Heuristic Coarsening for Generating Multiscale Transport Networks

Krishnakumari, Panchamy; Cats, Oded; van Lint, Hans

DOI
10.1109/TITS.2019.2912430

Publication date
2019

Document Version
Final published version

Published in
IEEE Transactions on Intelligent Transportation Systems

Heuristic Coarsening for Generating Multiscale Transport Networks

Panchamy Krishnakumari, Oded Cats, and Hans van Lint

***Abstract*— Graphs at different scales are essential tools for many transportation applications. Notwithstanding their relevance, these graphs are created and maintained manually for most applications, in both research and practice. In this paper, we develop a heuristic method for automatically generating multiscale graph representations without significantly compromising their topological properties. This makes the resulting graphs widely applicable. The method is demonstrated on the open street map network of Amsterdam with four different application cases. Various graph metrics are used for evaluating the performance of coarsening on the topological characteristics of the network. Our results show that the method is able to successfully reduce the Amsterdam network by up to 96% of its original size at a computation time of no more than 15 min with a limited loss of information, indicated by the preservation of key network characteristics. For example, the method maintains trip length distributions and limits the maximum shortest path deterioration between any major origin and destination nodes to no more than 0.025% for the coarsest graph. Moreover, by setting its parameters it can cater for preservation of important network elements or entire sub-networks, which is of special importance in multiscale traffic modeling and simulation. The versatility of the algorithm—in contrast to algorithms dedicated to for example traffic assignment applications—makes it useful for a wide range of applications within the transportation domain and beyond. To support further research an open-source implementation of the algorithm is made available.**

***Index Terms*— Multiscale graph, heuristic, coarsening, geographic data, opensource, transport networks.**

I. Introduction

DIRECTED graphs are vital tools in many areas of transportation science and practice. Particularly for the design and study of ITS; accurate graph representations at the appropriate level of detail are of quintessential importance. There are many readily available detailed directed graph representations. These representations are based on structured, reusable and standardized geographic and dynamic data such as open street map (OSM), and dedicated maps maintained by public administrations and road and rail operators. However, multiscale representations of these networks are more difficult to come by, despite their relevance.

Manuscript received October 23, 2018; revised March 11, 2019 and April 10, 2019; accepted April 14, 2019. This work was supported by the SETA Project funded by the European Union's Horizon 2020 Research and Innovation Program under Grant 688082. The Associate Editor for this paper was S. C. Wong. *(Corresponding author: Panchamy Krishnakumari.)*

The authors are with the Department of Transport and Planning, Delft University of Technology, 2628 Delft, The Netherlands (e-mail: p.k.krishnakumari@tudelft.nl).

Digital Object Identifier 10.1109/TITS.2019.2912430

Multiscale graph decomposition has been studied extensively in different fields such as scientific computing, gaming, Very Large Scale Integration (VLSI) system design, to name a few, using methods based on random walks, diffusion maps, spectral graph theory and various coarsening schemes [1]–[3]. In transportation, studies involving graph decompositioning focus mainly on graph partitioning problems for speeding up shortest path routing [3]–[6], and applications in the context of traffic assignment and/or equilibrium sensitivity analysis [7]–[11]. However, there are many other transport applications that may benefit from *consistent* network representations at different levels of scale, obtained from either detailed graph data (e.g. OSM) or coarse schematics. Examples include multiscale modeling and simulation [12], [13]; traffic estimation and prediction [14]–[18]; and even public transport service network analysis [19] to name but a few. In fact, there are very few areas within transportation science, where no schematic graph representation of either the physical or service network is needed. In practice today, such simplified schematic representations are often created and maintained manually, which is time consuming and error prone.

Given the wide range of applications for transportation network analysis, automation of the process of generating such coarser graphs from whatever data available offers scientists and practitioners large benefits in terms of effort spent. This calls for the development of a generic simple solution for generating and maintaining a set of mutually consistent and accurate directed graphs on the basis of the available geographic data.

Definition 1: A multiscale graph is a set of increasingly coarser graphs G_i, G_{i+k},..., k = 1,2,3; representing the same transport infrastructure (or service network).

We propose that a consistently coarsened graph G_{i+1} with respect to some finer base graph G_i should match the following criteria:

- G_{i+1} has considerably fewer links and nodes than G_i
- G_{i+1} preserves important global topological characteristics of G_i (connectivity, shortest path distribution, diameter, total network length, centrality)
- G_{i+1} preserves important domain specific link and node attributes encoded in (or defined on) G_i
- G_{i+1} preserves consistent and accurate local (dynamic) topological attributes of G_i such as the shortest paths between origins and destinations (at approximately the same locations)

Note that where we use the words "preserve" (certain properties), one may also read "gracefully degrades", in the sense

that in some cases, some degradation of information density is inevitable when cutting out nodes and/or links. We return to this point in more detail in the validation experiments we provide.

To this end, we propose a heuristic coarsening technique based on topological and/or data-driven information of the directed graphs. A constrained version of this coarsening approach using data-driven parameter is briefly noted in [17]. Here, we present a more detailed and generic framework that supports more widespread application. What makes our approach different from existing coarsening techniques tailored for specific transport applications—e.g. routing and assignment, which we discuss below—is that it provides a generic and flexible tool to simplify large transport networks into consistent coarser ones for many applications, ranging from topological analysis, modeling, simulation or visualisation, to name just a few. In our research lab this method has significantly reduced the effort in generating graphs for these common research tasks, and to the best of our knowledge no such generic method has been reported in the transportation literature and/or made available in code. We demonstrate the framework for four such applications on the large scale network of Amsterdam city. We use readily available topology information like the length, type, node-density, or other physical attributes of the graph to assign the weights and define the coarsening rules. The detailed graph representation and the physical attributes are obtained from Open Street Map (OSM), an open-source geographic data source. To support the research community in using and further developing efficient tools for graph coarsening we offer an open-source version of the code that implements our framework.[1]

The paper is organized as follows: Section II first overviews the basics of network coarsening, using related work in (mostly) disciplines other than transportation. In section III we then discuss the proposed coarsening framework and the algorithms that will be applied to transportation networks. In section IV we discuss the (Amsterdam) data; and the methods and performance indicators to assess how well our approach succeeds in generating consistent coarsened graph representations of the Amsterdam, the Netherlands. We quantitatively and qualitatively discuss the results in section V and conclude the paper in section VI.

II. Related Works

Within transportation, a limited number of studies report explicit algorithmic work on graph coarsening. In [8] and [10] a bush-based approach is proposed for replacing a regional network with a smaller one, containing all of the sub-network, and zones. Artificial arcs are created to represent "all paths" between each origin and sub-network boundary node, under the assumption that the set of equilibrium routes does not change. Similarly, [9] and [11] present method(s) for network aggregation under Stochastic User Equilibrium (SUE), using sensitivity analysis, in which the measure for assessing the resulting coarse network representation is based on how well perturbations in either demand or supply characteristics (i.e. changes in the OD matrix and/or changes in the link cost functions respectively) affect the result of the assignment. These methods are insightful, but based on a huge set of assumptions specific (and relevant) to the assignment problem, but not to other transportation problems. This hinders their relevance and transferability to other application domains. A second and related class of transportation problems for which graph coarsening plays an important role is speeding up shortest path routing algorithms [3]–[6], [20]. Bast *et al.* [21] gives an extensive overview of the multilevel methods for routing in transport networks. They conclude out that there are not many studies available within the transportation domain that discuss how—for a much broader range of applications other than assignment and speeding up routing problems—the topological characteristics of multiscale graphs differ with respect to the original fine-scaled graph. There is, however, a rich body of work available in other domains. Here, we present an abridged overview on coarsening research that is directly relevant for this work.

Multilevel methods were introduced during the 1990's to improve efficiency and quality of combinatorial optimisation problems [2]. Multilevel based algorithms try to solve complex problems by creating a hierarchy of problems that represent the original problem with fewer degrees of freedom. This process is coined coarsening. These hierarchies at different scales can be sequentially projected back to reconstruct the original problem space, known as uncoarsening. The coarsening and uncoarsening stages together constitute the multilevel framework. There are a couple of papers that provide an overview of multilevel techniques [22], [23]. In this work, we are only interested in the coarsening phase of the framework. Coarsening can be broadly classified into two types - strict and weighted coarsening. In strict coarsening, nodes are aggregated together to form a single node in the "coarsened space". The nodes in the coarsened space are called aggregates [2]. In weighted coarsening, each node is divided into fractions and these fractions can belong to different aggregates in the coarsened space [24]. More details on the principal differences between these two methods in graph terms can be found in [23].

