Pre-processing Road Networks for Graph Partitioning Using Edge-Betweenness Centrality

Bachelor's Thesis

Daan Reijnders (4302001)

Supervisor:
Rob Bisseling
Mathematical Institute
Utrecht University

Utrecht, June 2017

Acknowledgements

I would like to express my gratitude to Prof. Dr. Bisseling of the Mathematical Institute at Utrecht University for guiding me through the process of writing this thesis. Through the fruitful discussions during our meetings he has been able to get me moving again multiple times after I had been stuck in gear while overlooking intuitive solutions.

I would also like to thank Prof. Dr. Bisseling for providing me access to Cartesius, the national supercomputer. Although I did not have any previous experience with parallel computing, learning how to effectively use parallel techniques significantly widened my computational capabilities.

On this note, I would like to express my gratitude to the SURFsara help desk, for helping me out with how to use the most powerful computer I have ever worked with thus far.

I would also like to thank Christian Schulz for providing me with the source code of Buffoon, which allowed me to put the performance of the preprocessing heuristic proposed in this thesis into perspective.

Lastly, I wish to thank Chris Walshaw for providing me with his set of benchmark graphs, which enabled me to test the new preprocessing heuristic proposed in this thesis on other types of graphs, allowing me to investigate whether this heuristic could prove to be useful outside of road networks.

I hope the preprocessing heuristic proposed in this thesis can provide useful new insights for the rest of the graph partitioning community to use.

Contents

1 Introduction

Road networks are an everyday and intuitive, real-world example of (approximately) planar graphs. As is the case for many other types of graphs, graph partitioning algorithms are used for partitioning road networks to solve a variety of problems, for instance in route-planning [8] or for simplifying traffic analysis [19].

Graph partitioning is concerned with dividing a graph $G = (V, E)$ into smaller parts of approximately equal size. One objective can be to find a partitioning with the smallest possible cut-size, meaning that the number of edges spanning between vertices in different partitions is kept minimal. As the difficulty of finding the optimal partitioning increases exponentially, researchers are trying to find efficient heuristic methods that provide sufficiently small cut-sizes whilst keeping computation times short.

Heuristic algorithms for finding partitionings with small cut-sizes have been proposed since the 1970s [10], when Brian Kernighan and Shen Lin developed one of the earliest and simplest graph partitioning algorithms. The Kernighan–Lin (KL) algorithm follows a straightforward procedure: it takes an existing partitioning, which could initially be obtained at random, and in each pass identifies and swaps the pair of vertices in different partitions such that the cut-size is reduced the most [17]. Continuously choosing the best improvements ensures that the KL algorithm identifies an optimal solution, although this solution can in theory be a *locally* optimal solution, which may be of a significantly lesser quality than the *globally* optimal solution.

After the Kernighan-Lin algorithm was proposed, many other partitioning techniques have been developed. The Fiduccia-Mattheyses heuristic, for example, adapts elements from the KL algorithm, but runs a lot faster since it is able to run in linear time [9]. Other algorithms take a completely different approach to graph partition, examples of which include probabilistic implementations such as simulated annealing [14] or genetic algorithms [22].

Another class of graph partitioning algorithms are the *multilevel graph partitioning* (MGP) schemes. MGP algorithms employ methods that are oriented towards finding globally optimal solutions and are able to provide partitionings of a reasonably good quality with a small cost compared to other algorithms [16]. MGP methods take a graph G_0 as input and work by contracting vertices to obtain a much coarser version of the graph. When the coarse graph G_i is sufficiently small, a high-quality algorithm is run to partition the graph, after which the graph can be uncoarsened. The partitionings are refined during each step in the uncoarsening phase, for example by using the Kernighan-Lin heuristic or variations thereof. After uncoarsening all contracted vertices, a partitioning of the original graph G_0 is obtained.

The first two multilevel graph partitioning heuristics were independently proposed in 1993 [13, 6], after which multiple other MGP algorithms have been developed. These implementations all have a similar structure, but vary in methods for pre-processing graphs, partitioning the coarsened graph, and refining the partitionings during the uncoarsening phase. Some MPG heuristics even employ strategies found in other graph partitioning algorithms, for example by implementing simulated annealing routines [16] or methods inspired by genetic algorithms [7].

As networks can take on many different structures, the performance of different heuristics for the graph partitioning problem varies significantly. The performance of an algorithm can depend on many factors, such as local network structures, planarity, centrality, existence of power-laws, or sparsity.

Obvious properties of road networks are their near-planar structure and their low vertex degrees. These vertex degrees are distributed evenly and are seldom higher than five, making road networks "random networks", which can never be scale-free [25]. However, power laws exist within the betweenness centrality of the edges of a road network. These and other properties of road networks can influence the performance of graph partitioning algorithms when applied to these networks.

It has been shown that for road network partitioning, simulated annealing and the classical KL algorithm perform poorly [10]. MGP algorithms perform noticeably better on road networks than many other types of algorithms do. As graph partitioning has important applications for road networks, dedicated MGP heuristics have been developed for partitioning road networks. These algorithms make sophisticated use of the spatial and topological properties that road networks possess to identify natural cuts occurring in road networks. Using dedicated pre-processing techniques tailored to road networks can significantly improve the cut-size of the final partitioning.

In this thesis, we present a novel heuristic for partitioning road networks. This heuristic is based on the notion that in certain cases, edges with high betweenness centralities are excellent cut-edge candidates. This is called the ***H**igh **E**dge-**B**etweenness centrality **cut*** (HEB-cut) heuristic. We developed a simple implementation of this heuristic, which pre-processes road networks to obtain a weighted version of the graph that can be used as input to existing MGP algorithms, such as METIS [16]. We show that our simple implementation of the HEB-cut heuristic performs well on large road networks. Our implementation of the HEB-cut heuristic is simple and can further be refined in multiple ways. With this thesis, we hope to inspire further research into using the HEB-cut heuristic for graph partitioning, as well as for more sophisticated implementations of this heuristic to be developed.

First, we will introduce the necessary terminology and notations, after which we will describe how multilevel graph partitioning algorithms can be applied to road networks. The techniques employed by general purpose MGP algorithms as well as MGP schemes specifically developed for road network partitioning are briefly explained. We will then explain the motivations leading us to investigate the HEB-cut heuristic, after which we will describe this heuristic in more detail. We will then describe several experiments that show in which cases the HEB-cut heuristic proves to be effective. Lastly, we briefly compare the effectiveness of our implementation of the HEB-cut heuristic to that of sophisticated road network partitioning algorithms, and we briefly touch upon the usefulness of the HEB-cut heuristic for other types of networks.

2 Preliminaries

A road network can be represented as a graph $G = (V, E)$. Each point where two or more roads in the road network intersect is represented by a vertex $v \in V$, and each road segment connecting these points of intersection is represented by an edge $e = \{u, v\} \in E$. The number of vertices in a graph is denoted by n, and m denotes the number of edges. Any given road within a road network can therefore be described as a set of edges joined by vertices. For simplicity, we consider only those roads that are publicly drive-able for motor vehicles to be part of road networks. Moreover, although road networks can be represented as directed graphs by taking the driving direction of a road into account, we simplify the network by making it undirected. Two-way streets are thus represented by one edge.

For the sake of this thesis we take G to be simple and connected. Vertices and edges can have positive weights, $w_v(v)$ and $w_e(e)$ respectively, such that for a set $B \subseteq V$, $w_v(B) = \sum_{v_i \in B} w_v(v_i)$, and for $D \subseteq E$, $w_e(D) = \sum_{e_i \in D} w_e(e_i)$.

In a graph $G = (V, E)$, $\sigma(v, u)$ is the number of shortest paths between a pair of vertices v, u and $\sigma(v, u|e)$ is the number of shortest paths between v and u that pass through e. The edge-betweenness centrality of an edge e, is defined as $EBC(e) = \sum_{v,u \in V} \frac{\sigma(v,u|e)}{\sigma(v,u)}$, where $v \neq u$, and indicates the relative prevalence of an edge e in the shortest paths between all pairs of vertices. Edge-betweenness centrality, which is abbreviated as *EBC*, can be normalised by $\frac{2}{n(n-1)}$, which will be assumed to be the case whenever EBC is mentioned.

We can find a partitioning $P = \{P_1, P_2, \ldots, P_k\}$ of V in k partitions $P_i \subseteq V$, for which $P_i \cap P_j = \emptyset$ for $i \neq j$, such that $\cup_{i=1}^{k} P_i = V$. Any edge $e = \{u, v\}$ for which $u \in P_i$, but for which $v \notin P_i$, is an edge transcending two partitions. These edges are referred to as *cut-edges*. Given a set $S \subseteq V$, the cut-edges running from vertices within S to vertices not in S are denoted by $\kappa(S) = \{\{u, v\} : \{u, v\} \in E, u \in S, v \notin S\}$. The edges running between the different partitions in a partitioning P are therefore denoted by $\kappa(P)$ and we call $|\kappa(P)|$ the *edgecut* of a partitioning. Lastly, we define a cost function for a partitioning P as follows: $cost(P) = w(\kappa(P))$. Partitionings with a low cost are also referred to as high quality partitionings.

