde{S}(v_0, k) = \widetilde{S}(v_0, e_{min}) = \frac{|S| \cdot \widetilde{E}(v_0, e_{min})}{|E_G|} \geq k$$

or

$$\widetilde{E}(v_0, e_{min}) \geq \frac{k \cdot |E_G|}{|S|}$$

The computation of the minimum proper range distance radius e_{min} is different for the proposed methods. In the Global method the edges estimator is equal to: $\widetilde{E}(v_0, e) = \overline{deg} \cdot \frac{e^2}{\overline{w}^2}$, thus we have:

$$\overline{deg} \cdot \frac{e_{min}^2}{\overline{w}^2} \geq \frac{k \cdot |E_G|}{|S|}$$

or

$$e_{min} \geq \overline{w} \sqrt{\frac{k \cdot |E_G|}{|S| \cdot \overline{deg}}}$$

In Local and Kernel method the edge estimator is equal to: $\widetilde{E}(v_0, e) = NLED_{v_0} \cdot \overline{deg} \cdot \frac{e^2}{\overline{w}^2}$, thus we have:

$$NLED_{v_0} \cdot \overline{deg} \cdot \frac{e_{min}^2}{\overline{w}^2} \geq \frac{k \cdot |E_G|}{|S|}$$

or

$$e_{min} \geq \overline{w}\sqrt{\frac{k \cdot |E_G|}{|S| \cdot \overline{deg} \cdot NLED_{v_0}}}$$

The normalized local edge density $NLED_{v_0}$ for node v_0 is computed using the presented formulae of Local and Kernel methods.
In the Binary method the computation of e_{min} is based on the inequality $\widetilde{E}(v_0, e_{min}) \geq \frac{k \cdot |E_G|}{|S|}$ and the algorithmic procedure of Figure 13.

Algorithm Binary-$e_{min}(E_G, |S|, k)$

---

1. m=0
2. **do**
3. m = m + 1
4. e = m * w_u
5. compute the estimator $\widetilde{E}(v_0, e)$
6. **while** $\widetilde{E}(v_0, e) < \frac{k \cdot |E_G|}{|S|}$
7. return e

Figure 13: Computing the e_{min} with Binary method.

5.2.3 I/O cost of Range Queries

The I/O cost of spatial queries is strongly depended on the storage scheme of the spatial objects and the spatial network. In this section, we estimate the I/O cost assuming a storage scheme similar to that of [15]. More specifically, Papadias et al [15], propose to separate the storage of spatial objects from that of the underlying spatial network, to index the spatial objects using an R-tree, and to index the spatial network using a second R-tree and adjacency components which contain the adjacency lists of the network. The adjacency lists of nodes close in space are placed in the same disk page using Hilbert ordering.

Let us assume that we follow the range query processing methodology of [15], and more specifically the Range Euclidean Restriction (RER) algorithm. As RER first performs a range query into the spatial objects data set within Euclidean distance e from the starting node v_0 by accessing the spatial objects R-tree, we will have the first significant I/O cost. This I/O cost is equal to the total number of disk accesses required to execute the range query using the R-tree indexing the spatial objects. This cost is denoted by DA_S. In the next step of the algorithm, a network expansion is performed by examining all edges within network distance e from v_0, in order to detect and delete the false positives objects which may have been returned by the previous step. For simplicity, we assume that this is achieved again by executing another range query within distance e in the network R-tree. Evidently, this way we over-estimate the number of disk accesses, since we perform a Euclidean-based range query on the network R-tree which may produce false alarms that therefore must be eliminated. The associated I/O cost is equal to the total number of disk accesses required for the execution of the range query in the network R-tree. We denote this cost by DA_G. As both range queries are performed only once, we can estimate the total I/O cost of the spatial range query by summing the two components: $DA_S + DA_G$.