Multilevel algorithms have been used in many disciplines including games [25], [26], mechanical engineering [27], infectious disease spread studies [28] and graph optimisation problems [23]. The graph partitioning problems and graph optimisation applications within transportation that focus on speeding up shortest path routing algorithms [3]–[5], [20], [21], [29]–[31] typically use strict coarsening for generating the hierarchies. That most multilevel methods for transport networks use hierarchical techniques makes sense, since road networks are inherently hierarchical. This was first fully exploited in the highway hierarchies (HHs) [4] method. The highway hierarchies contains two main building blocks - edge reduction and node reduction. Edge reduction preserves the edges in the middle of long distance paths and node reduction contracts nodes of degree one and two (i.e. nodes that only connect one or two adjacent links).

A simpler version of HHs are so-called contraction hierarchies (CHs), introduced by Geisberger *et al.* [3], [32], which

[1] https://github.com/Panchamy/Heuristic-Coarsening/wiki

are among the most effective (shortest route) speedup techniques. In general, coarsening techniques work by replacing edges in the graph with so-called shortcuts. In CHs, the shortcuts are added iteratively by contracting nodes following a given order of importance. The node ordering eliminates one of the major drawbacks of classical methods - the unpredictability of the contraction results. The main reason for this can be attributed to the random choice of nodes for the coarse level graph in classic methods [2]. Edge reduction is used in HHs to minimise the explosion of average node degree in the coarsened network but in CHs, this shortcoming is eliminated using a more sophisticated node contraction. Node contraction in CHs adds shortcuts only if shortest paths are preserved in the coarse scale after each node contraction. However, checking if the shortest path is preserved is time consuming. There are various solutions to speed up this process including limiting the space for shortest path search [3], using GPU [33] and customizable contraction hierarchies [26].

All these studies are based on graph methods that have not (yet) been explored in the traffic domain other than for routing applications. In this paper, we seek a (heuristic) approach for network coarsening that can be used (insofar possible) in most transportation applications where graph coarsening might be useful. This method should offer a generic mechanism to assess the quality of the procedure based on topological information and/or data available in the application at hand. Based on the simplicity and success of CHs, we propose a heuristic approach with some of the building blocks of CHs—node ordering and node contraction. In [17], we briefly show how a constrained version of CHs can be easily used for network complexity reduction for traffic predictions. In the current contribution we further develop, formalize, apply and test the proposed approach to provide a more generic heuristic framework based on CHs that can be deployed in various applications.

III. Heuristic Coarsening Framework for Multiscale Graph Generation

The general idea of coarsening is that, given graph G with n nodes, a more compact graph with a smaller number of nodes can be found which yields a good representation of the original graph. The multiscale graph G_{i+1} is constructed from the previous finer scale graph G_i by collapsing together the nodes and edges that have similar matching criteria. The matching can be computed in different ways, for example, by using aggregates [2]; by considering dominant route flows [34]; or based on node density [35]. In this work, the matching is based on the edge difference or variance of the edge weights. On top of the building blocks of CHs, we also use pruning to further reduce the network. This section will detail the steps required to derive these multiscale graphs. The coarsened graph can be constructed using the following four steps [2]. Note that each step may be detailed according to application-specific requirements or constraints.

1) Assign weights to the links in the directed graph;
2) Prioritize the nodes so that they can be removed in a strict order for generating the next coarsened level;
3) Determine contraction and pruning decision rules based on the edges weights, and;
4) Determine the new weights of the links for the coarse graph(for potentially a next iteration).

TABLE I
Graph Notations. Examples Are Based on Figure 1

$G_i(V,E)$	Fine network with set of nodes V and set of ordered pair of edges or links E
$G_{i+1}(V',E')$	Coarse network with the updated nodes V' $\subset$ V and the updated edge set E'
w_{uv}	weight of edge (u,v). eg: $w_{uv}=2$
$N(v)$	neighboring links of node v. eg: $N(v)=\{(u,v),(v,w),(v,x)\}$
$N^-(v)$	incoming links of node v. eg: $N^-(v)=\{(u,v)\}$
$N^+(v)$	outgoing links of node v. eg: $N^+(v)=\{(v,w),(v,x)\}$
$\delta(v)$	$\lvert N(v)\rvert$, degree of node v which is the total number of incoming and outgoing links of node v. eg: $\delta(v)=3$
$\delta^-(v)$	in-degree of node v. eg: $\delta^-(v)=1$
$\delta^+(v)$	out-degree of node v. eg: $\delta^+(v)=2$

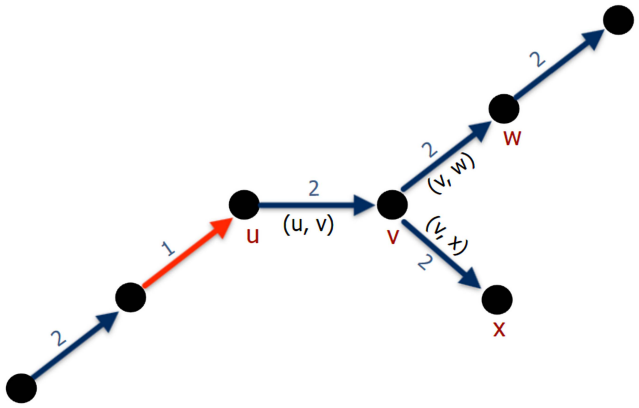


Fig. 1. Example network.

Notation: We use the standard notations used in graph theory as detailed in Table I, illustrated using the example network shown in Figure 1. Here, the graph $G=(V,E)$ is a weighted directed graph where V is the set of nodes and E is the set of ordered pairs of edges or links. The edge $(u,v)\in E$, in Figure 1, is an incoming link with respect to node v where v is the target node and u is the source node. (v,w) and (v,x) are the outgoing links of node v where v is the source node and w and x are the target nodes. Arbitrary edge weights of the example network are also indicated in Figure 1.

A. Step 1 - Assigning Edge Weights

Edge weights are an essential element in solving graph problems such as coarsening, partitioning, etc. The weight can correspond to link length, width, type characteristics such as the link flow, inductance (for electric applications) or speed (for transport applications). We propose a generic weight measure, w_{uv} for the link (u,v) in the form of a weighted average over the application-relevant edge weights:

$$w_{uv}=\sum_{i=1}^{n}\beta_i w_{uv}^i \tag{1}$$

where n is the number of attributes, β varies typically between 0 and 1 and reflects the influence of these attributes on the generic edge weight, and w^i_{uv} is the i^{th} attribute of the link (u, v). Clearly, the value of β may differ based on the application.

B. Step 2 - Ranking the Nodes

The order in which the nodes are removed is important for graph coarsening for computational reasons (only) [2]. In general, the seed nodes (nodes in the original graph considered for collapsing) are chosen randomly. In this work, we use a deterministic approach based on node ordering such that the nodes from the priority queue are contracted across the network in a uniform way, rather than contracting nodes randomly. For example, nodes can be ordered based on geographical scale (e.g. metropolitan areas; cities; neighbourhoods); traffic hierarchy function (freeways; motorways; main arterials; etc); spatial subdivision types such as grid-based [36] and polygon-based (e.g. clustering based on postal codes).

To illustrate this process, we use node degree (i.e. the number of edges connected to a node) as the decision rule for prioritising the nodes. The more neighbours the node has, the higher the rank, and the node will be contracted later. The underlying assumption is that a node that connects a lot of edges is likely to be more important for the transportation network and flow distribution—at least locally. Thus, the nodes are contracted by increasing order of node degree. Suppose, $(u, v) \in E$ where $u, v \in V$ then the rank of the nodes u and v will satisfy the following condition:

$$r(u) > r(v), \quad if \ \delta(u) > \delta(v) \tag{2}$$

where $\delta(u)$ and $\delta(v)$ are the degree of node u and v respectively. Thus, based on the contraction rule, v will be contracted before u. Node contraction affects the priorities of other nodes. Therefore, the priority queue is rebuilt after each node collapse. Since this process can become computationally expensive, we have implemented an iterative approach instead of re-evaluating the priorities, which is more efficient and provides robust results. In the iterative approach, we evaluate the priority once at the beginning of the iteration and collapse the nodes according to this queue. The neighbours of the nodes are updated at the end of the iteration. The iteration ends when all the nodes are visited at least once for collapsing consideration. The method converges when the iteration provides the same result as the previous iteration.