Partitionings are often required to be approximately balanced, meaning that in a partitioning P, the vertex weight of each of the k partitions is approximately equal. This is done by requiring the vertex weight of each partition in P to be at most $(1 + \epsilon)\lceil w_e(V)/k \rceil$, such that ϵ is a measure of the allowed between the different partitions *imbalance* [7]. To obtain perfectly balanced partitionings, i.e. finding partitions of equal vertex weights, the imbalance requirement would then be $\epsilon = 0$.

When we initially represent a road network by a graph, we treat all road segments equally and therefore set unitary edge and vertex weights, i.e., $w(v) = 1 \; \forall \; v \in V$, and $w(e) = 1 \; \forall \; e \in E$.

Using these conventions we can formally define the objective of graph partitioning for road networks. For a graph $G = (V, E)$ representing a road network, we wish to find a partitioning P of G for which the $cost(P) = w(\kappa(P))$ is minimal. This problem has been shown to be NP-complete [11]. Notice that for unitary edge weights, we have $cost(P) = w(\kappa(P)) = |\kappa(P)|$. Since the initial graph representation of a road network consists of edges and vertices with unitary weights, the graph partitioning problem for road network reduces to finding a partitioning with a minimal edgecut.

3 Multilevel Graph Partitioners

Although multiple multilevel graph partitioning implementations exist, they all follow a similar approach. This section first describes the general structure of MGP algorithms, after which we focus on METIS [16], the algorithm used for testing our pre-processing routine. We also briefly discuss how road networks can be partitioned using MGP algorithms and we examine the techniques used by two dedicated road network MGP algorithms, PUNCH [7] and Buffoon [21].

3.1 MGP explained

Multilevel graph partitioning algorithms generally follow a three-phase structure, first introduced by Hendrickson and Leland [13] and Bui and Moon [6]. First, the input graph G_0 is pre-processed by contracting vertices to obtain a coarser version, G_1, of the graph. This process is usually repeated multiple times, until a small enough version of the graph, G_i, is found. Then, using a high-quality graph partitioning algorithm, a partitioning of G_i is obtained, which is referred to as the initial partitioning. Finally, the graph is uncoarsened by expanding the contracted vertices, which can be done in multiple stages. At each stage the partitioning of the finer graph is refined to finally recover a good partition of G_0. A schematic overview of the three general phases of an MGP algorithm can be found in figure 1.

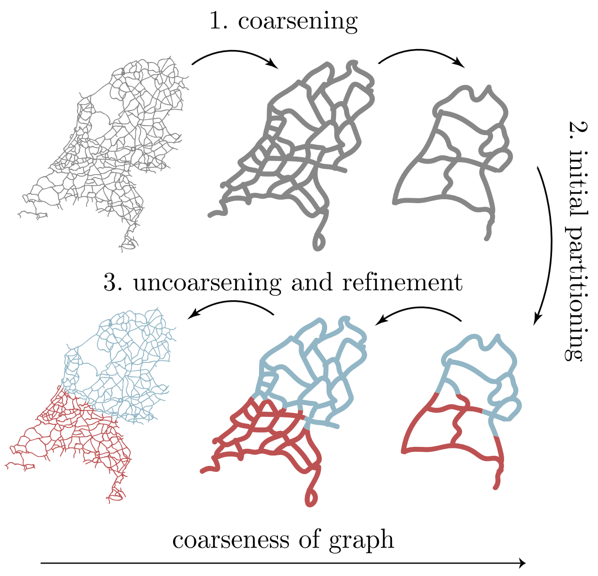


Figure 1: Schematic overview of the three phases of an MGP algorithm applied to a simplified version of the connected component of the Dutch national road network. The partitioning found through an MGP algorithm does not necessarily have the lowest possible edgecut. In this schematic example, the partitionings are also not balanced.

3.1.1 Coarsening Phase

During each step in the coarsening phase, the size of the graph $G_i = (V_i, E_i)$ decreases, such that $|V_i| < |V_{i-1}|$. This is done by merging sets of two or more vertices V_i^v together to form a *multinode* v [16]. The weight of vertex v is the sum of the weights of the vertices in V_i^v, such that during the partitioning phase, v forms an accurate, weighted representation of the contracted vertices. The total vertex weight of the graph, $w_v(G_i)$, is therefore also conserved between the different coarsening levels i. Similarly, if two or more vertices in V_i^v are connected to a vertex w, the weight of the new edge $\{v, w\}$ simply equals the sum of the edges running from V_i^v to w. This ensures that the connection of the contracted vertices with surrounding vertices is accurately represented. The cost of a partitioning of the coarse graph will therefore remain the same when we expand the contracted vertices.

There are various ways in which vertices can be contracted. A popular method is to let edges *collapse*, and to merge the vertices connected by those edges [16]. A *matching* defines which edges in G_i will be contracted [16, 13, 6]. A matching is a set of edges which has the criterion that no two edges share the same endpoint. In each coarsening step, the aim is to find a matching such that no extra edge can be added without breaking this criterion. Such a matching is called a *maximal matching*, and a matching is required to be maximal to ensure that the graph size decreases sufficiently.

Several ways to obtain a maximal matching have been proposed [13, 6, 16]. A straightforward method is to visit vertices in a random order and in the case that a vertex is not yet included in the matching, to see if there is a neighbouring vertex available such that their edge can be included. As each vertex only needs to be visited once, the *Random Matching* [13] algorithm runs in linear time. This algorithm is both simple and efficient, and since it produces maximal matchings, the number of coarsening levels needed to obtain a small enough partition is minimized.

Another algorithm, *Heavy Edge Matching* [16], prioritises the inclusion of edges with high weights in the matching. For each coarsening level, the total edge weight of the graph decreases by the edge weight of the matching. Collapsing edges with high weights therefore results in a coarsened graph with relatively low edge weights, such that the cost of the initial partitioning is also lowered [15].

The graph will be coarsened until G_i becomes too small, or when the size reduction obtained through graph coarsening is no longer significant. If a partitioning in k partitions is desired, there must be at least k vertices available in the coarsened graph to obtain a partitioning. The minimum number of vertices required for the coarsened graph is generally at least an order higher than k, but this varies per algorithm, as different algorithms use different partitioning techniques in the partitioning phase.

3.1.2 Partitioning Phase

In the partitioning phase, an initial k-way partitioning of the coarse graph $G_c = (V_c, E_c)$ is obtained. Since the vertex weights of the vertices in G_c accurately represent the weights of the contracted vertices, finding a partitioning of G_c corresponds to finding a partitioning of the original graph G_0, with the same cost. The partitioning is usually required to be approximately balanced, with an ϵ that is often not higher than 0.05. Since the total vertex weight is conserved between coarsening phases, a balanced partitioning of the coarse graph will correspond to a balanced partitioning of the original graph.

In principle, any graph partitioning algorithm technique can be used for partitioning the coarsened graph G_c, and the algorithm used differs per implementation. The Kernighan-Lin heuristic [17] is an example of an algorithm that can be used for obtaining the initial partitioning of the coarse graph [16]. Since the coarsened graph is small, a KL-partitioning can be computed quickly. Although KL-partitioning may be far from optimal on large graphs, the algorithm can quickly be run multiple times on the coarsened graph, after which the partitioning with the smallest edgecut is selected.

3.1.3 Uncoarsening and Refinement phase

After an initial partitioning has been obtained, the graph can be uncoarsened. This is simply done by expanding the contracted vertices from the matching of each coarsening level. Each set of vertices V_i^v in G_i that was contracted to form vertex v in G_{i+1} is simply assigned the partition P_i to which v belongs.

The coarsened graph G_c might have been partitioned such that the partitioning P_c of the coarsened graph G_c is at a *local minimum*, meaning that there is no vertex or vertex pair for which moving to another partition or swapping partitions will lead to a decrease in the cost of the partitioning. This is for example the case when the initial partitioning is obtained through KL-partitioning. However, even if P_{i+1} is at a local minimum, P_i might not be, since the expansion of the contracted vertices in G_i introduces more degrees of freedom to the graph. Local refinement methods might therefore be able to find a partitioning that has a lower cost than the initial partitioning after uncoarsening.

To obtain locally refined partitionings, vertices will need to be moved between partitions. This is done by swapping pairs of vertices from distinct partitions, such that the partitions remain balanced. The objective is to find a set of vertex pairs, such that swapping them will bring the cost of the partitioning to a local minimum. The Kernighan-Lin heuristic works well for this purpose, since it fullfills exactly this requirement. Variants of the KL-algorithm, such as the Fiducia-Mattheyses (FM) heuristic [9] and adaptations thereof are widely used as local refinement methods, since they produce similar results as the KL-algorithm, within a much smaller run-time.