An efficient estimator of total disk accesses for spatial range queries on R-trees is presented in [26]. More specifically, having the data set size and density, the average node capacity of the R-tree and the query window size, the total disk accesses of the query can be efficiently estimated.

For simplicity, let us assume that the spatial objects are 2-dimensional points and the underlying spatial network is 2-dimensional too. Then, using the formula proposed by [26], the total disk accesses of the spatial objects R-tree for a range query with radius e, are:

$$DA_S = 1 + \sum_{j=1}^{1+\lceil \log_{f_S} \frac{|S|}{f_S} \rceil} \left(\sqrt{D_j} + \frac{2e}{D_G} \cdot \sqrt{\frac{|S|}{f_S^j}} \right)^2$$

where

$$D_j = \left(1 + \frac{\sqrt{D_{j-1}} - 1}{\sqrt{f_S}} \right)^2,$$

$D_0 = 0$ is the density of the data set S (which is 0 as it contains only points), f_S is the average node capacity of the spatial objects R-tree, $|S|$ is the data set size and D_G is the diameter of the underlying network G. We must note that the proposed formula actually works for hyper-rectangular window queries and not hyper-spherical. Thus, always we take the MBR of the hyper-sphere to apply the formula, making an overestimation. In our example of 2-dimensional spaces, we take the square range area with edge equal to $2e$.

Similarly, we can estimate the total disk accesses of the network R-tree:

$$DA_G = 1 + \sum_{j=1}^{1+\lceil \log_{f_G} \frac{|G|}{f_G} \rceil} \left(\sqrt{D'_j} + \frac{2e}{D_G} \cdot \sqrt{\frac{|G|}{f_G^j}} \right)^2$$

where

$$D'_j = \left(1 + \frac{\sqrt{D'_{j-1}} - 1}{\sqrt{f_G}} \right)^2,$$

D'_0 is the density of the network G (which can be computed from the MBR's of the poly-line components), f_G is the average node capacity of the network R-tree, $|G|$ is the total number of the network R-tree nodes and D_G is the diameter of the network G.

Therefore the total I/O cost for an e-range query is:

$$DA_S + DA_G = 2 + \sum_{j=1}^{1+\lceil \log_{f_S} \frac{|S|}{f_S} \rceil} \left(\sqrt{D_j} + \frac{2e}{D_G} \cdot \sqrt{\frac{|S|}{f_S^j}} \right)^2 + \sum_{j=1}^{1+\lceil \log_{f_G} \frac{|G|}{f_G} \rceil} \left(\sqrt{D'_j} + \frac{2e}{D_G} \cdot \sqrt{\frac{|G|}{f_G^j}} \right)^2$$

5.2.4 I/O cost of k-NN Queries

Having computed the corresponding e_{min} distance for the k-NN queries using all the proposed methods, we can efficiently estimate the I/O cost of such queries using the presented storage scheme and formulae of subsection 5.3.3. Therefore, by substituting in these formulae the range distance e with the e_{min} distance, we have the total I/O cost for k-NN queries:

$$DA_S + DA_G = 2 + \sum_{j=1}^{1+\lceil \log_{f_S} \frac{|S|}{f_S} \rceil} \left(\sqrt{D_j} + \frac{2e_{min}}{D_G} \cdot \sqrt{\frac{|S|}{f_S^j}} \right)^2 + \sum_{j=1}^{1+\lceil \log_{f_G} \frac{|G|}{f_G} \rceil} \left(\sqrt{D'_j} + \frac{2e_{min}}{D_G} \cdot \sqrt{\frac{|G|}{f_G^j}} \right)^2$$

where

- for the Global Method: $e_{min} = \overline{w}\sqrt{\frac{k \cdot |E_G|}{|S| \cdot \overline{deg}}}$
- for the Local and Kernel Method: $e_{min} = \overline{w}\sqrt{\frac{k \cdot |E_G|}{|S| \cdot \overline{deg} \cdot NLED_{v_0}}}$ and
- for the Binary Method: $e_{min} = \text{Binary-}e_{min}(E_G, |S|, k)$