C. Step 3 - Defining the Contraction and Pruning Rules

Once the nodes are sorted in increasing order, the contraction rules based on edge weights for collapsing them are defined. When a node is collapsed, its neighbouring links are joined together to form new links. Figure 2 presents some examples of different cases of node contraction. If the node collapse results in the same or even a larger number of links than before its collapse as shown in case (6) in Figure 2, there is no reason to collapse that node. Collapsing nodes without any regulation can lead to explosion of average node degree in the coarse level graph [3]. Therefore, a criterion

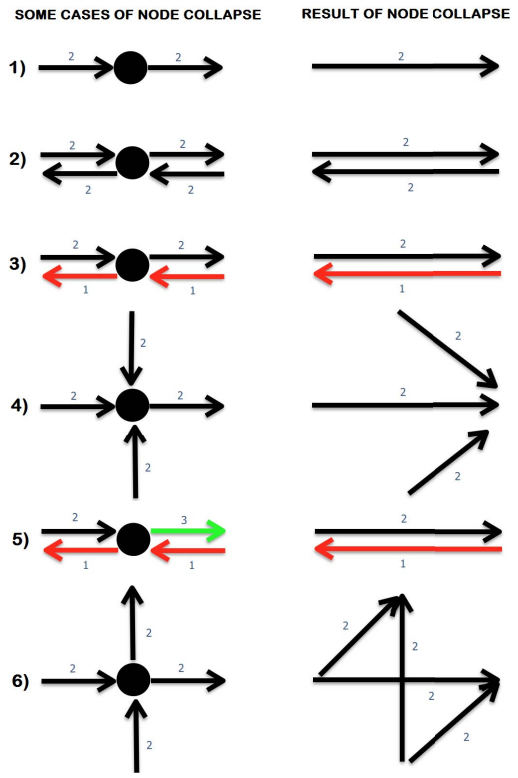


Fig. 2. Examples of node collapse.

TABLE II
DECISION RULES

		Node Collapse Rules
$c_1(v)$	$=$	$\begin{cases} 1, & \text{if } \delta(v') < \delta(v) \text{ where } v \in V \ \& \ v' \in V' \\ 0, & \text{otherwise} \end{cases}$
$c_2(v)$	$=$	$\prod \begin{cases} 1, & \text{if } \sigma^2(w_{iv}, w_{vj}) \leq \rho \\ 0, & \text{otherwise} \end{cases} \forall \begin{matrix} i \in N^-(v) \\ j \in N^+(v) \end{matrix}$
		Node Deletion Rules
$c_3(v)$	$=$	$\begin{cases} 1, & \text{if } \delta^+(v) = 0 \text{ or } \delta^-(v) = 0 \\ 0, & \text{otherwise} \end{cases}$
$c_4(u, v)$	$=$	$\begin{cases} 1, & \text{if } u = v \\ 0, & \text{otherwise} \end{cases} \forall \ (u, v) \in E$

$c_1(v)$ (Table II) is set to decide if the contraction of a given node will contribute to a reduction in network complexity. Note that in case the application requires a coarse graph with fewer nodes but the number of links is not a priority, this constraint can easily be adjusted accordingly.

The edge difference is used to define the next rule $c_2(v)$ for inclusion or exclusion of that node for contraction (Table II). This rule is checked for each of the incoming-outgoing link pairs of the given node v. A lower threshold, ρ, implies a tighter constraint on the node collapse. The edge difference or variance of the edge weights w_{iv} and w_{vj} of node v, defined

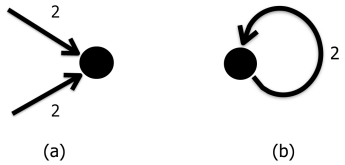


Fig. 3. Example cases of pruning (a) Dead-ends (b) Self-loops.

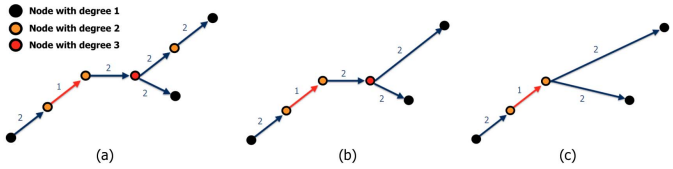


Fig. 4. Node collapse results in the example network with ρ 0. (a) Nodes are marked based on their rank. (b) A node with degree 2 is collapsed. (c) A node with degree 3 is collapsed.

in equation (3), is used as the matching criterion.

$$\sigma^2(w_{iv}, w_{vj}) = |w_{iv} - \mu|^2 + |w_{vj} - \mu|^2, \quad \text{where}$$
$$\mu = \frac{w_{iv} + w_{vj}}{2}, \tag{3}$$

This is based on the idea that nodes should not be collapsed if they serve as the connection between two inherently different links. For example, a node that is connecting a highway and a city road is topologically important and results in small edge difference. If the ρ is set to 0 then this node would not be collapsed. If the ρ is set higher then the nodes that connect links with a smaller weight difference will be collapsed. For example, a node that connects a highway and a service road is hierarchically and topologically informative and hence should not be collapsed. Setting a proper ρ can prevent this. In the case (5) in Figure 2, if the ρ is set to 0, the node will not be collapsed as the edge difference is not 0. (i.e. there is a change in hierarchical level)

The most expensive computation for most of the methods mentioned in the literature, including CHs, relates to checking whether the shortest path is preserved after each node collapse [3]. This condition is not included in our heuristic approach under the premise that if the node collapse is performed according to the proposed method, there will only be minimal deterioration in the shortest path, which is acceptable for most applications. *In section V, we will explicitly examine the validity of this assumption.*

Collapsing nodes can only reduce the complexity of the network to the highest edge difference threshold. To further reduce the network, pruning can be performed. Pruning refers to removing unimportant (in an application-specific sense) nodes or links from the network, instead of collapsing them. Depending on the application, pruning can be allowed or disabled. In this work, pruning is used for removing dead-ends (nodes without either incoming or outgoing links) and self-loops in the graph. Examples of these two cases are illustrated in Figure 3. Given that pruning is allowed, two conditions are defined to identify the dead ends and self-loops - $c_3(v)$ and $c_4(u, v)$, respectively.

D. Step 4 - Assigning Weights to New Links

Assigning weights to the new links of the coarsened graph is the final step in the multiscale graph generation algorithm. The new edge weight is a function of the weights of the edges that are joined to make the new edge. Suppose the node v in Figure 1 satisfies both criteria $c_1(v)$ and $c_2(v)$, then the incoming-outgoing link pair (u, v, w) is joined to form a new directed link (u, w) and the weight of this link is determined as follows:

$$w_{uw} = f(w_{uv}, w_{vw}) \tag{4}$$

Depending on the edge weight, this function may represent any mathematical (e.g. logical or statistical) operation on the original weights. For example, if the edge weights represent the link length, the logical choice is a summation function. The same holds true if the edge weights represents costs or travel time. A common edge weight in different application domains is link capacity. To combine different link capacities, a minimum function is employed as illustrated in the examples in Figure 2. However, for traffic assignment applications, a minimum function might cause a reduction in overall network capacity. For this application, a stricter constraint with respect to pruning and edge difference combined with a maximum function might be more appropriate.

The pseudo-code for the heuristic coarsening is given in Algorithm 1. A step-by-step node collapse for the example network with pruning disabled is illustrated in Figure 4. The nodes are ranked based on their node degree. The new weights are computed using a minimum function. Figure 4(a) shows the graph with the ranked nodes. Since pruning is disabled, the node with degree 1 cannot be collapsed because of the initial stopping criterion $c_1(v)$. Figure 4(b) shows the result of the collapse of degree 2 nodes with the ρ set at 0. Lastly, the higher degree nodes that satisfy both the conditions are collapsed as shown in Figure 4(c).

IV. Experimental Setup

In this study, we present four application cases of the coarsening scheme. The applications illustrate various aspects of the algorithm and how restrictions can be added for different purposes. We study in detail whether the coarsening results satisfy the requirements of multiscale graphs proposed in Section I using several verification measures. Note that we do not claim these four cases provide conclusive evidence that under all application constraints the requirements in Section I are met. In this section, we explain the application cases; describe the data used; how the weights are assigned for the case study networks; and the verification measures.

A. Application Cases

Before describing the four cases, let us briefly mention that for each of these we need to set two general parameters associated with our method: ρ ($threshold$) and $pruning$. For $pruning$, there are only two possible values; either enabled (1) or disabled (0). The ρ value corresponds to the restriction

Algorithm 1 Heuristic Coarsening Approach

Function
  Input : Node list V, edge set E,iterations M, weights w, $pruning$ and ρ
  Output: Coarsened edge set E'
  $E' \leftarrow E, i \leftarrow 0, iter \leftarrow 0, flag \leftarrow 1$
  while *flag* $= 1$ *and iter* $< M$ **do**
    /* Step 2 - Node ordering */
    $E \leftarrow E', V' \leftarrow$ sorted V
    while $i \neq |V'|$ **do**
      $v \leftarrow V'[i]$
      /* Step 3 - Contraction rules */
      if $c_1(v) = 1$ **then**
        Find $N^-(v)$ and $N^+(v)$. Eg: $(u, v), (v, w), (v, x)$
        Pair up $\{N^-(v), N^+(v)\}$. Eg: $(u, v, w), (u, v, x)$
        if $c_2(v) = 1$ **then**
          $E' \leftarrow E' - [(u, v), (v, w), (v, x)]$
          $E' \leftarrow E' + [(u, w), (u, x)]$
          /* Step 4 - Assign weights to new links */
          $w_{uw} = f(w_{uv}, w_{vw})$, $w_{ux} = f(w_{uv}, w_{vx})$
      $i \leftarrow i + 1$
    /* Step 3 - Pruning rules */
    if *pruning is* $True$ **then**
      foreach $v \in V'$ **do**
        if $c_3(v) = 1$ **then**
          $E' \leftarrow E' - N(v)$
      foreach $(u, v) \in E'$ **do**
        if $c_4(u, v) = 1$ **then**
          $E' \leftarrow E' - (u, v)$
    $V' \leftarrow$ update neighbors of V'
    $iter \leftarrow iter + 1$
    if $|E| = |E'|$ **then**
      $flag \leftarrow 0$

on the edge weight difference and in most applications, the ρ values are bounded as the edge weights are bounded. In this work, we demonstrate the coarsening for two instances of ρ values - minimum and maximum ρ for all the applications. This will define the upper bound and lower bound of the coarsening results for a particular application case. So, for each application there are four scenarios - pruning [0, 1] and ρ $[minimum, maximum]$.