The uncoarsening and refinement of the graph continues until the original graph G_0 is retrieved and the partitioning P_1 is refined to obtain a final partitioning P_0 of the original, fine graph. The obtained final partitioning is at a *local* minimum, but since it is derived from the partitioning of the coarsened graph, the final partitioning also accounts for the *global* structure of the graph.

The strength of multilevel graph partitioning algorithms lies in a powerful simplification of the graph partitioning problem. Obtaining a partition of a coarsened graph is significantly easier than partitioning the full graph, and between the different uncoarsening levels, a partitioning does not need to be overhauled completely, but merely needs some refinement. This simplification proves to be so efficient that the MGP scheme is widely used in graph partitioning software [10, 2]. The methods for coarsening the graph, creating the initial partitioning, and applying local improvements to the partitioning differ amongst the various software implementations, but the three-step structure can still be observed.

3.2 METIS

Over the past two decades, many multilevel graph partitioning algorithms have been developed. Some are tailored towards providing high-quality partitionings, attempting to find partitions with the smallest possible cost. Others are focused on speed, and return a partitioning of networks with over a million vertices within a second. METIS [16] is a graph partitioning algorithm which attempts to find good partitionings within a short period of time. METIS was initially released in 1995 as one of the earliest multilevel graph partitioning software packages. It therefore follows much of the original structure of the MGP heuristic introduced by Hendrickson and Leland [13]. It is designed to handle large irregular graphs, a category which also includes road networks.

One important feature of METIS is its introduction of the heavy edge heuristic for matching vertices. It currently employs this technique by first sorting the vertices of a graph by their degree. It then selects the vertex with the highest degree that is not yet part of the matching. Then, from the edges that are connected by this vertex, it selects the one with the highest weight that is still available to be included in the matching. This version of heavy edge matching is called *Sorted Heavy-Edge Matching* (SHEM), and is currently one of the most efficient coarsening techniques available [2]. Although currently many other graph partitioning algorithms are available that provide partitionings of a much higher quality than METIS does, METIS remains a popular partitioning algorithm due to its speed.

3.3 Partitioning Road Networks using MGP algorithms

Multilevel graph partitioning algorithms can also be used to partition road networks. The coarsening phase in MGP algorithms introduces a welcome size reduction to national road networks, that often contain millions of vertices and edges. This introduces a significant reduction in computation time.

In principle, MGP algorithms work similarly on road networks as they do on any other network. However, general purpose MGP implementations like METIS do not exploit the characteristic structure of road networks to deliver optimal results. Specialised graph partitioning algorithms for road network partitioning have been developed to deliver high quality partitionings by explicitly making use of this characteristic structure. PUNCH [7] was the first such algorithm, and makes use of *natural cuts* in road networks. Natural cuts are dense regions that are separated by barriers within a network, such as mountains, rivers, or borders.

PUNCH does not closely follow the MGP heuristic, but some of its key concepts are still represented. For example, PUNCH also contracts vertices to obtain a smaller version of the graph. This is done by contracting vertices that do not contribute to natural cuts, such that natural cuts are still represented in the smaller graph, which is subsequently partitioned.

Buffoon [21] is another graph partitioning package that uses natural cuts as a pre-processing technique to coarsen the graph. Although Buffoon uses highly sophisticated techniques to obtain high quality partitionings, it still follows the three-stage structure of classic MGP algorithms.

Buffoon and PUNCH both outperform METIS on partition quality [7, 21]. The natural cut heuristic yields partitionings with significantly smaller costs. In the next section, we will further describe how natural cuts occur in road networks. We will also further explore how the structure of road networks can be exploited to pre-process graphs, for instance to improve the functioning of existing MGP algorithms when applied to road networks.

4 Pre-processing using Edge-Betweenness Centrality Weights

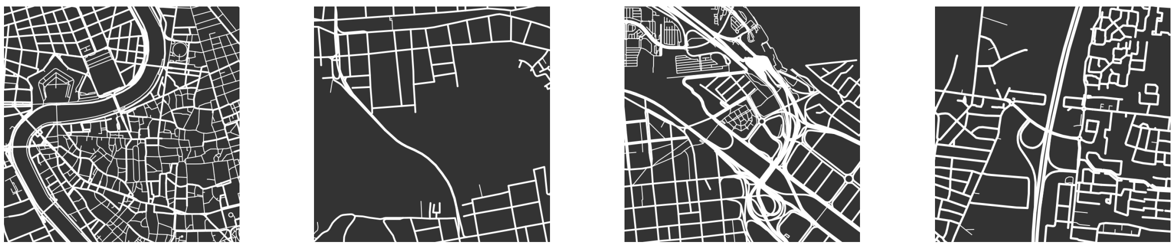
Understanding the anatomy of a network is vital for finding good partitionings. Road networks possess distinct features that may be utilised for optimising graph partitioning schemes. This section will first discuss the distinct properties of road networks. This includes how edge-betweenness centrality can reveal the characteristic role that certain roads play within the network and how this can reveal good cut-edge candidates. We will use this to create a new graph partitioning heuristic. We will then explain how approximate edge-betweenness centrality can be used as a good and easily obtainable approximation of edge-betweenness centrality, which will make our heuristic easier to implement. Finally, we will introduce a simple implementation of our heuristic, which uses edge-betweenness centrality to introduce a weight-bias in road networks. This pre-processing method yields a weighted version of the original graph that can be used in existing graph partitioning software, essentially implementing the heuristic for existing MGP algorithms.

4.1 Natural Cuts and Edge-Betweenness Centrality

The structure of a road network is highly dependent on local natural environments. Roads meander through mountain valleys, get halted by seas and rivers, and avoid deep cliffs and steep climbs. To overcome these natural obstacles, humans have engineered bridges and tunnels. As building these structures remains a costly enterprise, bridges and tunnels are built relatively sparsely.

Next to bodies of water, hills, mountains, and other natural obstacles, road networks are also separated by artificial barriers, such as land borders, or even roads themselves, for example when a large highway divides two suburbs. Barriers in road networks can easily be observed by looking at a road map (see figure 2). Tourist maps of cities like Rome contain a river dividing a city into two parts. Each of the resulting sections often forms a tight-knit road network, but these sections are joined by only a handful of bridges. In this example, a river provides an obvious partitioning of the road network of the city, with the bridges serving as cut-edges.

Identifying barriers in road networks is convenient for partitioning road networks, since they can provide a natural outline to partition along. A bootstrap partitioning of cities like



(a) The river Tiber and its bridges in Rome (Italy)

(b) Noel Rosa Tunnel underneath the hills of Rio de Janeiro (Brasil)

(c) The border crossing near Tijuana (Mexico - United States)

(d) The A27 highway separating Laren and Eemnes (Netherlands)

Figure 2: Examples of natural and artificial barriers in road networks, causing different areas to be connected through few roads. The roads traversing these barriers can be shown to have high edge-betweenness centralities. Images produced using OSMnx [3].

Paris, Budapest, or Istanbul can quickly be obtained by marking the two riverbanks as distinct partitions. Another example is the dense urban area surrounding the Mexican-US border at the West Coast of the North-American continent. Although over 21 million cars crossed the border between San Diego and Tijuana in the year 2015 [23], there are only two border crossings within this area, which would therefore serve as good cut-edges for partitioning the region. As natural and artificial barriers separate different areas in a road network, the sparsely built roads that overcome these barriers naturally serve as good cut-edge candidates, and are therefore *natural cuts*.

Natural cuts are sets of relatively sparse cut-edges connecting denser areas [7], and are therefore important connections within a road network. Roads represented by these edges often handle large volumes of traffic. When a road network is represented as a graph, the traffic flow in a road segment can be roughly approximated by the betweenness centrality of the corresponding edge [18]. Generally, natural cuts can be characterised by having high edge-betweenness centralities (EBC). When a road network is split into parts by natural barriers, many shortest paths between vertices from distinct sections of the graph will go through edges traversing those barriers, which significantly contributes to the EBC of those edges.

Not only natural cuts, which overcome natural and artificial barriers within a road network are good candidates to be cut-edges for forming partitionings, but other edges with high betweenness centralities can also qualify as good cut edges. An example of such edges are motorways that connect two cities. Cities are represented by centres with a high edge-density. The edge-density gradually decreases further from the city centre. Different cities are often connected through only a handful of motorways. Since shortest paths between pairs of vertices from different cities will likely run through the motorways connecting these cities, the edges corresponding to these motorways will have high betweenness centralities. These edges also provide good cut-edges. Consider the simple example of a road network consisting of two cities connected by a highway. A 2-way partitioning P with $cost(P) = 1$ is easily found by cutting through the motorway and assigning the vertices of each city to a distinct partition. Although highways can therefore be good candidates for being cut-edges, they are not natural cuts, since they are not necessarily directly adjacent to densely populated regions. The transitioning of dense urban centres to highways can instead be gradual.