6 Conclusions

We have presented methods to estimate the number of vertices and edges that are lying within a distance e from a vertex in a spatial network. First, as a baseline method, we introduced a naive approach, the MDS-Grid Method, which gives efficient estimations only for small graphs, at the expense of large space and time requirements. Then, three different methods have been also examined: a) the Global Parameters Estimation method, (b) the Local Densities method, and (c) the Binary Encoding method. We have given analytic solutions for all methods, as well as specific space and time bounds. We have applied the proposed methods in both synthetic and real spatial networks, and we have demonstrated performance results. In conclusion: (a) the Global Parameters Estimation method performs efficient estimations in regular uniform or almost regular uniform graphs, (b) the Local Densities methods offer better estimations in non-regular uniform graphs, (c) the Binary Encoding method offers the most accurate estimations in all graph types, with small adjustable space requirements.

An interesting direction for future work is to study the application of the proposed techniques in other network types (e.g., directed networks) and for more complex queries (e.g., spatial joins).

References

[1] A. Belussi and C. Faloutsos, "Estimating the Selectivity of Spatial Queries Using the (Correlation) Fractal Dimension", *Proceedings 21st International Conference on Very Large Data Bases (VLDB)*, pp.299-310, Zurich, Switzerland, 1995.

[2] I. Borg and P. Groenen: "*Modern Multidimensional Scaling*", Springer, 1997.

[3] T.F. Cox and M.A.A. Cox: "*Multidimensional Scaling*", Chapman and Hall, 1994.

[4] J. Chen, X. Meng, Y. Guo and S. Grumbach, H. Sun: "Modeling and Predicting Future Trajectories of Moving Objects in a Constrained Network", *Proceedings 7th International Conference on Mobile Data Management (MDM)*, p.156, Nara, Japan, 2006.

[5] C. Faloutsos, B. Seeger, A. Traina, and C. Traina: "Spatial Join Selectivity Using Power Laws", *Proceedings ACM International Conference on Management of Data (SIGMOD)*, pp.177-188, Dallas, Texas, 2000.

[6] A.V. Goldberg, H. Kaplan and R. Werneck: "Reach for A*: Efficient Point-to-Point Shortest Path Algorithms", Microsoft Research, Technical Report MSR-TR-2005-132, 2005.

[7] J.C. Gower: "Some Distance Properties of Latent Root and Vector Methods Used in Multivariate Analysis", *Biometrika*, Vol.53, pp.325-338, 1966.

[8] S. Gupta, S. Kopparty and C. Ravishankar: "Roads, Codes, and Spatiotemporal Queries", *Proceedings 23rd ACM Symposium on Principles of Database Systems (PODS)*, pp.115-124, Paris, France, June 2004.

[9] M. Hadjieleftheriou, G. Kollios and V. Tsotras: "Performance Evaluation of Spatiotemporal Selectivity Estimation Techniques", *Proceedings 15th International Conference on Scientific and Statistical Database Management (SSDBM)*, pp.202-211, Cambridge, MA, 2003.

[10] J. Hwang, S. Lay and A. Lippman: "Nonparametric Multivariate Density Estimation: a Comparative Study", 1994.

[11] C.S. Jensen, J. Kolarvr, T.B. Pedersen and I. Timko: "Nearest Neighbor Queries in Road Networks", *Proceedings 11th ACM International Symposium on Advances in Geographic Information Systems (GIS)*, pp.1-8, New Orleans, LO, 2003.

[12] M.C. Jones: "Simple Boundary Correction for Kernel Density Estimation", *Statistics and Computing*, Vol.3, pp.135-146, 1993.

[13] M. Klusch, S. Lodi and G. Moro: "Distributed Clustering Based on Sampling Local Density Estimates", *Proceedings 18th International Joint Conference on Artificial Intelligence (IJCAI)*, pp.485-490, Acapulco, Mexico, 2003.