1) Application I - Maximum Network Reduction Possible Without Any Restrictions: For the first application, we coarsen a large scale network with the simple aim of reducing the network complexity as much as possible. This objective may, for example, arise when visualizing properties of the network in time-critical applications (websites, mobile apps, etc). Clearly, the trivial maximum possible reduction of a network is to reduce it to a single node. However, the aim of our coarsening is to reduce the number of nodes as much as possible while reducing the number of links according to the constraint $c_1(v)$. Further reduction of coarsened graph by relaxing this constraint will lead to an explosion of links. Therefore, the maximum possible reduction of a network, in our case, corresponds to the maximum reduction of links and this is bounded by two parameters - pruning and ρ.

2) Application II - Network Reduction Restricted Based on Node Type: The second application is a constrained version of the first where we try to coarsen the network while preserving all of the intersections. This case may, for example, arise when constructing a network model for traffic simulation with a focus on developing or ex ante evaluation of (coordinated) intersection control algorithms, or conversely, on driving behavioural models for conflict negotiation. An intersection is a node representing any kind of discontinuity such as a crossing, converging or diverging links, etc. In graph terms, we consider an intersection as any node with more than 1 outgoing link and 1 incoming link and also with $\rho = 0$ as edge difference is a form of discontinuity. The $c_1(v)$ is adjusted to represent this constraint as:

$$c_1(v) = \begin{cases} 1, & \text{if } \delta(v') < 2 \text{ where } v \in V \ \& \ v' \in V' \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

This aims at maximum network reduction while preserving the information of all discontinuous nodes.

3) Application III - Network Reduction Restricted Based on Area: The third application case pertains to having different scales within the same network, for example in case the study area is in fine detail whereas the area outside the study boundary is less detailed, which is particularly useful for hybrid modeling. By using an additional constraint for the nodes $c_5(v)$, we can create a subset of nodes as the exception node list to achieve this. This subset of nodes can be created manually or by defining a polygon boundary for the study area. In our case, we use a rectangular boundary for the study area defined as $[x_{min}, y_{min}, x_{max}, y_{max}]$, thus the constraint $c_5(v)$ is defined as:

$$c_5(v) = \begin{cases} 0, & \text{if } x_{min} < x_v < x_{max} \ \& \ y_{min} < y_v < y_{max} \\ 1, & \text{otherwise} \end{cases} \tag{6}$$

where x_v and y_v are the co-ordinates of the node v. Thus, we can create an exception list of nodes that are prohibited from being collapsed or deleted in that rectangular area. This is also useful for Dynamic Traffic Assignment applications such that certain origin-destinations can be added to the exception list so that they are not removed.

4) Application IV - Network Reduction Based on Data Driven Parameters: The fourth and the final application is data driven coarsening. The difference between this and the first application is that now the edge weights used for the coarsening are aggregates of dynamic quantities. This can be useful for real time predictions for large scale networks where the complexity increases with the size of the network, as time-dependent networks are used for this purpose [17].

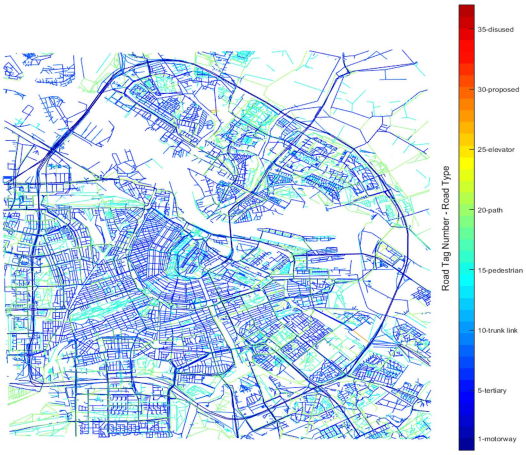


Fig. 5. Amsterdam network with 30 757 links and 34 935 nodes.

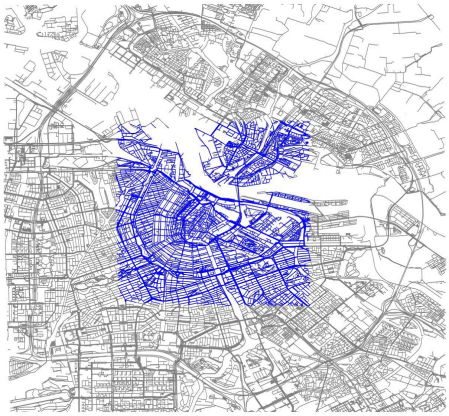


Fig. 6. Application III - Study area (in blue) in relation to the Amsterdam network.

(a) (b)

Fig. 7. Application IV - (a) Sub-network of Amsterdam (in blue) (b) Speed per link at time 16:00 for a particular day for the sub-network.

We used speed per link as the weights for coarsening the network for this application. The new weights of the links after node collapse are found using a mean function instead of a minimum function used as is done for the other applications.

B. Data

The real-world large-scale network of Amsterdam is used in the experiments (Figure 5). The Amsterdam network was extracted from the open-source open-street map(OSM) and contains 30 757 links and 34 935 nodes. Assigning weights to the links of the directed graph of Amsterdam is the first step of the heuristic graph coarsening. Given the limited availability of (open-source) data for *all* these links, we define the weight of an edge *(u,v)* with nodes u and v simply as:

$$w_{uv} = 1/t_{uv}$$

where t_{uv} corresponds to a value that depicts the type of the road network, which is readily available. Here, we use $\beta = 1$ in (1) because of the lack of additional meta information about the relative importance of these road types. In OSM, the type of the link refers to the standardized classification of the roads defined in OSM data source such as primary-link, secondary-link, access-ramp, etc, which is often used as a proxy for free-flow speed. There are 36 tags in OSM to define the type of the road segment. Each of the ordinal road classification tags is transformed into a numerical scale ranging from 1 to 36 based on the link importance of each tag described in [37] and this is assigned to t_{uv}.

For the Amsterdam network, there are 22 links which are tagged as 'road'. Since only 22 links are not properly tagged, the performance of the method is not significantly hampered. Another drawback of OSM network is that not all the nodes in the graph representation are correctly-noded [38]. This might lead to the graph being weakly connected with multiple connected components. For the Amsterdam network, there are 6759 such components with 90% of them having no more than four links. For most of the applications, these small "islands" are not that important for the study and pruning can be enabled to remove them.

For the first application, we use the Amsterdam network given in Figure 5. The edge weight is the numerically mapped road type, thus the minimum ρ is 0 and maximum is $\sigma^2(w_{min}, w_{max})$, where w_{min} is 1 and w_{max} is $\frac{1}{36}$ edge weights respectively. The same network is used for the second application case related to preserving the intersections. The rectangular study area (center of Amsterdam) boundary in relation to the larger Amsterdam network is shown in Figure 6 which is used for the third application.

In the fourth application case, we use travel time data from a license plate recognition system in Amsterdam to derive the speed per link. There were 314 pairs of start and end camera observations for the whole of Amsterdam network so the whole network is not completely utilized. The sub-network within the recognition system coverage is shown in Figure 7(a). The sub-network has 7512 links and 6528 nodes and it is a single connected component. The data preparation and conversion of travel time to speed per link is described in detail in [17]. The traffic state of Amsterdam at time 16:00 for one particular day is shown in Figure 7(b). The speed per link is used as the link weights of the sub-network.

All the applications have four scenarios as explained before with pruning $[0, 1]$ and ρ $[minimum, maximum]$. The ρ for applications I to III corresponds to the change in functional road class of links and the change in speeds for application IV. To preserve the change in speed or road class, the minimum

ρ is set to 0. The maximum ρ was set at 10000 (an arbitrarily high value) for applications I, III and IV whereas for application II the maximum ρ is also set at 0. This is because of our definition of intersection which prohibits coarsening nodes that have discontinuities in edge difference. The maximum number of iterations for all the applications was set at 10000, although the process stopped in all cases well before that.