In contrast, any partitioning that contains cuts within either of the city centres would have a high cost, since having the partition boundaries in these areas would entail cutting through many edges in these dense areas. Moreover, the edge-betweenness centrality of edges within the densely populated areas is smaller than that of the highway.

In the simple example of the two cities, any shortest route between the two cities will go through the highway and removing this edge will highly disrupt the network. The removal of edges within the dense areas is less likely to cause similar disruptions.

4.2 The HEB-cut Heuristic

Instead of only identifying natural cuts for graph pre-processing, we can extend the set of cut-edge candidates to include all edges with high betweenness centralities. These cut-edge candidates will be referred to as *HEB-cuts* (***H****igh* ***E****dge-***B****etweenness centrality* ***cuts***). Many natural cuts in a road network will also be HEB-cuts, as these edges often have high betweenness centralities.

We can make use of HEB-cuts in multilevel graph partitioning, for example by including edges with low betweenness centralities in matchings of vertices in the coarsening phase. We can let these edges collapse early on in the MGP process. In contrast, HEB-cuts will more likely be represented in the coarsened graph. The initial partitioning of the coarse graph will therefore be formed by cutting through relatively many HEB-cuts. In the refinement phase, the borders of the partitions may be further refined, but edges with high-betweenness centralities might provide good outlines for obtaining a bootstrap partitioning and are therefore valuable to retain amongst the different coarsening levels.

An analysis of the isolated road network of the city of Utrecht (the Netherlands)[1] shows that the edges with the highest betweenness centrality are highways (A2, A12, A27), underpasses (Lageweideviaduct) and bridges (Vleutenseweg), which are indeed all good cut-edge candidates. Moreover, edges with the lowest betweenness centralities all lie within residential neighbourhoods, where vertices can easily be contracted and where cutting would be expensive. This agrees with our motivations for proposing the HEB-cut heuristic.

The notion of using edges with high betweenness centrality to pre-process graphs might not only be useful for road networks, but can also be extended to other types of networks that share the same characteristics of road networks. In essence, the technique is based on introducing a bias for edges with low edge-betweenness centrality to be contracted during the coarsening phase of an MGP algorithm. In principle this technique can be used to pre-process any graph, but this bias might prove not to be advantageous for some types of graphs.

4.3 Approximate Edge-Betweenness Centrality

Although using edge-betweenness centrality might provide us with a good pre-processing method for obtaining good cut-edges, it is expensive to compute. The fastest known algorithm for calculating the betweenness centrality of an edge, proposed by Brandes [4], requires $\mathcal{O}(nm)$ time for unweighted graphs. Since we are trying to use edge-betweenness centrality to approximate which roads fulfil important connecting functions within a network, we could alternatively focus on rich data about the network that is already available. Many road networks are extensively classified, and the function of each road can be determined through resources such as OpenStreetMap. By doing so, important roads such as bridges, highways, and tunnels can quickly be identified without any expensive calculations. However, this requires extra data about the road network to be available, which might often not be the case.

Brandes and Pich [5] proposed a strategy to approximate betweenness centrality. Instead of computing the complete edge-betweenness centrality by comparing shortest paths for all $\frac{n(n-1)}{2}$ possible vertex pairs in a graph, edge-betweenness centrality can be approximated by only computing the shortest paths for a small sample of vertex pairings $\{v, u\}$ with $v, u \in K$ such that $K \subset V$. We select vertices uniformly at random since this has been shown to be more effective than strategic vertex selection [5]. The approximate betweenness centrality of an edge simply becomes

$$EBC_{approx}(e) = \frac{1}{n(n-1)} \sum_{v \neq u \in K, e \in E} \frac{\sigma(v, u|e)}{\sigma(v, u)}, \tag{1}$$

which can be computed in $\mathcal{O}(|K|m)$ time.

Approximate betweenness centrality using random sampling performs well when used on road networks [1]. The quality of the approximation increases with the sample size.

[1]OpenStreetMap data obtained and analysed through OSMnx[3]

However, for the purpose of identifying HEB-cuts, a rough approximation will suffice. In a large road network, randomly chosen vertices are unlikely to all be centred within a small area of the graph. We expect the chosen vertices to be spread out over the network. Shortest path calculations are therefore likely to reveal much of the general anatomy of the network, including important HEB-cuts. Shortest paths between different cities will likely run over highways and natural cuts, but are unlikely to include many unimportant suburban roads and cul-de-sacs.

Approximate betweenness centrality of edges in denser regions that we wish to contract is less likely to be a good reflection of the real betweenness centrality of these edges, since much of the contribution to their EBC comes from shortest paths between neighbouring vertices within these regions. The probability that specifically these vertices will be included in the sample is low, such that the approximate betweenness centrality for these edges is likely to be an underrepresentation of their real betweenness centrality. This, however, is not problematic, since we would like to contract these edges anyway. Moreover, we choose to not reflect the distinction between high and low edge-betweenness centralities in the matching phase too heavily. This means that we do not exclusively collapse edges with low betweenness centralities, but that we merely make them more susceptible to collapse. This way, edges with higher centralities can still be collapsed occasionally, and conversely, some edges with low edge-betweenness centralities will still exist in the coarsened graph.

4.4 EBC pre-processing in Existing MGP Algorithms

Introducing a new MGP algorithm that makes use of the edge-betweenness centrality heuristic is beyond of the scope of this thesis. Instead, we propose a method by which road networks can be pre-processed to improve the quality of partitionings found by existing general purpose MGP implementations. Furthermore, we hope to inspire further research into utilising edge-betweenness centrality in graph partitioning algorithms.

In the context of multilevel graph partitioning, the edge-betweenness centrality metric can be used to distinguish between edges that are potential cut-edges, which we wish to carry over to the coarsened graph, and edges that are of little importance to the structure of the graph, which we like to contract during the coarsening phase. We can take advantage of the Sorted Heavy Edge Matching matching algorithm, that is for example used in METIS, to achieve this goal. When a road network is initially translated into a graph, it does not possess edge weights, or equivalently; all edge weights are unitary. By instead assigning high edge weights to edges with a low edge betweenness centrality and low edge weights to edges with a high betweenness centrality, we prioritise the inclusion of edges with a low betweenness centrality in the matching, which will be collapsed as a result. Edges with a high betweenness centrality are therefore more likely to survive the coarsening phase and are consequently better represented in the initial partitioning.

Although two edges might be equally suitable candidates to be used as cut-edges for a partitioning, the EBC of two edges is seldom equal. Moreover, when EBC is approximated, the importance of certain edges can be gravely misrepresented. Edges that are marked as HEB-cuts when EBC is calculated using all vertices in V might not be found when only a small vertex sample is taken. In an extreme case, a region with a substantial number of vertices might be linked to the rest of the graph by only one connecting edge, which would therefore in theory be an excellent cut-edge. However, if the vertices in the vertex sample all happen to fall outside of this region, all edges within this region will have an approximate

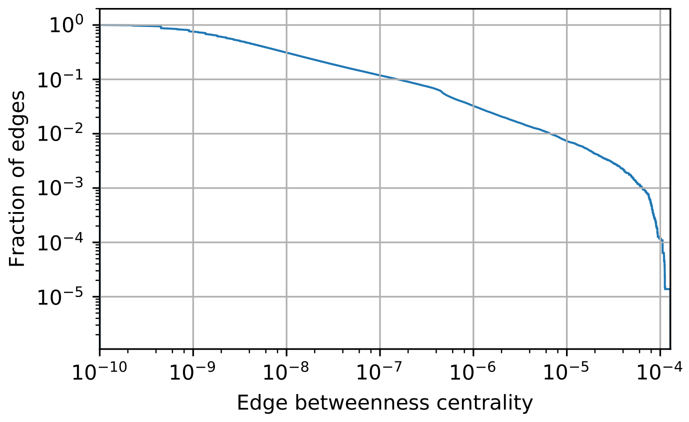


Figure 3: Cumulative distribution function for the normalised approximate betweenness centrality of edges in the national road network of the Netherlands (see table 1). The graph indicates which fraction of the total number of edges is above a certain EBC value. The log-log relationship found here indicates a power-law structure [20]. For this approximation, the vertex sample consisted of 2217 vertices, which corresponds to approximately 0.1% of all vertices.

betweenness centrality of zero, and the connecting edge will not be identified as a HEB-cut.

To avoid creating too much distinction between edges with similar cut-edge suitability, and to reduce the effect of misrepresented EBCs when the chosen vertex sample is small, we translate EBCs to edge weights in a discrete manner, meaning that all edges within a specific range of EBC will be assigned the same weight. Consequently, there will not be much contrast in the weights of HEB-cut edges that fulfil comparably important functions in the graph, but these edges are still distinguished from unimportant edges with low EBCs residing within deeply connected areas of the graph.