[14] J.B. Kruskal: "Multidimensional Scaling by Optimizing Goodness of Fit to a Nonmetric Hypothesis", *Psychometrika*, Vol.29, pp.1-27, 1964.

[15] D. Papadias, J. Zhang and N. Mamoulis: "Query Processing in Spatial Network Databases", *Proceedings 29th International Conference on Very Large Databases (VLDB)*, pp.802-813, Berlin, Germany, 2003.

[16] R-tree Portal, `http://www.rtreeportal.org/main.html`.

[17] J. Sankaranarayanan, H. Alborzi and H. Samet: "Efficient Query Processing on Spatial Networks", *Proceedings 13th ACM International Symposium on Geographic Information Systems (GIS)*, pp.200-209, Bremen, Germany, 2005.

[18] E. Schikuta: "Grid-Clustering: an Efficient Hierarchical Clustering Method for Very Large Data Sets", *Proceedings 13th International Conference on Pattern Recognition (ICPR)*, pp.101-105, Banff, Canada, 1996.

[19] D.W. Scott: "*Multivariate Density Estimation: Theory, Practice and Visualization*", Wiley, 1992.

[20] G.A.F. Seber: "*Multivariate Observations*", Wiley, 1984.

[21] C. Sevcik: "On Fractal Dimension of Waveforms", *Chaos, Solitons and Fractals*, pp.579-580, 2006.

[22] C. Sevcik: "A Procedure to Estimate the Fractal Dimension of Waveforms", *Complexity International*, Vol.5, 1998.

[23] X. Shen and S. Agrawal: "Kernel Density Estimation for an Anomaly Based Intrusion Detection System", *Proceedings International Conference on Machine Learning; Models, Technologies and Applications (MLMTA)*, pp.161-167, Las Vegas, NV, 2006.

[24] B.W. Silverman: "*Density Estimation for Statistics and Data Analysis*", Chapman and Hall, 1986.

[25] Y. Tao, C. Faloutsos and D. Papadias: "Spatial Query Estimation without the Local Uniformity Assumption", *Geoinformatica*, Vol.10, No.3, pp.261-293, 2006.

[26] Y. Theodoridis and T. Sellis: "A Model for the Prediction of R-tree Performance", *Proceedings 15th ACM Symposium on Principles of Database Systems (PODS)*, pp.161-171, Montreal, Canada, 1996.

[27] M. Thorup: "Integer Priority Queues with Decrease Key in Constant Time and the Single Source Shortest Paths Problem", *Proceedings 35th Annual ACM Symposium on Theory of Computing (STOC)*, pp.149-158, San Diego, CA, 2003.

[28] E. Tiakas, A.N. Papadopoulos, A. Nanopoulos, Y. Manolopoulos: "Selectivity Estimation in Spatial Networks", *Proceedings of the 23rd ACM Symposium on Applied Computing (ACM SAC 2008)*, Track: Advances in Spatial and Image-Based Information Systems, Fortaleza, Ceara, Brazil, March 16-20, 2008.

[29] L.R. Tucker: "A Method for the Synthesis of Factor Analysis Studies", Technical Report 984, Department of the Army, 1951.

[30] M.P. Wand and M.C. Jones: "*Kernel Smoothing*", Chapman and Hall, 1995.

[31] D.G. Weeks and P.M. Bentler: "A Comparison of Linear and Monotone Multidimensional Scaling Models", *Psychological Bulletin*, Vol.86, pp.349-354, 1979.