C. Evaluation Metrics

The multiscale graphs generated using our node collapse and pruning for the four applications are evaluated based on the five criteria proposed in Section I, we use several graph measures and the computational performance for creating these graphs is also considered. All runs were done on a 64-bit computer with Intel Xeon CPU E5-1620 v3 processor of 3.5GHz, 16.0GB RAM and no GPU. In this section, we revisit the criteria for multiscale graphs and detail the four graph aspects used to quantify these.

1) Network Reduction: The reduction in the network complexity is computed based on the reduction in the number of links. The network reduction, $r^{G_{i+1}}$, for a coarsened graph G_{i+1} in relation to the original graph G_i is defined as:

$$r^{G_{i+1}} = \frac{|E| - |E'|}{|E'|} * 100\ \% \tag{7}$$

where $|E|$ and $|E'|$ are the number of links in the original graph and coarsened graph, respectively.

2) Global Topological Characteristics: The desirable property of collapsing nodes and edges is that the topological characteristics of the graph are preserved, except when pruning is enabled which alters the topology of the network. For evaluating the global topological characteristics of the coarsened graph, we use five metrics: connectivity, trip length distribution, diameter, total network length and centrality.

Graph connectivity is measured using the number of connected components in a network. The connected components are found using depth first search based algorithm [39].

Trip length distribution is equal to the shortest path distribution (in distance) given that only the shortest path is considered between origins and destinations. The influence of coarsening on the shortest path between two arbitrary nodes is important since large shifts in shortest paths between origins and destinations imply fundamental changes in network topology relevant for many applications in transportation, ranging from simulation, to planning and to applied ITS tailored at providing information of the network state to travelers, operators and service providers. Checking this requirement is particularly interesting as this check is not built in the method itself. Assuming that there are N number of nodes in the coarse network, there are $N \times N$ OD (origin-destination) pairs in that graph and hence, $N \times N$ shortest paths. By comparing the distribution of these $N \times N$ shortest path cost of these ODs in the coarse and fine scale, we can observe the effect of coarsening on these shortest paths. The shortest path is found using Dijkstra's algorithm for weighted directed graphs [40] and the weights can be distance or travel time based on the application.

Diameter is the shortest path between the two most distant nodes in the network and is measured for a given network as maximum of all calculated trip length between all the origin-destination pairs in that network.

Total network length is the total length of the transport network (in km) and measured by summation of the length of all links in the network.

Centrality characterizes the (global) importance of a node in a network. In this work, we use betweenness centrality $g(v)$ as defined in [41] and [42]:

$$g(v) = \sum_{s \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{8}$$

where σ_{st} is the number of shortest paths going from s to t and $\sigma_{st}(v)$ is the number of shortest paths from s to t through node v. Betweenness centrality can be considered as a proxy for flow throughput as a node is said to have high centrality if a large number of the shortest paths in the network go through that node. Thus, a similar betweenness centrality distribution needs to be maintained for the different scales as this implies similar hierarchy among nodes at different scales. This distribution is known to follow a power law for most transport networks, defined as:

$$g(v) \ \sim \ v^{-\eta} \tag{9}$$

where η is the power law exponent [41]. The value of power law exponent is typically in the range $2 < \eta < 3$ for scale-free networks, although not in all cases [43]. Our assumption is that the exponent value should not degrade significantly between different scales, at least for coarsening without pruning. So, if a network is originally scale-free, it is desired to maintain this property also for the coarsened network. Other than the power law exponent, it is also important to maintain the quality of the power law fit. In this work, we use the R^2 value as the goodness-of-fit measure for the regression model which varies between 0 and 1. A value of 1 implies a perfect fit to the data.

3) Domain Specific Characteristics: Depending on the application, the domain specific attributes of the network are either static link attributes such as functional road class, speed limits, capacities, geo-information or accurate aggregates/averages of dynamic quantities such as average flows, average speeds, travel times at a particular time for the network. These attributes define the link weights of the network which is then used for coarsening. In the algorithm, we have a hard constraint $c_2(v)$ (see Table II), in which the ρ determines to what degree the attributes are preserved while coarsening. If ρ is 0 in $c_2(v)$,

$$|w_{iv} - \mu|^2 + |w_{vj} - \mu|^2 = 0 \implies w_{iv} = w_{vj} \tag{10}$$

Thus, for links (i, v) and (v, j), the links are coarsened only if the link weights are the same and hence the link weights are fully preserved for $\rho = 0$ and by mathematical induction, it holds for all links in the network and for different ρ values.

4) Local Topological Characteristics: While the trip length distribution shows the global trend of shortest paths in that network, it does not show if the same shortest path is maintained in the coarsened graph as the original graph. To observe

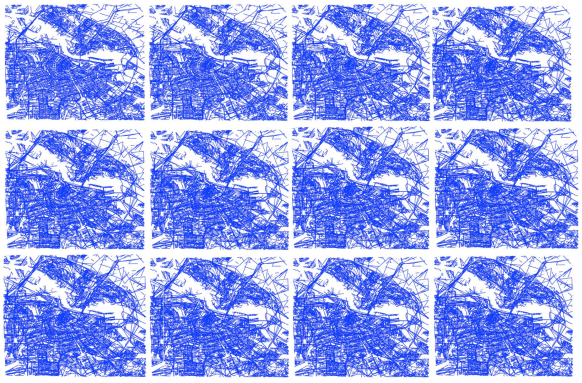


Fig. 8. Evolution of the Amsterdam network during 12 iterations for scenario 2 of application I (without pruning), with network reduction ranging from 21% to 47%.

this, we use the shortest path deterioration of each OD pair st of $N \times N$ OD pairs for a given coarse scale graph G_{i+1} defined as:

$$D_{st}^{G_{i+1}} = \frac{c_{st}^{G_{i+1}} - c_{st}^{G_i}}{c_{st}^{G_i}} * 100\ \% \tag{11}$$

where $c_{st}^{G_{i+1}}$ and $c_{st}^{G_i}$ is the shortest path cost between nodes s and t in the coarsened graph G_{i+1} and the original graph G_i respectively.

V. Results and Discussion

In this section, we present the results of the four applications and their corresponding scenarios. We evaluate application I in depth in relation to the evaluation metrics since this application aims at the maximum possible reduction of network size without any restrictions using our coarsening framework. Hence, it is expected to manifest the maximum deterioration of the topological properties and thus offers the most conservative performance assessment. We discuss the other applications more briefly. Table III shows the coarsening results for the four applications and their scenarios.

For application I, the node coarsening leads to a network edge reduction of $21-47\%$ without pruning giving the same number of connected components as in the original network (6759 components). Thus, the connectivity is preserved without pruning. Since the approach is iterative, the 47% reduction is achieved after 12 iterations. This implies there are 12 multiscale graphs available with a reduction within the range of $21-47\%$ respectively compared to the complete Amsterdam network as shown in Figure 8. *With* pruning, the edge reduction ranges between $59-96\%$, resulting in a more compact network of 1103 components for the minimum ρ and a single component for the maximum ρ after 15 iterations as shown in Figure 9. (Note that only pruning has an effect on the number of components, node collapsing has no influence on the network connectivity.) The computation time for coarsening is 10 to 15 minutes with and without pruning respectively. We return to the results of the other applications listed in Table III further below.

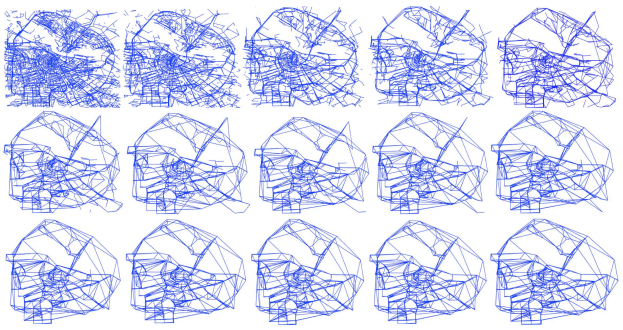


Fig. 9. Evolution of the Amsterdam network during 15 iterations for scenario 4 of application I (with pruning), with network reduction ranging from 59% to 96%.