The edge-betweenness centrality of edges in a road network follows a power-law structure [18]. A power-law can also be observed when EBC is approximated (see figure 3). Only a small portion of the edges has a high EBC, and a large portion of the edges in the graph exists within denser regions. Although the EBC of two edges might be an order of magnitude apart, they might be equally important as HEB-cuts. When we assign these edges weights, these should therefore not be magnitudes apart. Moreover, edges that have EBCs of the same order of magnitude should be assigned the same weight.

We propose a mapping of edge betweenness centrality to discrete edge weights in the following form:

$$w_e(e) = \left\lceil \log_b\left(\frac{1}{EBC(e)}\right) \right\rceil, \tag{2}$$

where $EBC(e)$ can be either exact or approximate EBC, and where $b \geq 1$ is the *base factor*.

The base factor determines the level of discretisation and controls how far edge weights will lie apart. A lower base factor leads to more contrast in edge weights, such that edges with different betweenness centralities are more likely to be assigned different edge weights.

When EBC is approximated, there will be edges for which $EBC(e) = 0$, such that equation (2) becomes invalid. To work around this problem, we simply assign the highest weight in the graph to these edges, since although technically possible, these edges are unlikely to be

HEB-cuts if the EBC would have been calculated exactly. The edges that fulfil important connecting functions within the graph can already be revealed by a low vertex sample. In contrast, edges of which the approximate EBC equals zero are much more likely to lie in those regions we wish to contract.

A major advantage of pre-processing the graph using EBC weights separately from the MGP algorithm is that this procedure only needs to be carried out once. As soon as a weighted version of the graph has been obtained, this can be used as input for METIS or other MGP algorithms. To calculate many different k-way partitionings, METIS can be run on the newly acquired weighted graph without further ado, such that this pre-processing only needs to happen once to acquire partitionings for a large spectrum of values of k.

In the next section, we will apply the HEB-cut pre-processing heuristic for multiple instances of road networks. We will try to determine good values for the contrast factor and analyse in which cases the HEB-cut heuristic can improve the partitioning cost for partitions obtained through METIS.

5 Experiments

5.1 Methodology

We will test the effectiveness of the HEB-cut heuristic for multiple national and continental road networks (see table 1). After a suitable base factor is found, weights will be assigned to graphs using equation (2) for various EBC vertex samples and sample sizes. Then, partitionings of the unweighted and weighted versions of these graphs will be obtained using METIS [16]. Note that METIS returns the same partitionings when it is run multiple times for the same graph with the same settings. For our computations, we will use the default settings of METIS. For all experiments, we require a maximum imbalance of 3%, therefore setting $\epsilon = 0.03$. Since the weights on the weighted graph merely serve the purpose of introducing a bias in METIS, we will compare the edgecut instead of the cost of the weighted and unweighted partitioning.

Network	n	m
Asia	11950757	12711603
Belgium	1441295	1549970
Germany	11548845	12369181
Great Britain	7733822	81556517
Italy	6686493	7013978
Luxembourg	114599	119666
Netherlands	2216688	2441238

Table 1: The road networks used in these experiments. These networks were made available as part of the 10th DIMACS Implementation Challenge and can be found online at `http://www.cc.gatech.edu/dimacs10/archive/streets.shtml`.

We use METIS for its high computation speed and its use of the SHEM algorithm. Although using a different MGP algorithm might produce different results, we simply try to investigate whether the HEB-cut heuristic can contribute to the field of multilevel graph partitioning. If this heuristic demonstrates to provide an improvement in the edgecut in METIS, this is indicative that identifying HEB-cuts is a useful technique for multilevel graph partitioning.

For ease of use, EBC is calculated using NetworkX [12], which uses the $\mathcal{O}(|K|m)$ adaptation of the algorithm by Brandes [4]. When EBC is approximated, instead of normalising EBC values by $\frac{1}{n(n-1)}$, n can also be replaced by the size of the set of sampled vertices. However, we use the normalisation by $\frac{1}{n(n-1)}$ that is default in NetworkX, which should not influence our results, since the relative difference in edge-betweenness centrality remains independent of this scaling factor.

For small vertex sample sizes in graphs with relatively few vertices (e.g. Luxembourg, the Netherlands, or Belgium), approximate EBC can be calculated using a consumer laptop. For larger networks and vertex sample sizes, we use Cartesius, the Dutch national supercomputer, such that EBC calculations can be performed in parallel, with sufficient memory available.

We will not report any calculation times, since we have not been able to set up a systematic computing environment for this purpose. Instead, we will only investigate whether the HEB-cut heuristic can provide improvements to the quality of partitionings of road networks

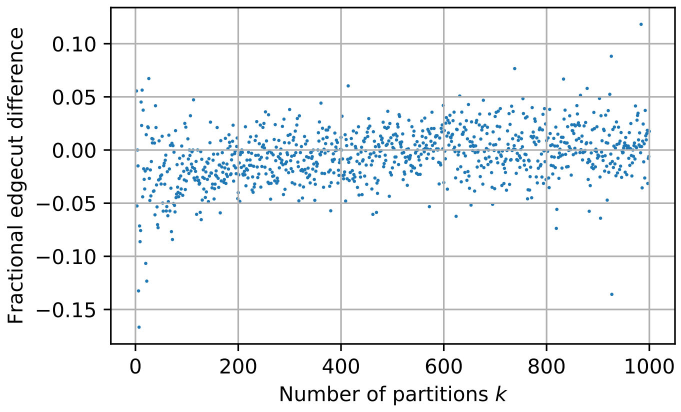


Figure 4: Improvement in edgecut of the weighted graph with respect to the edgecut of the unweighted graph of Luxembourg for $b = 10$, with a sample of 11460 vertices. Negative values indicate an improvement when using the HEB-cut heuristic.

obtained through METIS.

5.2 Main Results

5.2.1 Base Factor

For testing our heuristic, we first need to establish which value of b provides good results. To investigate this, we perform tests using the road network of Luxembourg, since it can be partitioned quickly. We calculate the edge betweenness centrality using 11460 vertices, a sample which accounts for approximately 10% of the country's road network. The approximated EBC then takes on a maximum value of 3.255×10^{-2} and the smallest nonzero EBC is 2.538×10^{-11}. An initial guess of $b = 10$ would then produce a discretisation into 6 weights, ranging from 5 to 11.

To analyse the effectiveness of our choice of base factor, we partition the weighted and unweighted graph in k partitions, where k ranges from 2 to 1000, and we compare the edgecut of the two graphs for each value of k, the results of which are shown in figure 4. The fractional edgecut difference indicates the fraction of cut-edges obtained through METIS by which the HEB-cut heuristic improves or worsens the partitioning quality when compared to the unweighted graph. For example, a fractional edgecut difference of -0.10 implies that the HEB-cut heuristic yields a decrease in the amount of cut-edges by 10%. As can be seen, the performance of the HEB-cut heuristic cannot easily be examined, as the fractional difference in edgecut between the weighted and unweighted graph is not consistent between different values of k.

To properly study possible settings for b, we will instead look at the *cumulative* fractional edgecut difference. For each value of k, we add the fractional improvement of all previous values of k. By doing so, we are able to discern trends in the performance of the HEB-cut heuristic. A decrease in the cumulative fractional edgecut difference implies that the HEB-cut heuristic generally improves the quality of partitionings, whereas an increase means it performs worse.

We compare the values 2, 10, and 100 for the base factor when obtaining partitionings

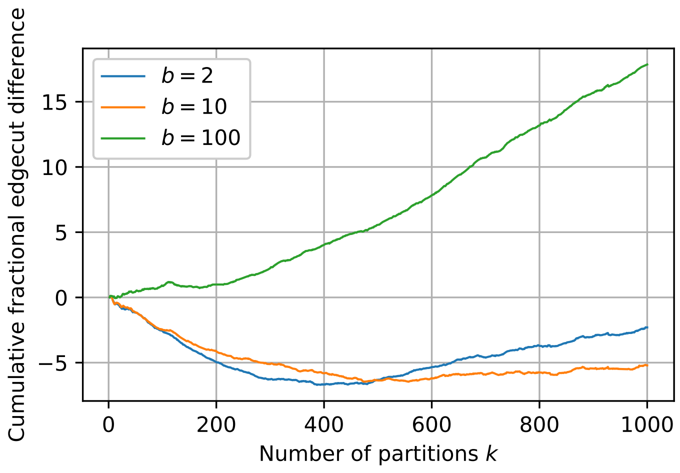


Figure 5: Comparison of different values for b when partitioning Luxembourg using 11460 sample vertices. The fractional edgecut difference is taken with respect to the edgecut of the unweighted graph.

of Luxembourg with the same 11460 sample vertices (see figure 5). Indeed our initial choice of $b = 10$ seems to perform well. Initially $b = 2$ (weights ranging between 15 and 36) does a better job, but at around $k = 300$, the improvements of using the HEB-cut heuristic seem to be cancelled out by worse partitionings provided by this technique. After this, using the HEB-cut heuristic leads to a worse performance than METIS does out-of-the-box. The value $b = 100$ (weights ranging between 3 and 6) does an overall poor job, and we would be better off not assigning edge weights for a large number of values of k. When we choose $b = 10$, HEB-cut heuristics on average improve the partitioning up until around $k = 500$. Thereafter, the behaviour for higher values of k remains relatively stable. Further experiments will therefore use a base factor of $b = 10$.