Appendix

Theorem 6.1. *Let G be an almost regular uniform spatial network, and also let:*

1. *the selected starting node v_0 to belong in a region of G defined by its central node v_c (at this point, as central node we define the median node of the shortest path with length D_G, where D_G is the diameter of the graph), and a network distance equal to $\frac{D_G}{4}$, and*
2. *the desired range distance e to lie in the interval $[0, \frac{D_G}{4}]$.*

Then, the estimator for the number of nodes $N(v_0, e)$:

$$\widetilde{N}(v_0, e) = \widetilde{N}(e) = \frac{\overline{deg}}{2} \cdot \frac{e}{\overline{w}} \cdot \left(\frac{e}{\overline{w}} + 1\right) + 1$$

have estimation error $Error[\widetilde{N}]$ which satisfy the following upper bound:

$$Error[\widetilde{N}] \leq \max\left\{ \left| \frac{\lceil \overline{deg} \rceil + 1}{\overline{deg}(1+a)} - 1 \right|, \left| \frac{\lfloor \overline{deg} \rfloor - 1}{\overline{deg}(1-a)} - 1 \right| \right\}$$

In addition, if the average degree $\overline{deg}$ is close tp 4 (with the same deviation a of the almost regular uniform definition of G),then the estimator for the number of edges $E(v_0, e)$:

$$\widetilde{E}(v_0, e) = \widetilde{E}(e) = \overline{deg} \cdot \frac{e^2}{\overline{w}^2}$$

have estimation error $Error[\widetilde{E}]$ *which satisfy the following upper bound:*

$$Error[\widetilde{E}] \leq \max\left\{ \left| \frac{\lceil \overline{deg} \rceil + 1}{\overline{deg}(1+a)^2} - 1 \right|, \left| \frac{\lfloor \overline{deg} \rfloor - 1}{\overline{deg}(1-a)^2} - 1 \right| \right\}$$

Proof. If G is almost regular uniform, then its edge weights and node degrees must satisfy the following:

$$(1-a)\overline{w} \leq w(v_i, v_j) \leq (1+a)\overline{w}, \quad \forall (v_i, v_j) \in E_G$$
$$deg(v_i) \in \{\lfloor \overline{deg} \rfloor - 1, \lfloor \overline{deg} \rfloor, \lceil \overline{deg} \rceil, \lceil \overline{deg} \rceil + 1\}, \quad \forall v_i \in V_G$$

where a is a small positive real number (e.g., $0 < a < 0.2$).

As the edge weights of G fall into the interval $[(1-a)\overline{w}, (1+a)\overline{w}]$, and the node degrees can be any of the four integers: $\lfloor \overline{deg} \rfloor - 1, \lfloor \overline{deg} \rfloor, \lceil \overline{deg} \rceil, \lceil \overline{deg} \rceil + 1$, it is evident that the largest deviations from averages will produce the maximum estimation errors. These worst cases are:

1. for edges and nodes contained in the region of interest, all edge weights are equal to $(1-a)\overline{w}$ and all node degrees are equal to $\lfloor \overline{deg} \rfloor - 1$, and
2. for edges and nodes contained in the region of interest, all edge weights are equal to $(1+a)\overline{w}$ and all node degrees are equal to $\lceil \overline{deg} \rceil + 1$.

Thus, let $\check{G}$ and $\hat{G}$ be the graphs that satisfy the former and latter case, respectively. As on these graphs the properties $d(v_0, v_c) \leq \frac{D_G}{4}$ and $0 \leq e \leq \frac{D_G}{4}$ continue to hold, we can apply Lemmas 3.1-3.2 to derive the numbers of nodes lying in the region defined by node v_0 and the network distance e:

$$\check{N}(v_0, e) = \check{N}(e) = \frac{\lfloor \overline{deg} \rfloor - 1}{2} \cdot \frac{e}{(1-a)\overline{w}} \cdot \left(\frac{e}{(1-a)\overline{w}} + 1 \right) + 1$$

$$\hat{N}(v_0, e) = \hat{N}(e) = \frac{\lceil \overline{deg} \rceil + 1}{2} \cdot \frac{e}{(1+a)\overline{w}} \cdot \left(\frac{e}{(1+a)\overline{w}} + 1 \right) + 1$$