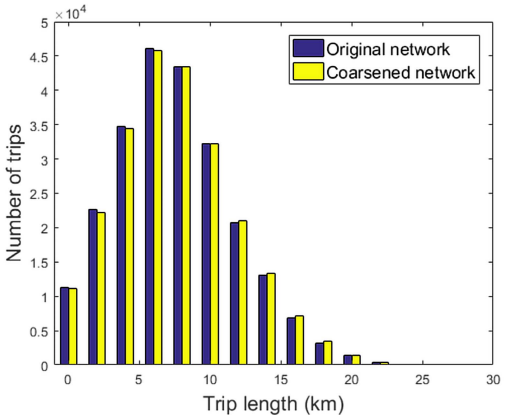
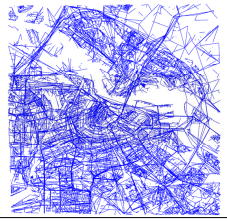
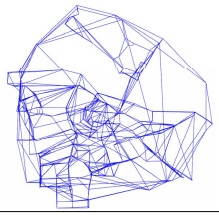
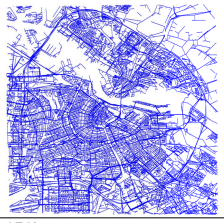
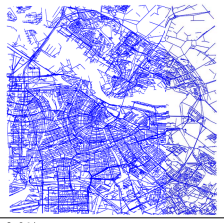
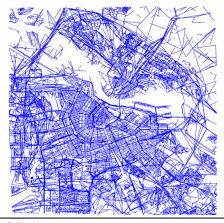
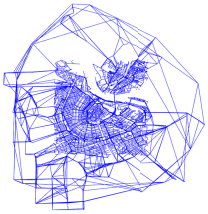
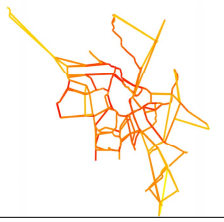
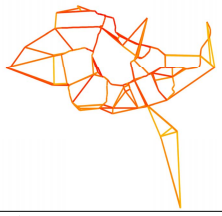


Fig. 10. Trip length distribution of the OD pairs in the Amsterdam network for application I.

We first examine some global topological characteristics of the multiscale graphs resulting from application I. First, the trip length distribution of the network. There are 427 nodes remaining in the final coarsened graph (application I, scenario 4). Therefore, there are 427×427 origin-destination (OD) pairs leading to 182 329 shortest paths. The link length is used to estimate the shortest path cost for application I. The trip length distribution of these shortest paths in the original and coarsened network are shown in Figure 10. As can be seen, the trip length distribution of the original and coarsened graph is quite similar, even for the maximum network reduction. We have also investigated the complete trip length distribution of the original and coarsened graph and found that while the shape of the distribution remains similar, some of the ODs have deflated shortest path lengths, especially due to node removal. This is also evident from the network diameter reported in Table III for all applications.

The diameter of the original network is 38.92km. From Table III, it can be seen that the diameter of the network decreases for all scenarios. The slight decrease in diameter when pruning is disabled is unexpected as we expect it to be unchanged. But, this is because of shortcuts which are created due to node coarsening that are duplicates of links already existing in the network. Consequently, there are two separate links connecting the same nodes with different link length. Since the minimum of these length will be used for computing

TABLE III
COARSENING RESULTS OF THE FOUR APPLICATIONS

		Scenarios			
		pruning = 0 ρ = minimum iterations = 1	pruning = 0 ρ = maximum iterations = maximum	pruning = 1 ρ = minimum iterations = 1	pruning = 1 ρ = maximum iterations = maximum
Application I	Results				
	Network reduction	21%	47%	59%	96%
	Connectivity	6759	6759	1103	1
	Network length	3746 km	4945 km	2268 km	1213 km
	Diameter	38.80 km	38.69 km	38.80 km	29.52 km
	η, R^2	3.29, 1.00	3.08, 0.99	2.98, 0.99	2.13, 0.90
	Computation time	10 - 15 minutes			
Application II	Results				
	Network reduction	17%	26%	57%	88%
	Connectivity	6759	6759	1139	7
	Network length	3617 km	3617 km	2201 km	1036 km
	Diameter	38.80 km	38.80 km	38.80 km	38.80 km
	η, R^2	3.25, 1.00	3.09, 0.99	2.99, 0.99	2.26, 0.96
	Computation time	10 - 15 minutes			
Application III	Results				
	Network reduction	14%	32%	43%	68%
	Connectivity	6759	6759	2379	1588
	Network length	3716 km	4580 km	2577 km	1782 km
	Diameter	38.80 km	38.69 km	38.80 km	32.13 km
	η, R^2	3.10, 0.99	3.08, 0.99	2.94, 0.99	3.19, 0.99
	Computation Time	6 - 10 minutes			
Application IV	Results				
	Network reduction	38%	74%	40%	85%
	Connectivity	1	1	1	1
	Network length	311 km	366 km	309 km	249 km
	Diameter	51.27 km	48.39 km	51.29 km	19.03 km
	η, R^2	1.95, 0.81	1.66, 0.89	1.94, 0.81	1.49, 0.75
	Computation time	30 - 78 seconds			

the shortest paths that traverse this link, there is a decrease in the diameter.

The other global characteristics considered is the total network length. The total network length of the original network is 3622km. From Table III, it can be seen that the network length increases significantly for coarsening without pruning while decreases for coarsening with pruning. The decrease in network length is expected as links are removed due to

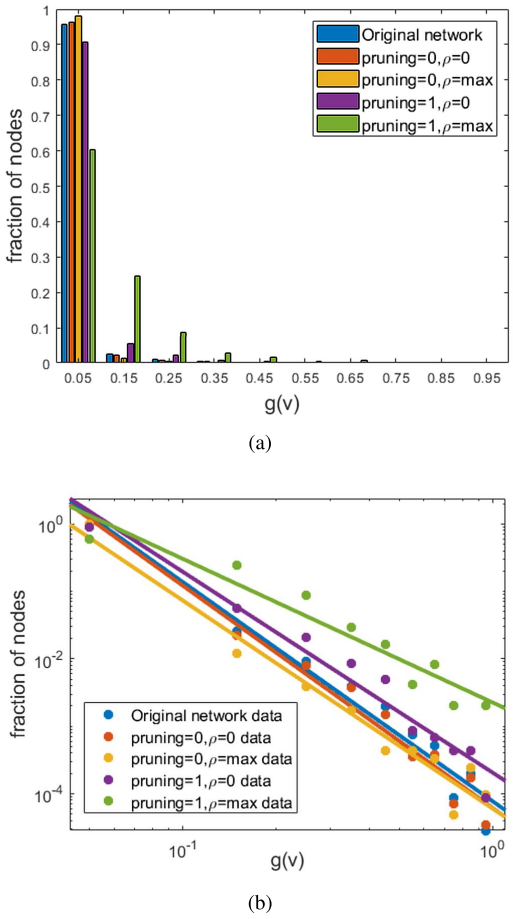


Fig. 11. (a) Influence of coarsening on node centrality distribution; (b) Log-log plot of node centrality with the data as dots and the corresponding line showing the power law fit.

pruning. However, the increase in network length is due to the shortcuts created during coarsening leading to direct connections between nodes as shown in example cases (2) and (6) in Figure 2.

We also consider node centrality characteristics of the multiscale graphs, which exhibit similar distribution as compared to the original graph as illustrated in Figure 11(a). This distribution of the multiscale graphs follows a power law and the corresponding power law parameters that define the relationship between centrality and the number of nodes as $g(v) \sim v^{-\eta}$. Estimated power law parameters for the relation of the betweenness centrality versus the number of nodes are shown in Figure 11(b). The original Amsterdam network has a power-law with exponent $\eta \approx 3.25$ with $R^2 = 1.00$ whereas for coarsening without pruning has $\eta \approx 3.29 - 3.08$ with approximately the same R^2 as shown in Table III. The power-law exponents for coarsening with pruning is $\eta \approx 2.98 - 2.13$.

Thus, these multiscale graphs have similar exponent as that of a scale-free network by employing pruning which is significantly different than that of the original network. However, the exponents without pruning are similar to those of the original network and this is also evident from Figure 11(b).

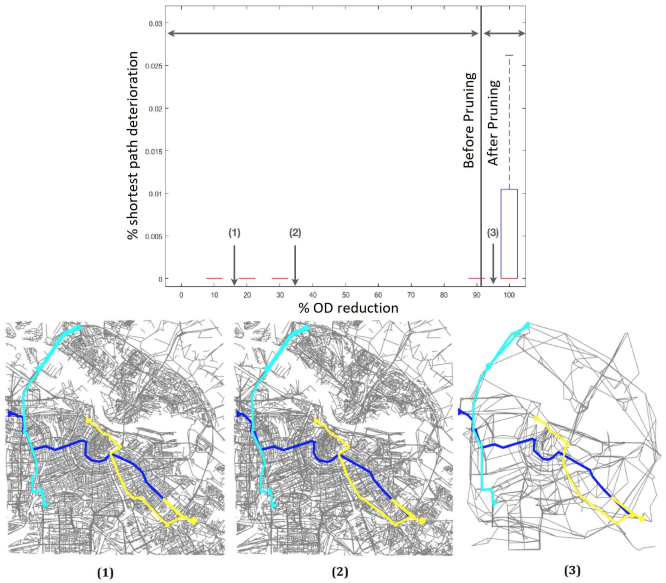


Fig. 12. Boxplot of shortest path deterioration versus the share of nodes removed (OD reduction) for all iterations in all scenarios for application I. (1), (2) and (3) represent the multiscale graphs at 16%, 34% and 93% OD reduction respectively. Same three shortest paths with the largest deterioration are shown in (1), (2) and (3) for illustration.