5.2.2 Variation in Approximate EBC

Since approximate edge-betweenness centrality is calculated using vertices chosen uniformly at random, there will be a variation in the edge betweenness centrality of different edges. Although the discretised assignment of edge weights will cause many edges to still be assigned the same weights, we want to find out to what extent the partitioning quality will vary for repeated EBC calculations with the same sample size but with different sample vertices.

The variation in partition quality for different EBC vertex samples should become smaller for higher EBC vertex sample sizes, as larger samples provide more reliable approximations of the exact EBC. However, we would like to be able to use small values in the HEB-cut heuristic as well such that we can do our calculations relatively fast. With this in mind, we looked at the variation of partition quality for different vertex samples of a relatively small size.

We computed partitionings of Luxembourg for various graphs which were weighted using an EBC vertex sample of 100 vertices, which amounts to approximately $0,087\%$ of the vertices in Luxembourg. The fractional difference in edgecut of the different partitionings of the weighted graph behaves erratically. We observed that oftentimes, while some of the weighted graphs are partitioned worse than the unweighted graph for a certain k, others perform better and vice-versa. However, when calculating the cumulative fractional edgecut difference, the overall behaviour of the differently weighted graphs becomes more visible, and we observe that although there is quite some variation in the performance for different samples, these samples do follow similar trends (see figure 6).

We conclude that when EBC is approximated, the quality improvement gained by the HEB-cut heuristic heavily depends on the sample vertices. This makes drawing quantitative conclusions about the HEB-cut heuristic difficult, unless sufficiently many samples are considered, but if for all different values for k the HEB-cut heuristic would improve the partitioning quality, this would be enough evidence for qualitative conclusions about the effectiveness of HEB-cut heuristics.

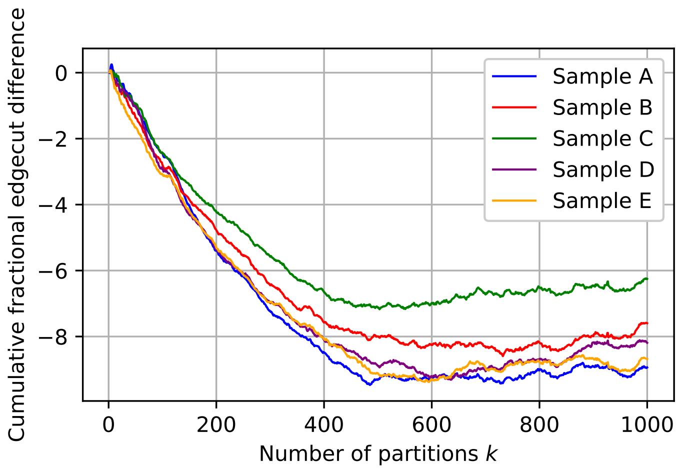


Figure 6: Comparison of the partitioning quality using EBC calculated using five different sets of 100 vertex samples in the case of Luxembourg.

5.2.3 Vertex Sample Size

When approximating edge-betweenness centrality, smaller vertex sample sizes mean that the approximate EBC values can be calculated much faster. A smaller vertex sample, however, might sacrifice in quality of a partitioning, since higher vertex sample sizes provide better approximations of the exact EBC. We studied the effect of different vertex sample sizes on the partition quality.

For the road network of Luxembourg, we studied vertex sample sizes of 100 and 11460 vertices, which corresponds to approximately 0.087% and 10% of n respectively. We also studied the road networks of the Netherlands. In this case, EBC calculations become too costly for high vertex sample sizes, so we studied vertex sample sizes of 100, 400, and 2217 vertices, which translate to approximately 0.005%, 0.02%, and 0.1% of the total graph size.

We discovered no noticeable edgecut reductions as the size of the vertex samples increases. If, theoretically, there would be any such improvements, they are likely to be overshadowed by the variations in quality caused by the random nature of EBC vertex samples. Due to computational limitations, we have not been able to average our calculations over multiple EBC calculations. Doing so can potentially reveal variations in edgecut for different vertex sample sizes, but these calculations are costly.

We conclude that if these variations do exist, they are thus likely to be small. Moreover, the fact that edge weights are assigned in a discrete fashion contributes to the mitigation of these variations. Since a small EBC vertex sample already reveals the most important HEB-cut edges in the graph, an increase of the EBC sample size will further pronounce the importance of these edges, but not necessarily introduce many new HEB-cuts. Hence, the distinction between these and unimportant edges will be visible for both small and large EBC sample sizes. Although possible, it is unlikely that a potentially good HEB-cut remains hidden because of an unfortunate selection of sample vertices in the case the sample is too small.

5.2.4 HEB-cut Heuristics for Different Road Networks

We have calculated the edgecut of partitionings of different road networks (see table 1), for different vertex sample sizes. During these experiments, we calculated the edgecut for different values of k, where we took k to be different powers of 2, as well as multiples of 50. We investigate both, such that we get a good picture of the performance of the HEB-cut heuristic in both a exponentially and linearly growing context.

For most calculations, we have only been able to use one EBC sample, since the calculation of EBC becomes more expensive for graphs with large amounts of edges and vertices. In spite of that, bear in mind that our goal is not to optimise partitionings using HEB-cut heuristics and use this technique to find partitionings with a lowest possible edgecut, but that we want to show that the HEB-cut heuristic can be useful within the field of multilevel graph partitioning. Additionally, we saw in subsection 5.2.2 that there is variation in partitioning quality when different EBC vertex samples are used, but nonetheless do we see the same trends in the effectiveness of the heuristic for each EBC vertex sample. Therefore, not having an average of samples can still provide us with useful, qualitative insights. If the HEB-cut heuristic provides an improvement for almost all values of k for our randomly chosen vertex samples, we can conclude it to be a useful heuristic.

Since calculating approximate betweenness centrality can get expensive, we have limited ourselves to only two vertex sample sizes for each network. Results of these experiments can be found in appendix A. We compared the edgecut of the partitionings found when the HEB-cut heuristic is applied with partitionings found without this heuristic.

For low values of k, the results are a little mixed, but for all graphs with the exception of Luxembourg, applying the HEB-cut heuristic leads to a reduction in edgecut for most values of k. Confirming our earlier observations in subsection 5.2.3, it is hard to notice any difference in the performance of the HEB-cut heuristic for the two different vertex sample sizes.

A possible reason for the mixed results in the case of Luxembourg might be its size, which could be important for the effectiveness of using the HEB-cut heuristic. The HEB-cut heuristic is based on the existence of natural cuts within the network, as well as the existence of larger dense regions that are connected with one another through only a few edges. In small road networks, such as that of Luxembourg, the HEB-cut heuristic might prove to be inefficient. The heuristic tries to identify sparse and natural cuts, but in the case of Luxembourg, most edges in the network exist within the dense eponymous capital city. Although the edges will still have a spectrum of different EBC values, high EBC edges might not reflect roads that would indeed be good cut-edge candidates, since HEB-edges can also lie within dense regions, where they might not be good cut-edge candidates. Moreover, there are simply not many

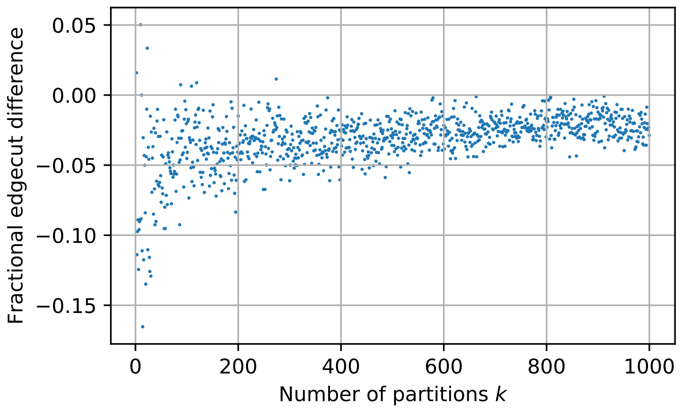


Figure 7: Fractional edgecut difference of partitionings obtained through the HEB-cut heuristic compared to partitionings obtained from unweighted graphs. In this case the HEB-cut heuristic has been applied to the Netherlands with an EBC sample size of 100 vertices.

natural cuts available in the network, such that it makes sense that our heuristic does a poor job in identifying them, especially for high values of k.

We can confirm the positive effect of the HEB-cut heuristic on larger graphs, for example by considering the road network of the Netherlands and looking at the fractional edgecut difference for individual values of k, where we let k range between 2 and 1000 (see figure 7). For an EBC vertex sample size of only 100 vertices, the HEB-cut heuristic improves partitionings in all but 7 cases. Partitionings are improved by around 3% on average.

This is remarkable, since a sample size of 100 vertices corresponds to only 0.005% of V. For all road networks, except Luxembourg, the HEB-cut heuristic is effective when few vertices are used as EBC vertex samples, only representing a small fraction of the total amount of vertices in each graph. This indicates that in large enough road networks, only a very small sample of vertices reveals many of the HEB-cuts in the network, proving our HEB-cut heuristic to be powerful.

5.2.5 Comparison with Buffoon

Since the aim of our paper is to study whether the HEB-cut heuristic can be useful within the context of MGP, we compare the performance of our simple implementation of this heuristic to that of Buffoon [21]. Buffoon is a sophisticated MGP algorithm that uses natural cuts for coarsening the graph.

We specifically study the case of the Netherlands. Alongside the edgecut of partitionings found using the HEB-cut heuristic and the original unweighted graph, table 8 in appendix A also contains the edgecut found by Buffoon.

We conclude that although Buffoon manages to find much better partitionings than our implementation of the HEB-cut heuristic, the improvements brought about by our heuristic are not negligible. More sophisticated implementations of this heuristic might therefore yield edgecut improvements that come closer to those obtained by Buffoon.

We re-iterate that although the computation of EBC is costly, this does only need to be done once to obtain a weighted version of the graph. The weighted graph can then be used to obtain any number of k partitions, without having to re-calculate EBC. Since METIS is

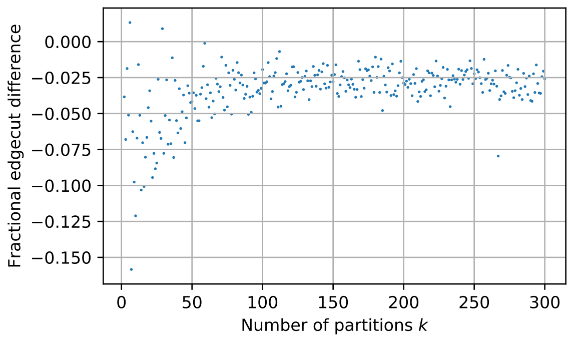


Figure 8: Fractional edgecut difference of partitionings obtained through the HEB-cut heuristic when applied to the graph `bcsstk29`, using EBC calculated from 140 sampled vertices.

much faster than Buffoon, this is where the advantage of our implementation of the HEB-cut heuristic lies: when partitionings need to be calculated for sufficiently many values of k, the fast performance of METIS might balance out the lengthy EBC calculation.

5.2.6 HEB-cut Heuristics for Other Graphs

This thesis proposes the HEB-cut heuristic for finding solutions to the road network partitioning problem. Nevertheless, HEB-cut heuristics may also prove to be useful for partitioning other types of graphs. This assumption is especially tenable when the structure of a graph resembles that of a road network; near-planar with low vertex degrees and highly connected clusters connected to each other by only a handful of edges. Nonetheless would we like to see how the heuristic performs on graphs that possess different structures.

Preliminary experiments on different types of graphs from a widely used graph benchmark set [24] provides mixed results. We conducted the experiments with the same parameters and values of k as chosen for road networks. For some graphs, the HEB-cut heuristic worsens the partition quality in almost all values of k. However, for many graphs, the results are fluctuating, significantly improving the partitioning quality for certain values of k while worsening it for others. These results are reminiscent of the behaviour found for the road network of Luxembourg.

For one particular graph, `bcsstk29` ($n = 13992$, $m = 302748$), our HEB-cut implementation significantly improves the partitioning quality for almost all values of k initially tested for. The graph in question represents a large eigenvalue problem found in structural engineering, specifically a buckling model of a Boeing 767 rear pressure bulkhead. We further analysed the decrease in edgecut after applying our HEB-cut heuristic implementation for k ranging from 2 to 300, the results of which can be found in figure 8. Although the graph corresponding to this problem has a structure that is in many ways different than that of a road network, the HEB-cut heuristic still manages to improve the edgecut of the partitioning obtained by METIS by around a remarkable 4% on average, and is effective for almost all values of k.

As with road networks, further refinement of the implementation might improve the efficiency of the heuristic when applied to other types of networks. Moreover, further enquiry into the HEB-cut heuristic for other types of graph may contribute to making it a viable pre-processing method within MGP algorithms.

6 Conclusion

We have proposed a new pre-processing heuristic for partitioning road networks. This heuristic uses approximate edge-betweenness centrality to identify good cut-edge candidates, so-called HEB-cuts. This heuristic can be implemented for existing multipurpose MGP algorithms, such as METIS, by assigning discretised edge weights to graphs. Edges with a low EBC are prioritised to be included in the matching and will therefore be collapsed during the coarsening phase, but HEB-cuts are less likely to be contracted. As a result, relatively many HEB-cuts will survive the coarsening phase and are represented in the coarse graph.

Our implementation of the HEB-cut heuristic improves the quality of partitionings in large road networks. Only a small sample of vertices needs to be included in the approximate EBC calculation to yield good results, which alleviates the calculation time needed to obtain a good EBC sample.

Although more advanced MGP algorithms, possessing specialised routines for partitioning road networks, perform much better than our HEB-cut heuristic implementation, we do show that the novel HEB-cut heuristic can successfully improve partitioning quality.

The unpolished implementation of the HEB-cut heuristic as presented here can further be refined to obtain better results. This could for instance be done by further studying a good configuration for the base factor, or by embedding the HEB-cut heuristic within a fine-tuned MGP algorithm.

Moreover, the application of the HEB-cut heuristic to networks other than road networks can further be researched. As our preliminary research shows that extensions of the HEB-cut heuristic to different types of networks might often not yield fruitful results, it is worthwhile looking into for which graphs the edgecut can be improved using the HEB-cut heuristic.

We hope that this thesis inspires further research into HEB-cut heuristics for these and other applications.

References

[1] David A Bader, Shiva Kintali, Kamesh Madduri, and Milena Mihail. Approximating betweenness centrality. In *International Workshop on Algorithms and Models for the Web-Graph*, pages 124–137. Springer, 2007. 10

[2] Charles-Edmond Bichot and Patrick Siarry. *Graph partitioning*. John Wiley & Sons, 2013. 6, 7

[3] Geoff Boeing. OSMnx: New Methods for Acquiring, Constructing, Analyzing, and Visualizing Complex Street Networks. 2017. Manuscript under review. doi:10.2139/ssrn.2865501. 8, 10

[4] Ulrik Brandes. A faster algorithm for betweenness centrality. *Journal of mathematical sociology*, 25(2):163–177, 2001. 10, 14

[5] Ulrik Brandes and Christian Pich. Centrality estimation in large networks. *International Journal of Bifurcation and Chaos*, 17(07):2303–2318, 2007. 10

[6] Thang Nguyen Bui and Curt Jones. A heuristic for reducing fill-in in sparse matrix factorization. In *6th SIAM Conf. Parallel Processsing for Scientific Computing*, pages 445–452, 1993. 1, 4, 5

[7] Daniel Delling, Andrew V. Goldberg, Ilya Razenshteyn, and Renato F. Werneck. Graph partitioning with natural cuts. *Proceedings - 25th IEEE International Parallel and Distributed Processing Symposium, IPDPS 2011*, pages 1135–1146, 2011. 1, 3, 4, 7, 9

[8] Daniel Delling, Peter Sanders, Dominik Schultes, and Dorothea Wagner. Engineering route planning algorithms. In *Algorithmics of large and complex networks*, pages 117–139. Springer, 2009. 1

[9] Charles M Fiduccia and Robert M Mattheyses. A linear-time heuristic for improving network partitions. In *Papers on Twenty-five years of electronic design automation*, pages 241–247. ACM, 1988. 1, 6

[10] Per-Olof Fjällström. Algorithms for graph partitioning: A survey. *Linköping Electronic Articles in Computer and Information Science*, 3(10), 1998. 1, 2, 6

[11] Michael R Garey, David S. Johnson, and Larry Stockmeyer. Some simplified np-complete graph problems. *Theoretical computer science*, 1(3):237–267, 1976. 3

[12] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy2008)*, pages 11–15, Pasadena, CA USA, August 2008. 14

[13] Bruce Hendrickson and Robert W Leland. A multi-level algorithm for partitioning graphs. *SC*, 95(28), 1995. 1, 4, 5, 7

[14] David S Johnson, Cecilia R Aragon, Lyle A McGeoch, and Catherine Schevon. Optimization by simulated annealing: an experimental evaluation; part i, graph partitioning. *Operations research*, 37(6):865–892, 1989. 1

[15] George Karypis and Vipin Kumar. Analysis of multilevel graph partitioning. In *Proceedings of the 1995 ACM/IEEE conference on Supercomputing*, page 29. ACM, 1995. 5

[16] George Karypis and Vipin Kumar. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998. 1, 2, 4, 5, 6, 14

[17] Brian W Kernighan and Shen Lin. An efficient heuristic procedure for partitioning graphs. *The Bell system technical journal*, 49(2):291–307, 1970. 1, 5

[18] Stefan Lämmer, Björn Gehlsen, and Dirk Helbing. Scaling laws in the spatial structure of urban road networks. *Physica A: Statistical Mechanics and its Applications*, 363(1):89–95, 2006. 9, 12

[19] Amir Meshkat and JLM Vrancken. Multi-objective road network partitioning. *Procedia-Social and Behavioral Science*, 2014. 1

[20] Mark Newman. *Networks: an introduction.* Oxford University Press Inc., 2010. 12

[21] Peter Sanders and Christian Schulz. Distributed evolutionary graph partitioning. In *2012 Proceedings of the Fourteenth Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 16–29. SIAM, 2012. 4, 7, 19

[22] Thang Nguyen Bui and Byung Ro Moon. Genetic algorithm and graph partitioning. *IEEE Transactions on Computers*, 45(7):841–855, 1996. 1

[23] United States Department of Transportation, Bureau of Transportation Statistics. Border crossing/entry data: Query detailed statistics. `https://transborder.bts.gov/programs/international/transborder/TBDR_BC/TBDR_BCQ.html`, Dec 2016. 9

[24] Chris Walshaw. Benchmarks for graph partitioning. `http://chriswalshaw.co.uk/partition/`. 20

[25] Feng Xie and David Levinson. Measuring the structure of road networks. *Geographical analysis*, 39(3):336–356, 2007. 2

A Test Results

Asia			
k	HEB-cut heuristic edgecut		unweighted edgecut
	12 sample vertices	120 sample vertices	
2	11	7	9
4	21	24	23
8	71	77	73
16	165	157	162
32	337	321	335
64	589	579	611
128	1055	1047	1108
256	2149	2080	2134
512	4110	4077	4191
1024	7773	7784	8001
50	478	463	516
100	874	843	881
150	1225	1228	1290
200	1607	1570	1714
250	2030	2007	2116
300	2478	2439	2494
350	2836	2829	2959
400	3223	3268	3347
450	3573	3583	3739
500	3994	4016	4102
550	4373	4381	4489
600	4750	4805	4813
650	5175	5111	5296
700	5559	5518	5655
750	5884	5868	5936
800	6155	6147	6265
850	6517	6547	6697
900	6884	6925	6938
950	7267	7236	7386
1000	7628	7668	7752

Table 2: Edgecut of partionings of the road network of Asia.

Belgium			
k	HEB-cut heuristic edgecut		unweighted edgecut
	144 sample vertices	1441 sample vertices	
2	99	89	97
4	224	209	241
8	441	435	435
16	703	724	768
32	1142	1139	1234
64	1842	1838	1825
128	2788	2819	2930
256	4345	4349	4477
512	6564	6563	6786
1024	9895	9871	10151
50	1551	1531	1608
100	2382	2407	2452
150	3132	3164	3231
200	3705	3764	3869
250	4293	4239	4454
300	4777	4818	4931
350	5317	5254	5397
400	5712	5730	5775
450	6168	6145	6338
500	6599	6463	6679
550	6798	6904	7055
600	7324	7257	7424
650	7545	7555	7779
700	7918	7919	8137
750	8178	8165	8430
800	8510	8485	8796
850	8857	8791	8987
900	9191	9176	9441
950	9304	9435	9700
1000	9789	9851	10028

Table 3: Edgecut of partionings of the road network of Belgium.

Germany			
k	HEB-cut heuristic edgecut		unweighted edgecut
	12 sample vertices	115 sample vertices	
2	144	137	143
4	351	351	381
8	702	680	763
16	1205	1136	1264
32	1852	2016	2060
64	2999	3017	3254
128	4722	4688	4850
256	7144	7132	7347
512	10896	10879	11486
1024	16755	17029	17715
50	2578	2576	2687
100	3987	4057	4222
150	5114	5140	5370
200	6162	6158	6377
250	6937	7037	7300
300	7814	7830	8274
350	8538	8579	9023
400	9292	9341	9624
450	9936	9991	10436
500	10737	10739	11187
550	11416	11559	11833
600	11927	12203	12400
650	12778	12650	13212
700	13292	13468	13697
750	13865	13883	14334
800	14478	14438	15064
850	14912	15071	15442
900	15595	15650	16152
950	16134	16038	16504
1000	16762	16734	17045

Table 4: Edgecut of partionings of the road network of Germany.

Great Britain			
k	HEB-cut heuristic edgecut		unweighted edgecut
	77 sample vertices	773 sample vertices	
2	105	122	121
4	280	284	293
8	521	516	578
16	893	893	914
32	1465	1490	1587
64	2383	2346	2565
128	3638	3712	3976
256	5766	5847	6155
512	9036	9158	9487
1024	14218	14364	15000
50	1959	2019	2120
100	3245	3158	3264
150	4070	4134	4268
200	4882	4921	5251
250	5679	5729	5985
300	6427	6420	6691
350	7046	7111	7402
400	7662	7890	8169
450	8315	8489	8730
500	8991	9073	9366
550	9499	9567	10024
600	9945	10084	10511
650	10531	10537	11038
700	11104	11150	11601
750	11654	11746	12244
800	12118	12061	12677
850	12519	12666	13147
900	12975	13114	13580
950	13527	13564	14261
1000	13971	13982	14703

Table 5: Edgecut of partionings of the road network of Great Britain.

Italy			
k	HEB-cut heuristic edgecut		unweighted edgecut
	67 sample vertices	669 sample vertices	
2	58	58	44
4	160	147	143
8	273	260	304
16	474	472	544
32	874	889	876
64	1471	1458	1534
128	2370	2387	2532
256	3890	3886	3982
512	6290	6270	6374
1024	9877	9933	10211
50	1221	1208	1262
100	2015	2024	2179
150	2713	2693	2817
200	3228	3265	3397
250	3763	3725	3904
300	4263	4347	4402
350	4821	4753	5019
400	5270	5342	5439
450	5717	5768	5799
500	6083	6055	6253
550	6497	6571	6738
600	6939	6999	7071
650	7349	7282	7452
700	7788	7638	7887
750	7991	8074	8339
800	8315	8432	8608
850	8727	8739	8981
900	9057	9149	9322
950	9329	9415	9596
1000	9716	9737	9909

Table 6: Edgecut of partionings of the road network of Italy.

Luxembourg			
k	HEB-cut heuristic edgecut		unweighted edgecut
	115 sample vertices	11460 sample vertices	
2	18	17	18
4	57	55	58
8	91	102	98
16	173	173	170
32	283	278	279
64	429	430	453
128	671	659	687
256	1059	1050	1057
512	1721	1710	1684
1024	2940	2785	2789
50	377	363	375
100	573	582	585
150	733	720	748
200	884	895	898
250	1045	1012	1039
300	1164	1173	1155
350	1278	1325	1303
400	1445	1406	1429
450	1481	1558	1550
500	1675	1647	1668
550	1746	1812	1787
600	1904	1917	1926
650	2021	2042	2056
700	2087	2146	2092
750	2258	2226	2270
800	2385	2360	2386
850	2449	2495	2406
900	2547	2582	2554
950	2618	2659	2652
1000	2795	2874	2789

Table 7: Edgecut of partionings of the road network of Luxembourg.

the Netherlands				
k	HEB-cut heuristic edgecut		unweighted edgecut	Buffoon edgecut
	222 sample vertices	2217 sample vertices		
2	49	62	63	44
4	123	133	154	99
8	258	320	310	218
16	518	516	570	420
32	888	919	908	700
64	1529	1493	1576	1278
128	2526	2500	2623	2148
256	4222	4268	4435	3555
512	6799	6812	6978	5727
1024	10511	10639	10811	9108
50	1234	1289	1321	1127
100	2059	2075	2186	1850
150	2794	2858	2960	2445
200	3412	3518	3620	2927
250	4205	4152	4286	3489
300	4842	4669	4989	4026
350	5232	5260	5390	4349
400	5788	5763	5946	4777
450	6196	6150	6430	5301
500	6663	6817	6916	5555
550	6988	7072	7231	6010
600	7504	7396	7627	6462
650	7873	7947	8033	6639
700	8259	8257	8473	7166
750	8703	8584	8862	7499
800	8920	9004	9178	7688
850	9398	9369	9598	8149
900	9637	9655	9859	8431
950	10143	10118	10304	8854
1000	10468	10428	10605	9207

Table 8: Edgecut of partionings of the road network of the Netherlands.