Therefore, the node estimation error satisfies the following property:

$$0 \leq Error[\widetilde{N}] \leq \max\left\{ \frac{\left| \check{N}(v_0, e) - \widetilde{N}(v_0, e) \right|}{\widetilde{N}(v_0, e)}, \frac{\left| \hat{N}(v_0, e) - \widetilde{N}(v_0, e) \right|}{\widetilde{N}(v_0, e)} \right\}$$

The node estimation error bounds can be computed as:

$$\frac{\left| \check{N}(v_0, e) - \widetilde{N}(v_0, e) \right|}{\widetilde{N}(v_0, e)} = \frac{\left| \left[\frac{\lfloor \overline{deg} \rfloor - 1}{2} \cdot \frac{e}{(1-a)\overline{w}} \cdot \left(\frac{e}{(1-a)\overline{w}} + 1 \right) + 1 \right] - \left[\frac{\overline{deg}}{2} \cdot \frac{e}{\overline{w}} \cdot \left(\frac{e}{\overline{w}} + 1 \right) + 1 \right] \right|}{\frac{\overline{deg}}{2} \cdot \frac{e}{\overline{w}} \cdot \left(\frac{e}{\overline{w}} + 1 \right) + 1}$$

$$= \frac{\left| \left[\frac{\lfloor \overline{deg} \rfloor - 1}{2} \cdot \frac{e}{(1-a)\overline{w}} \cdot \left(\frac{e}{(1-a)\overline{w}} + 1 \right) \right] - \left[\frac{\overline{deg}}{2} \cdot \frac{e}{\overline{w}} \cdot \left(\frac{e}{\overline{w}} + 1 \right) \right] \right|}{\frac{\overline{deg}}{2} \cdot \frac{e}{\overline{w}} \cdot \left(\frac{e}{\overline{w}} + 1 \right) + 1}$$

$$\leq \left| \frac{\left[\frac{\lfloor \overline{deg} \rfloor - 1}{2} \cdot \frac{e}{(1-a)\overline{w}} \cdot \left(\frac{e}{(1-a)\overline{w}} + 1 \right) \right] - \left[\frac{\overline{deg}}{2} \cdot \frac{e}{\overline{w}} \cdot \left(\frac{e}{\overline{w}} + 1 \right) \right]}{\frac{\overline{deg}}{2} \cdot \frac{e}{\overline{w}} \cdot \left(\frac{e}{\overline{w}} + 1 \right)} \right|$$

$$= \left| \frac{\left(\lfloor \overline{deg} \rfloor - 1 \right) \cdot \left(\frac{e}{(1-a)\overline{w}} + 1 \right)}{\overline{deg} \cdot (1-a) \cdot \left(\frac{e}{\overline{w}} + 1 \right)} - 1 \right| =$$

$$= \left| \frac{\left(\lfloor \overline{deg} \rfloor - 1\right)}{\overline{deg} \cdot (1-a)^2} \cdot \left(1 - \frac{a}{\frac{e}{\overline{w}} + 1}\right) - 1 \right| \leq \left| \frac{\lfloor \overline{deg} \rfloor - 1}{\overline{deg}(1-a)} - 1 \right|$$

Similarly we have that:

$$\frac{\left|\hat{N}(v_0, e) - \widetilde{N}(v_0, e)\right|}{\widetilde{N}(v_0, e)} \leq \left| \frac{\lceil \overline{deg} \rceil + 1}{\overline{deg}(1+a)} - 1 \right|$$

Then, we have:

$$0 \leq Error[\widetilde{N}] \leq \max\left\{ \left| \frac{\lceil \overline{deg} \rceil + 1}{\overline{deg}(1+a)} - 1 \right|, \left| \frac{\lfloor \overline{deg} \rfloor - 1}{\overline{deg}(1-a)} - 1 \right| \right\}$$