The degradation of the exponent in case of pruning is probably because of the high number of disconnected components in the network. This assumption is confirmed in application IV which have a single component. For application IV, the original network has a power-law exponent of 2.43 with $R^2 = 0.80$ whereas the coarsened networks have exponents less than 2.00 while the R^2 remains similar. Thus, all the multiscale networks seem to follow a power law with similar goodness-of-fit as the original network, however the scale-free property of these networks are inconclusive using just the power law exponent. The exponent degradation between all scenarios in all applications is approximately 1, which seems marginal.

Figure 12 shows the influence of coarsening on the length of the shortest paths in the graphs for application I. The trip length distribution of the multiscale graphs were shown to be similar; however the actual shortest path has to be studied in detail to check if it is maintained within the different multiscale graphs. We used relative shortest path deterioration to check this local characteristics for application I. As can be seen from the figure, the effect of coarsening on the shortest path is minimal. With more than 95% of the nodes removed, there is only a maximum deterioration of 0.025% in the shortest path cost. Thus, even extreme pruning only has a marginal influence on the shortest path, which is an important finding for the application scope. For most of the traffic applications, a 0.025% deterioration in distance is negligible.

Table III summarizes the results for all the applications. The node coarsening for the preservation of intersection application provides a network reduction of $17 - 26\%$ without pruning after 7 iterations and $57 - 88\%$ with pruning after 17 iterations. The node coarsening for the study area application provides the least network reduction of $14 - 32\%$ without pruning after 8 iterations and $43 - 68\%$ with pruning after 13 iterations.

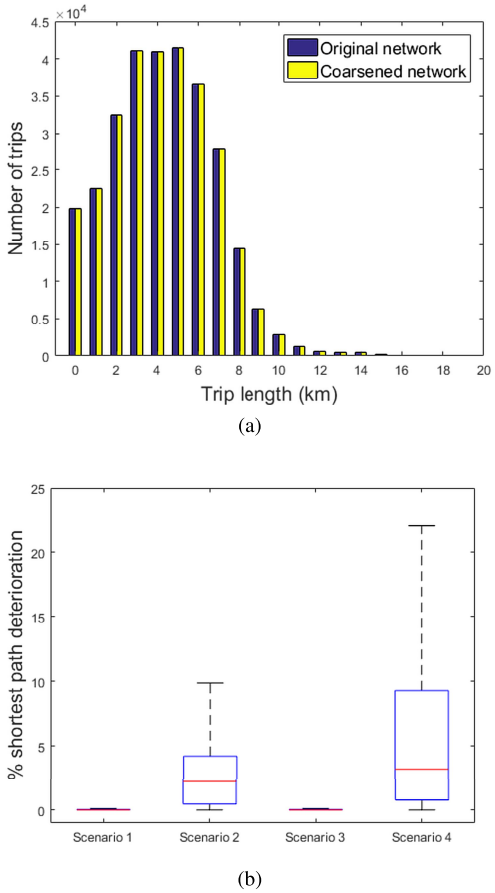


Fig. 13. Shortest path deterioration based on distance and travel time for application IV.

The higher number of components (see Table III) and lower network reduction compared to the previous applications are because of the unaltered study area which corresponds to 30% of the network size and comprises of 1500 components. This also corresponds to the low degradation of the values for the characteristics such as the diameter, network length, power law exponent and fit compared to application I as reported in Table III. All the centrality distribution of multiscale graphs follows the power law with similar R^2 value as those of the original network. The node coarsening results for the data driven application case provides a network reduction of $38-74\%$ without pruning after 10 iterations. With pruning, it provides a reduction of $40-85\%$ after 86 iterations.

For application IV, travel time is used to compute the shortest path. The travel time deterioration for the coarsened graph and the original graph for application IV for different scenarios are shown in Figure 13(b). There are 538×538 OD pairs for this application resulting in 289 444 shortest paths. There is a maximum deterioration of 22% in travel time. However, the shortest path distribution based on distance as the cost, shown in Figure 13(a), has an approximately zero deterioration, i.e. very minimal undesired alterations. This is because the distance of the link is preserved when coarsening whereas the link weights (average speed) are skewed because of averaging when $\rho > 0$. The travel time is preserved better for smaller ρ (scenarios 1 and 3) with a maximum deterioration of 0.1%.

VI. Conclusion

In this paper, we propose a generic heuristic coarsening technique for generating multiscale graphs. Compared to existing graph based methods popularized in Experimental Algorithms, we presented a heuristic method that is directly applicable and versatile for many transport applications. The method comprises of two main building blocks: node ordering and node contraction. Importantly, the explosion of the average node degree in the coarse network is eliminated by a simple decision rule, which can also be easily relaxed. By setting an edge difference variance threshold ρ, the graph can be generated at the required coarsened level. The method is demonstrated on the Amsterdam city network for four applications. The method was able to successfully reduce the Amsterdam network by up to 96% of its original size at a computation time of no more than 15 minutes with a limited loss of information as indicated by the preservation of key network characteristics. For offline use (e.g. transportation planning, simulation model generation), this performance is considered satisfactory.

An initial verification study analysing some of the topological measures provide some important observations. For global and local topology preservation applications, pruning must be disabled as it might remove a node or link that is necessary for the shortest path and connectivity. However, for applications that require a more compact network, pruning is useful. With more than 95% of the nodes removed, we found a maximum deterioration of just 0.025% in the shortest path of the ODs. This confirms the assumption that it is not necessary to check if the shortest path is preserved after each node collapse. We also showed that all the multiscale Amsterdam networks fit the power law with similar goodness-of-fit as the original network with a maximum deterioration of approximately 1 for the power law exponent. One of the key finding from the four application cases was that there is at least a minimum reduction of $14-38\%$ in the network complexity with the strictest constraints. This decrease can have significant impact on computational efficiency especially when dealing with time-dependent networks.

From the results we conclude that the algorithm offers an effective coarsening solution for many transport applications. The method can be used as a preprocessing step for many network-wide applications - either to reduce the network complexity or to generate multiscale representations of these networks. Application include a range of ITS applications from design to control such as hybrid modeling, traffic assignment, real time predictions, visualisations, etc. Most of the operations on large-scale networks require hardware capabilities or high-level optimisations to be viable for real-time or even exploratory studies. This simple open-source algorithm is an initial step to remove some of these complexities at network-level before adding high dimensional information on the graph. This is especially important in the age of big data, where data availability may no longer be the problem, but

rather the efficient capabilities to use the data may prove a new bottleneck for modelers and analysts.

ACKNOWLEDGMENT

Map data copyrighted OpenStreetMap contributors and available from https://www.openstreetmap.org.