Based on the discussion in the text regarding the accuracy of edge estimation, if the average degree $\overline{deg}$ is close to 4 (with the same deviation a, e.g., $0 < a < 0.2$), then the estimator for the number of edges of the graph $\check{G}$ and $\hat{G}$ are:

$$\check{E}(v_0, e) = \check{E}(e) = \left(\lfloor \overline{deg} \rfloor - 1\right) \cdot \frac{e^2}{(1-a)^2 \overline{w}^2}$$

$$\hat{E}(v_0, e) = \hat{E}(e) = \left(\lceil \overline{deg} \rceil + 1\right) \cdot \frac{e^2}{(1+a)^2 \overline{w}^2}$$

Then, the edge estimation error satisfies the following property:

$$0 \leq Error[\widetilde{E}] \leq \max\left\{ \frac{\left|\check{E}(v_0, e) - \widetilde{E}(v_0, e)\right|}{\widetilde{E}(v_0, e)}, \frac{\left|\hat{E}(v_0, e) - \widetilde{E}(v_0, e)\right|}{\widetilde{E}(v_0, e)} \right\}$$

Therefore, the edge estimation error bounds have as follows:

$$\frac{\left|\check{E}(v_0, e) - \widetilde{E}(v_0, e)\right|}{\widetilde{E}(v_0, e)} = \left| \frac{\check{E}(v_0, e)}{\widetilde{E}(v_0, e)} - 1 \right| = \left| \frac{\left(\lfloor \overline{deg} \rfloor - 1\right) \cdot \frac{e^2}{(1-a)^2 \overline{w}^2}}{\overline{deg} \cdot \frac{e^2}{\overline{w}^2}} - 1 \right| = \left| \frac{\lfloor \overline{deg} \rfloor - 1}{\overline{deg}(1-a)^2} - 1 \right|$$

$$\frac{\left|\hat{E}(v_0, e) - \widetilde{E}(v_0, e)\right|}{\widetilde{E}(v_0, e)} = \left| \frac{\hat{E}(v_0, e)}{\widetilde{E}(v_0, e)} - 1 \right| = \left| \frac{\left(\lceil \overline{deg} \rceil + 1\right) \cdot \frac{e^2}{(1+a)^2 \overline{w}^2}}{\overline{deg} \cdot \frac{e^2}{\overline{w}^2}} - 1 \right| = \left| \frac{\lceil \overline{deg} \rceil + 1}{\overline{deg}(1+a)^2} - 1 \right|$$

Thus, we have:

$$0 \leq Error[\widetilde{E}] \leq \max\left\{ \left| \frac{\lceil \overline{deg} \rceil + 1}{\overline{deg}(1+a)^2} - 1 \right|, \left| \frac{\lfloor \overline{deg} \rfloor - 1}{\overline{deg}(1-a)^2} - 1 \right| \right\}$$

and this completes the proof. □

As an example of checking these bounds, we can take the UN network, which is an almost regular uniform graph with $a = 0.2$ and $\overline{deg} = 3.992$. According to the theorem, we have:

$$Error[\widetilde{N}] \leq \max\left\{ \left| \frac{\lceil 3.992 \rceil + 1}{3.992(1+0.2)} - 1 \right|, \left| \frac{\lfloor 3.992 \rfloor - 1}{3.992(1-0.2)} - 1 \right| \right\} = \max\{0.044, 0.374\} = 37.4\%$$

$$Error[\widetilde{E}] \leq \max\left\{ \left| \frac{\lceil 3.992 \rceil + 1}{3.992(1+0.2)^2} - 1 \right|, \left| \frac{\lfloor 3.992 \rfloor - 1}{3.992(1-0.2)^2} - 1 \right| \right\} = \max\{0.130, 0.217\} = 21.7\%$$

Indeed, Figure 11 shows that for the whole e-range spectrum, the node and edge estimation errors remain under 20% for the Global Method's formulae.