REFERENCES

[1] G. Karypis and V. Kumar, "Multilevel graph partitioning schemes," in *Proc. ICPP*, vol. 3, Aug. 1995, pp. 113–122.
[2] C. Chevalier and I. Safro, "Comparison of coarsening schemes for multilevel graph partitioning," in *Proc. Int. Conf. Learn. Intell. Optim.*, Jan. 2009, pp. 191–205.
[3] R. Geisberger, P. Sanders, D. Schultes, and D. Delling, "Contraction hierarchies: Faster and simpler hierarchical routing in road networks," in *Proc. Int. Workshop Exp. Efficient Algorithms*. Berlin, Germany: Springer, May 2008, pp. 319–333.
[4] P. Sanders and D. Schultes, "Engineering highway hierarchies," in *Proc. ESA*. Berlin, Germany: Springer, Sep. 2006, pp. 804–816.
[5] D. Delling, P. Sanders, D. Schultes, and D. Wagner, "Engineering route planning algorithms," in *Algorithmics of Large and Complex Networks*. Berlin, Germany: Springer, 2009, pp. 117–139.
[6] Z. Cao, H. Guo, J. Zhang, D. Niyato, and U. Fastenrath, "Finding the shortest path in stochastic vehicle routing: A cardinality minimization approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 6, pp. 1688–1702, Jun. 2016.
[7] B. N. Janson, "Dynamic traffic assignment for urban road networks," *Transp. Res. B, Methodol.*, vol. 25, nos. 2–3, pp. 143–161, Apr./Jun. 1991.
[8] E. Jafari and S. D. Boyles, "Improved bush-based methods for network contraction," *Transp. Res. B, Methodol.*, vol. 83, pp. 298–313, Jan. 2016.
[9] R. D. Connors and D. P. Watling, "Assessing the demand vulnerability of equilibrium traffic networks via network aggregation," *Netw. Spatial Econ.*, vol. 15, no. 2, pp. 367–395, Jun. 2015.
[10] S. D. Boyles, "Bush-based sensitivity analysis for approximating subnetwork diversion," *Transp. Res. B, Methodol.*, vol. 46, no. 1, pp. 139–155, Jan. 2012.
[11] R. D. Connors and D. P. Watling, "Aggregation of transport networks using sensitivity analysis," in *Proc. Eur. Transp. Conf.*, 2008, p. 13.
[12] M. Joueiai, L. Leclercq, H. van Lint, and S. P. Hoogendoorn, "Multiscale traffic flow model based on the mesoscopic lighthill–whitham and richards models," *Transp. Res. Record*, vol. 2491, no. 1, pp. 98–106, Jan. 2015.
[13] Y. Xu, W. Cai, H. Aydt, and M. Lees, "Efficient graph-based dynamic load-balancing for parallel large-scale agent-based traffic simulation," in *Proc. Winter Simulation Conf.*, Dec. 2014, pp. 3483–3494.
[14] M. Ben-Akiva, M. Bierlaire, D. Burton, H. N. Koutsopoulos, and R. Mishalani, "Network state estimation and prediction for real-time traffic management," *Netw. Spatial Econ.*, vol. 1, nos. 3–4, pp. 293–318, Sep. 2001.
[15] Y. Yuan, J. W. C. Van Lint, R. E. Wilson, F. van Wageningen-Kessels, and S. P. Hoogendoorn, "Real-time lagrangian traffic state estimator for freeways," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 1, pp. 59–70, Mar. 2012.
[16] C. Lopez, L. Leclercq, P. Krishnakumari, N. Chiabaut, and H. van Lint, "Revealing the day-to-day regularity of urban congestion patterns with 3D speed maps," *Sci. Rep.*, vol. 7, no. 1, Oct. 2017, Art. no. 14029.
[17] C. Lopez, P. Krishnakumari, L. Leclercq, N. Chiabaut, and H. Van Lint, "Spatiotemporal partitioning of transportation network using travel time data," *Transp. Res. Rec., J. Transp. Res. Board*, vol. 2623, no. 1, pp. 98–107, 2017.
[18] X. Chen, S. Zhang, L. Li, and L. Li, "Adaptive rolling smoothing with heterogeneous data for traffic state estimation and prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 4, pp. 1247–1258, Apr. 2019.
[19] S. D. Dimitrov and A. Ceder, "A method of examining the structure and topological properties of public-transport networks," *Phys. A, Stat. Mech. Appl.*, vol. 451, pp. 373–387, Jun. 2016.
[20] D. Schultes, "Route planning in road networks," in *Proc. Ausgezeichnete Informatikdissertationen*, Feb. 2008, pp. 271–280.
[21] H. Bast *et al.*, "Route planning in transportation networks," in *Algorithm Engineering*. Cham, Switzerland: Springer, 2016, pp. 19–80.
[22] A. Brandt and D. Ron, "Multigrid solvers and multilevel optimization strategies," in *Multilevel Optimization in VLSICAD*, vol. 14. Boston, MA, USA: Springer, 2013, p. 1.
[23] I. Safro, D. Ron, and A. Brandt, "Multilevel algorithms for linear ordering problems," *J. Experim. Algorithmics*, vol. 13, p. 4, Feb. 2009.
[24] A. Brandt, "General highly accurate algebraic coarsening," *Electron. Trans. Numer. Anal.*, vol. 10, no. 1, p. 21, Feb. 2000.
[25] S. Storandt, "Contraction hierarchies on grid graphs," in *Proc. Annu. Conf. Artif. Intell.* Berlin, Germany: Springer, Sep. 2013, pp. 236–247.
[26] J. Dibbelt, B. Strasser, and D. Wagner, "Customizable contraction hierarchies," *J. Exp. Algorithmics*, vol. 21, pp. 1–5, 2016.
[27] C. Ollivier-Gooch, "Coarsening unstructured meshes by edge contraction," *Int. J. Numer. Methods Eng.*, vol. 57, no. 3, pp. 391–414, May 2003.
[28] A. Gutfraind, I. Safro, and L. A. Meyers, "Multiscale network generation," in *Proc. IEEE 18th Int. Conf. Inf. Fusion*, Jul. 2015, pp. 158–165.
[29] J. Dibbelt, B. Strasser, and D. Wagner. (2015). "Fast exact shortest path and distance queries on road networks with parametrized costs." [Online]. Available: https://arxiv.org/abs/1509.03165
[30] D. Delling, A. V. Goldberg, T. Pajor, and R. F. Werneck, "Customizable route planning in road networks," *Transp. Sci.*, vol. 51, no. 2, pp. 566–591, May 2017.
[31] S. Funke, A. Nusser, and S. Storandt, "On k-path covers and their applications," *Int. J. Very Large Data Bases,*, vol. 25, no. 1, pp. 103–123, Feb. 2016.
[32] R. Geisberger, P. Sanders, D. Schultes, and C. Vetter, "Exact routing in large road networks using contraction hierarchies," *Transp. Sci.*, vol. 46, no. 3, pp. 388–404, Aug. 2012.
[33] B. F. Auer and R. H. Bisseling, "Graph coarsening and clustering on the gpu," *Graph Partitioning Graph Clustering*, vol. 588, p. 223, Feb. 2012.
[34] I. Safro, P. Sanders, and C. Schulz, "Advanced coarsening schemes for graph partitioning," *J. Exp. Algorithmics*, vol. 19, p. 2, Feb. 2015.
[35] N. Tatti and A. Gionis, "Density-friendly graph decomposition," in *Proc. 24th Int. Conf. World Wide Web*, ACM, May 2015, pp. 1089–1099.
[36] L. Moreira-Matias, J. Mendes-Moreira, J. Gama, and M. Ferreira, "On improving operational planning and control in public transportation networks using streaming data: A machine learning approach," Ph.D. dissertation, Dept. Inform. Eng., Univ. Porto, Porto, Portugal, 2014.
[37] *Openstreet Maps Highway Tag Wiki*. Accessed: Aug. 15, 2016. [Online]. Available: https://wiki.openstreetmap.org/wiki/Key:highway
[38] *Wrongly-Noded Graph in Openstreet Maps*. Accessed: Aug. 15, 2016. [Online] Available: https://www.gaia-gis.it/fossil/spatialite-tools/wiki?name=graphs-intro
[39] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM J. Comput.*, vol. 1, no. 2, pp. 146–160, Jun. 1972.
[40] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Math.*, vol. 1, no. 1, pp. 269–271, Dec. 1959.
[41] M. Barthélemy, "Betweenness centrality in large complex networks," *Eur. Phys. J. B*, vol. 38, no. 2, pp. 163–168, Mar. 2004.
[42] M. Barthélemy, "Spatial networks," *Phys. Rep.*, vol. 499, nos. 1–3, pp. 1–101, Feb. 2011.
[43] K. Choromański, M. Matuszak, and J. Miękisz, "Scale-free graph with preferential attachment and evolving internal vertex structure," *J. Statist. Phys.*, vol. 151, no. 6, pp. 1175–1183, Jun. 2013.

Panchamy Krishnakumari received the M.Sc. degrees in computer science from the KTH Royal Institute of Technology, Sweden, and the Delft University of Technology, The Netherlands, as part of an European Master's program, EIT Digital, in 2015, where she is currently pursuing the Ph.D. degree with the Department of Transport and Planning. Her main research focuses on applying pattern recognition techniques on large scale transportation networks.

Oded Cats received the dual Ph.D. degrees from the KTH Royal Institute of Technology, Stockholm, and the Technion—Israel Institute of Technology. He co-directs the Smart Public Transport Lab, TU Delft, leading a research group that works closely with public transport authorities and operators. He is currently an Assistant Professor of public transport with the Delft University of Technology (TU Delft). He has coauthored more than 55 peer-reviewed journal publications. His research develops methods and models of multi-modal metropolitan passenger transport systems by combining advancements from behavioral sciences, operations research, and complex network theory. His research contributions focus on the development of dynamic transit assignment models, the optimization of passenger service operations, network robustness analysis, and real-time control methods. He is a member of several prestigious international committees including three Transportation Research Board committees and PTV Scientific Advisory Board. He is an Associate Editor of the *European Journal of Transport and Infrastructure Research*.

Hans van Lint received the M.Sc. degree in civil engineering informatics and the Ph.D. degree in transportation from the Delft University of Technology in 1997 and 2004, respectively. He was an Information Analyst and a Transport Engineer at various organizations. Nine years after receiving the Ph.D. degree, he was appointed as an Anthonie van Leeuwenhoek (AvL) Full Professor by the Executive Board of TU Delft in 2013 (an honor reserved for only a few young scientists and educators). His expertise lies on the interface between traffic flow theory and simulation, data analytics, and machine learning techniques. He has co-promoted eight Ph.D. students and has currently ten Ph.D.s, five postdocs, and two programmers under supervision in his lab. He has coauthored more than 55 peer reviewed journal articles. He is active in many international projects and collaborations. He serves as an Associate Editor for the IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS.