arXiv:1111.6745v1 [cs.CC] 29 Nov 2011

Fast Balanced Partitioning of Grid Graphs is Hard*

Andreas Emil Feldmann[1]

[1]Institute of Theoretical Computer Science, ETH Zürich, Switzerland
feldmann@inf.ethz.ch

12th June 2022

Abstract

We consider the k-BALANCED PARTITIONING problem in which the n vertices of a graph are to be partitioned into k sets of size at most $\lceil n/k \rceil$ while minimising the *cut size*, which is the number of edges cut. Due to our application we are interested in *fast* algorithms that will solve the problem on *solid grid graphs*. A *grid graph* is a finite subgraph of the infinite 2D grid. We call it *solid* if it is connected and has no holes. We are able to harness some recent results on the structure of cuts in solid grid graphs in order to improve the runtime of an algorithm by Simon and Teng [13] for these graphs. This algorithm will compute a solution in which the set sizes of the partition are at most $2\lceil n/k \rceil$. The cut size deviates by at most a factor of $\mathcal{O}(\log k)$ from an optimal solution to k-BALANCED PARTITIONING.

The main contribution of our work however is a proof that shows that, unless P=NP, no FPTAS exists that will compute partitions in which the set sizes are at most $(1+\varepsilon)\lceil n/k \rceil$ for any $\varepsilon > 0$. This is true even if the cut size may deviate by a factor of n^c/ε^d from optimum for any constants c and d where, for solid grid graphs, $c < 1/2$. These are the first bicriteria inapproximability results for this problem. We present a framework yielding such reductions if certain structural properties can be shown to exist for the considered graphs. The reduction technique can also be used for other graph classes and is therefore of independent interest.

1 The Model and Setting

We consider the k-BALANCED PARTITIONING problem in which a graph is to be cut into equally sized pieces while minimising the number of cut edges. More formally, a partition of the vertex set V of size n into k sets is asked for such that each set contains at most $\lceil n/k \rceil$ vertices. At the same time the *cut size*, which is the number of edges connecting vertices from different sets in the partition, is to be minimised. This problem has many applications ranging from branch-and-bound algorithms to VLSI design. In our case the motivation to

*The author is supported by the Swiss National Science Foundation under grant 200021_125201/1.

study the problem stems from parallel computations for finite element models. In these a continuous domain of a physical model is discretised into a mesh of discrete sub-domains (the elements). Each vertex in the resulting graph then corresponds to a computational task in the physical simulation at hand. An edge means that the incident tasks need to exchange data during the simulation. Since the model is usually very large, the computation is done in parallel. Hence the tasks are to be scheduled (i.e. partitioned) on to a number of machines (given by k) so that the loads of the machines (the set sizes) are balanced. At the same time the interprocessor communication (the cut size) is to be minimised since this constitutes a bottleneck in parallel computations. In this paper we focus on the case where the domain is embedded in the plane, i.e. 2D finite element models. For these the corresponding graph is a planar graph, often given by a tessellation of the plane. We therefore give special care to so called *solid grid graphs* which correspond to tessellations of the plane into squares: a *grid graph* is a finite subgraph of the infinite two-dimensional grid. An interior face of a grid graph with more than four edges surrounding it is called a *hole*. If a grid graph does not have any holes and is connected it is called *solid*.

We are interested in finding *fast* algorithms for this setting. That is, we are looking for algorithms that will solve the problem within a polynomial runtime of low degree. In general the k-`BALANCED PARTITIONING` problem is NP-hard [6] and it is even NP-hard to approximate the cut size within any finite factor [1]. However this hardness result relies on the fact that a general graph may not be connected and thus the optimal cut size can be zero. Since a 2D finite element model always induces a connected planar graph this strong hardness result may not apply. However in [4] it was shown that even for trees a similar hardness is achieved: already for this simple graph class it is NP-hard to approximate the cut size of an optimal solution to k-`BALANCED PARTITIONING` within n^c for any constant $c < 1$. The latter result however relies on the fact that the maximum degree of the tree is unbounded. Typically though, in a tessellation of the plane the maximum degree of the graph is bounded by a small constant, as for instance in grid graphs. In fact the results in [4] show that even though approximating the cut size in constant degree trees is APX-hard, there exists an approximation algorithm that computes a solution that is within $\mathcal{O}(\log(n/k))$ of optimum for such trees. This again raises the question of whether efficient approximation algorithms can be found for planar graphs that correspond to tessellations of the plane. In this paper we give a negative answer to this question by showing that it is NP-hard to approximate the cut size within n^c for any constant $c < 1/2$ for solid grid graphs. On the positive side we can show that it is possible to match this hardness by giving a fast algorithm that computes a $\mathcal{O}(\sqrt{n})$ approximation in $\mathcal{O}(kn)$ time.

The approximation factor given by the above algorithm is not satisfying for practical purposes. Therefore, instead of computing *perfectly balanced* partitions in which each set has size at most $\lceil n/k \rceil$, we allow the set sizes to deviate from this strict constraint. More formally we will consider bicriteria approximations that allow solutions in which the set sizes are at most $(1 + \varepsilon)\lceil n/k \rceil$ for some $\varepsilon \geq 0$. At the same time we will compare the computed cut size with the optimal cut size of a perfectly balanced solution, as is the rule in bicriteria approximation settings. Throughout this paper we will denote the approximation ratio on the cut size by α. In practical applications the value of ε should not be too large. In particular we desire an algorithm that computes a *near-balanced* partition in

which ε can be chosen to be any value greater than zero. In [4] an algorithm is presented that for any graph will compute a near-balanced solution such that the cut size is within $\alpha \in \mathcal{O}(\log n)$ of an optimal perfectly balanced partition. However this algorithm is a PTAS for which the exponent of the runtime depends on the inverse of ε. Therefore this algorithm is not viable for practical purposes. It is hence a natural question to ask whether a fast algorithm exists that will compute a near-balanced solution with a reasonable ratio α. Note that the factor α of the algorithm from [4] does not depend on ε. It therefore suggests itself to try and find an algorithm that will compensate the cost of being able to compute a near-balanced solution for any $\varepsilon > 0$ not in the runtime but in the cut size. In this paper however we show that there is no hope for a reasonable algorithm of this sort. More precisely we show that for solid grid graphs there is no FPTAS, i.e. no algorithm for which the runtime is a polynomial in the input length and the inverse of ε, with the following properties, unless P=NP. The solution computed by the supposed FPTAS should be near-balanced and the cut size may deviate from the optimal cut size of a perfectly balanced partition by a factor of $\alpha = n^c/\varepsilon^d$, for any constants c and d where $c < 1/2$.

1.1 Our Contributions

Our main contribution is a framework with which hardness results as the ones mentioned above can be achieved. We will identify some sufficient conditions on the considered graphs that will allow for the reductions. By proving that solid grid graphs exist that fulfil these structural properties we will be able to yield the hardness results. In particular we derive a proof that shows that approximating the cut size within a factor of n^c is NP-hard for any constant $c < 1/2$ when perfectly balanced partitions are desired. Additionally we also show that the properties allow the conclusion that no FPTAS computing near-balanced solutions and approximating the cut size within a factor n^c/ε^d, for arbitrary constants c and d where $c < 1/2$, exists for solid grid graphs, unless P=NP. We are also able to use the proposed reduction framework for other graph classes as for instance general graphs and trees, in order to give stronger hardness results as the ones given in the literature. In particular for general graphs we can show that, unless P=NP, no FPTAS exists that will compute a near-balanced solution and approximate the cut size within any finite factor. For trees we can prove that no such algorithm exists that guarantees a ratio of $\alpha = n^c/\varepsilon^d$, for arbitrary constants c and d where $c < 1$. To the best of our knowledge these are the first hardness results for bicriteria approximations of this problem.

On the positive side we also give two fast algorithms that approximate the problem on grid graphs. The first algorithm runs in $\mathcal{O}(kn)$ time and computes a perfectly balanced solution in which the cut size is approximated within $\mathcal{O}(\sqrt{n})$. The existence of this algorithm shows that both of the above hardness results are asymptotically tight since it constitutes a (trivial) FPTAS matching the approximation ratios. However the approximation guarantee of this algorithm is not satisfactory. We therefore additionally present a fast algorithm for solid grid graphs that computes a partition in which the balance of the set sizes deviates by at most a factor of two (i.e. $\varepsilon = 1$) from being perfectly balanced. In return $\alpha \in \mathcal{O}(\log k)$ for this algorithm. The existence of such an algorithm was already shown by Simon and Teng [13]. However we are able to harness

structural properties on cuts in solid grids developed in [3] in order to improve the runtime for these graphs. More specifically we can show that the algorithm runs in time $\mathcal{O}(n^{1.5} \log k)$ for solid grid graphs.

1.2 Related Work

For general graphs it was already shown by Garey and Johnson [6] that the k-BALANCED PARTITIONING problem is NP-hard. Andreev and Räcke [1] showed that it is even NP-hard to approximate the cut size within any finite factor α. This result shows that for general graphs it is necessary to settle for bicriteria approximations which also approximate the balance of the set sizes. Accordingly in [1] an algorithm was presented that computes a near-balanced solution for any constant $\varepsilon > 0$ in polynomial time and for which $\alpha \in \mathcal{O}(\log^{1.5}(n)/\varepsilon^2)$. This result was improved by Feldmann and Foschini [4] by showing that there is a corresponding algorithm for which $\alpha \in \mathcal{O}(\log n)$. If the balance of the set sizes is allowed to deviate by a factor of two from being perfectly balanced, Krauthgamer *et al.* [8] gave an algorithm guaranteeing a ratio of $\alpha \in \mathcal{O}(\sqrt{\log n \log k})$.

For trees it was shown in [4] that computing perfectly balanced solutions and approximating the cut size within $\alpha = n^c$, for any constant $c < 1$, is NP-hard. If the degree of the trees is restricted to 5 it was also shown that the k-BALANCED PARTITIONING problem is NP-hard, while for trees with maximum degree 7 it is APX-hard. In the same paper a PTAS is given that for any tree will compute a near-balanced solution for which $\alpha = 1$, i.e. the cut size is at most the cut size of an optimal perfectly balanced solution. To the best of our knowledge the only additional result for special graph classes is for graphs with excluded minors (such as planar graphs). For these the results in [7] can be applied together with a spreading metric relaxation in order to compute near-balanced solutions where $\alpha \in \mathcal{O}(1)$.

However Simon and Teng [13] gave a framework with which approximate solutions to k-BALANCED PARTITIONING can be computed for various graph classes. It is a recursive procedure that will repeatedly use a given approximation algorithm for *sparsest cuts*.[1] In contrast to the algorithms above the scheme in [13] is fast on certain types of graphs. The set sizes of the computed partition will deviate by at most a factor of two from being perfectly balanced. For general graphs the method can achieve a factor of $\alpha \in \mathcal{O}(\log n \log k)$ using results by Leighton and Rao [9] on approximate sparsest cuts. However the runtime will be unsatisfactory. On the other hand for planar graphs it is possible to achieve a ratio of $\alpha \in \mathcal{O}(t \log k)$ in $\mathcal{O}(n^{1.5+1/t}\sqrt{\Delta}\log^3 n \log k)$ time, where Δ is the maximum degree of the graph and t can be chosen arbitrarily. For this we use the results by Park and Phillips [11] on sparsest cuts in planar graphs. This is also the best result known for grid graphs (where Δ can be ignored since it is constant).

2 Two Fast Algorithms for Grid Graphs

In this section we present two algorithms that run in fully polynomial time and approximate the k-BALANCED PARTITIONING problem on connected grid graphs. The first one computes a perfectly balanced solution and approximates the cut

[1] We will give a more detailed account of how the algorithm works in the following section.

size within $\mathcal{O}(\sqrt{n})$. The second algorithm has a better approximation ratio on the cut size, namely $\mathcal{O}(\log k)$, but at the price of also approximating the balance of the solution: the computed sets can deviate by a factor of two from being perfectly balanced. In the next section we show that there is little hope for fully polynomial algorithms giving better approximation guarantees.

We first show that there is a fast algorithm that computes a perfectly balanced solution for which $\alpha \in \mathcal{O}(\sqrt{n})$. The algorithm uses a greedy strategy by recursively cutting along *separators*, which are cuts that partition the vertex set of a graph into two linear sized sets. More formally a *cut* $W \subset V$ is a non-empty subset of the vertices. Its *cut size* denoted by $C(W)$ is the number of edges connecting the vertices in W with those in $V \setminus W$. Given a constant $b \leq 1/2$, an edge separator is a cut W such that both W and $V \setminus W$ contain at least bn vertices. An optimal separator is one that minimises the cut size. Using this tool it is possible to cut out any desired number of vertices from a grid graph in linear time using a limited number of edges, as the next lemma shows.

Lemma 1. *For any $m \in \{0, \ldots, n\}$, given a grid graph with n vertices there is an algorithm that computes a cut W such that $|W| = m$ and $C(W) \in \mathcal{O}(\sqrt{n})$ in $\mathcal{O}(n)$ time.*

Proof. Diks et al. [2] show that for any planar graph one can find a separator with cut size $\mathcal{O}(\sqrt{\Delta n})$ in linear time for which $b = 1/3$, where Δ is the maximum degree of the graph. That is, for any grid graph (constituting a planar graph with $\Delta \leq 4$) with a total of n vertices we can efficiently find a cut with cut size $\mathcal{O}(\sqrt{n})$ that partitions the vertices into two sets of size at least $n/3$ each.

The approximation algorithm will attempt to cut out the m vertices from the given input grid in a recursive manner. In each recursion step i it will try to cut away a desired number m_i of vertices of the current grid graph using a separator, starting with the given input graph and $m_0 = m$. If at any particular recursion step i the number n_i of vertices in the current grid graph equals the desired number m_i, the recursion stops. Otherwise a separator containing between $n_i/3$ and $2n_i/3$ of the n_i vertices with cut size $\mathcal{O}(\sqrt{n_i})$ is computed. The algorithm will then attempt to cut out the desired set of vertices in the larger of the two vertex sets. If $m_i \leq n_i/2$ the algorithm will set $m_{i+1} = m_i$. Otherwise it will declare the current n_i vertices as the desired ones and set $m_{i+1} = n_i - m_i$ in order to correct for the additional vertices cut out in this recursion step. In both cases $m_{i+1} \leq n_i/2$ and hence the additional vertices needed can be found in the graph induced by the larger of the two vertex sets defined by the separator.

Clearly this recursive procedure will cut out m vertices from the given grid graph. Note that in each recursion step $n_i \leq (2/3)^i n$. From this we can both conclude that the number of edges cut in total can be bounded by $\mathcal{O}(\sqrt{n})$ and that the runtime is $\mathcal{O}(n)$. This can easily be seen since in each recursion step i the number of edges cut is $\mathcal{O}(\sqrt{n_i})$ and the runtime is $\mathcal{O}(n_i)$. □

The above greedy scheme to cut out any desired number of vertices can be put to work in an algorithm to compute a perfectly balanced solution as follows.

Theorem 2. *There is an algorithm for the k-`BALANCED PARTITIONING` problem on connected grid graphs that computes a perfectly balanced solution and achieves a ratio of $\alpha \in \mathcal{O}(\sqrt{n})$ on the cut size. The runtime of the algorithm is $\mathcal{O}(kn)$.*

Proof. The algorithm will repeatedly use the procedure given by Lemma 1 in order to cut out $\lceil n/k \rceil$ vertices from the given grid graph. Each resulting set of cut out vertices is made a set of the computed solution. These vertices are then removed from the grid before repeating the procedure. If at some point the resulting grid graph has at most $\lceil n/k \rceil$ vertices, all of them are assigned to a new set of the solution. If there are less than k sets resulting from this algorithm then empty sets are added to the solution in order to make it feasible.

Clearly the result of this algorithm is a perfectly balanced solution and the runtime is $\mathcal{O}(kn)$. Any perfectly balanced solution will contain at least $n/\lceil n/k \rceil \geq k/2$ non-empty sets. Since the input graph is connected such a solution will cut at least $k/2-1$ edges. This gives a lower bound on the optimum which can be compared with the $\mathcal{O}(k\sqrt{n})$ edges cut by the algorithm. Hence the approximation ratio on the cut size follows. □

Better approximation ratios on the cut size than the one presented in Theorem 2 can be achieved if the set sizes are allowed to deviate from being perfectly balanced. In the following we allow the balance to be off by a factor of two from a perfectly balanced solution. This allows us to find a fast algorithm for k-`BALANCED PARTITIONING` that approximates the cut size by a factor of $\mathcal{O}(\log k)$. For solid grid graphs the given algorithm improves on the runtime of the known algorithms by harnessing recent results on the structure of cuts in solid grids. To achieve this we use an algorithm by Simon and Teng [13] that can be applied to any graph. In order to determine the runtime we will briefly describe the workings of their algorithm.

Similar to the algorithm above, the algorithm given in [13] is based on recursively using a separator. However instead of using a separator for which an upper bound on the absolute cut size can be given, for this algorithm the cut size must be relative to the optimum in a given graph. For solid grid graphs the optimal separator can be computed in polynomial time using the algorithm in [5] which however is much too slow for our purposes. Instead the algorithm works by using an approximate separator and repeating recursively on the two resulting subgraphs. This is done until the cut out graphs are of size at most $2\lceil n/k \rceil$. In a second phase the algorithm will always merge the two smallest vertex sets resulting from the cutting phase until only k sets remain. The following theorem summarises the qualities of this algorithm.

Theorem 3 ([13])**.** *Let an algorithm be given that computes an approximate separator for which $b \leq 1/3$ and the cut size is within a factor β of optimum. There is an algorithm that recursively uses the former and computes a partition of the vertices into k sets such that each set has size at most $2\lceil n/k \rceil$. Moreover the number of edges cut is at most $\mathcal{O}(\beta \log k)$ times the number of edges cut by an optimal perfectly balanced solution to the k-`BALANCED PARTITIONING` problem.*

In the following we will show how to achieve a constant approximation algorithm for separators in solid grid graphs that runs in linear time. Leighton and Rao [9] showed how approximate separators can be computed using *sparsest cuts*. The *sparsity* of a cut $W \subset V$ is defined as

$$\frac{C(W)}{|W| \cdot |V \setminus W|} \text{ ,}$$

and the sparsest cut of a graph is a cut that minimises the sparsity. The algorithm will cut out an approximate sparsest cut and keep the smaller of the two resulting vertex sets to be added to the separator. This is continued recursively in the larger piece until at least bn vertices are collected for the separator. The following theorem is due to Leighton and Rao.

Theorem 4 ([9])**.** *Let an algorithm that computes a cut with a sparsity that is within a factor β of optimum, and two constants $b' \leq 1/2$ and $b'' \leq 1/3$ be given for which $b'' < b'$. There is an algorithm that recursively uses the latter to compute a separator in which both resulting vertex sets have size at least $b''n$. Moreover the number of edges cut is at most $\mathcal{O}(\beta)$ times the optimum used by a separator for which the vertex sets have size at least $b'n$.*

To use Theorems 3 and 4 we need to show how to compute a sparse cut for solid grid graphs in order to compute an approximate solution on these graphs. For this we need some observations on cuts in planar graphs. It is well-known that any cut in a planar graph G induces a set of simple cycles in the *dual graph*, i.e. the (multi-)graph which has the faces of G as its vertices and an edge between any two neighbouring faces. Given an embedding of the dual of a planar graph (which is itself planar), any simple cycle cuts the plane into two connected regions: the interior and the exterior. We say that a pair of simple cycles *cross* if one of them both contains vertices lying within the exterior and the interior region cut out by the other cycle. If they do not cross they are called *non-crossing*. Note that any pair of crossing cycles can be seen as a (different) pair of non-crossing cycles. Hence any cut in a planar graph induces a set of non-crossing simple cycles in the dual graph. We call any set of edges in G that corresponds to a simple cycle in the dual a *segment*. Accordingly any cut induces a set of non-crossing segments, i.e. a set of segments for which the corresponding cycles do not cross.

Using the results from [3] we can show that in solid grid graphs it suffices to restrict our attention to a small subset $\mathcal{K}$ of all possible segments in order to find a good approximation of a sparsest cut. In this family $\mathcal{K}$ only segments that have a simple shape with at most one right angled bend are included (see [3] for more details). Moreover the segments in the family are all connected to the boundary of the solid grid, i.e. they do not cut out a vertex set from the inside of the grid thereby producing a hole. The following theorem was proved in [3].

Theorem 5 ([3])**.** *Let C^* be the minimum cut size over all cuts containing m vertices of a solid grid graph G, and let $\varepsilon \in]0, 1]$. There exists a set of non-crossing segments from a family $\mathcal{K}$ that induces a cut containing $m' \in [(1-\varepsilon)m, (1+\varepsilon)m]$ vertices with a cut size of $\mathcal{O}(1/\sqrt{\varepsilon}) \cdot C^*$. Moreover $|\mathcal{K}| \in \mathcal{O}(n)$ and any segment from $\mathcal{K}$ corresponds to a simple cycle in the dual graph of G containing the exterior face.*

This theorem entails that a constant approximation to a sparsest cut can be found using segments from the family $\mathcal{K}$, as we show next.

Corollary 6. *For any grid graph G for which the connected components are solid, there is a set of non-crossing segments from the family $\mathcal{K}$ inducing a cut W with a sparsity which approximates the sparsest cut W^* of G by a constant factor.*

Proof. If G is disconnected then clearly any sparsest cut has sparsity 0 and contains all vertices of some subset of the connected components. This cut is trivially induced by the empty set of segments from $\mathcal{K}$. Otherwise the sparsest cut W^* has minimum cut size among all cuts of size $|W^*|$ in G. Hence Theorem 5 in particular means that for W^* there is a set of non-crossing segments from $\mathcal{K}$ inducing a cut W which contains between $(1-\varepsilon)|W^*|$ and $(1+\varepsilon)|W^*|$ vertices and $C(W) \in \mathcal{O}(1/\sqrt{\varepsilon}) \cdot C(W^*)$. To compare the sparsities of W and W^* we need to get a better understanding of the behaviour of the sparsity $C(W)/(|W|(n-|W|))$ of W. The denominator of this term is a function in the size of W and reaches a maximum at $n/2$ in the interval $[0, n]$. For smaller values than $n/2$ the function is increasing while for larger values it is decreasing. Hence when considering the interval $[(1-\varepsilon)|W^*|, (1+\varepsilon)|W^*|]$ the smallest value of the denominator lies at one of the extreme points $(1-\varepsilon)|W^*|$ and $(1+\varepsilon)|W^*|$. It can easily be verified that the smallest value is reached at $(1-\varepsilon)|W^*|$ if and only if $|W^*| \leq n/2$. Since we can assume that the latter is true w.l.o.g., we can lower bound the denominator of the sparsity of W by

$$(1-\varepsilon)|W^*| \cdot (n - (1-\varepsilon)|W^*|) \geq (1-\varepsilon)|W^*|(n - |W^*|).$$

For the sparsity itself this means that

$$\frac{C(W)}{|W|(n-|W|)} \in \mathcal{O}\left(\frac{1}{\sqrt{\varepsilon}(1-\varepsilon)}\right) \cdot \frac{C(W^*)}{|W^*|(n-|W^*|)}.$$

According to Theorem 5 we can choose ε freely between 0 and 1. Hence we can set $\varepsilon = 1/3$ which is the value at which the above term depending on ε is minimal. This means that the sparsity of W is at most a constant factor worse than the sparsity of W^*. □

The above theorem helps in finding a good approximation to a sparsest cut in linear time since the set of segments from $\mathcal{K}$ inducing the lowest sparsity turns out to be a set with only a single segment, as we will show next. As shown in [5] enumerating and computing the size of the induced cuts can be done in linear time since the size of $\mathcal{K}$ is $\mathcal{O}(n)$. A similar observation as the one given in the following lemma was also shown in [12, 13] in the context of *min ratio cuts* for planar graphs.

Lemma 7. *Let a family of segments $\mathcal{K}$ for a solid grid graph be given. Consider the cuts induced by non-crossing sets of segments from $\mathcal{K}$ and having lowest sparsity among these. There is one of these cuts which is induced by only a single segment.*

Proof. Consider a cut W among those induced by non-crossing sets of segments from $\mathcal{K}$, and the connected components induced by W and $V \setminus W$. If there is more than one segment in the set $S \subseteq \mathcal{K}$ inducing W then there must be more than one connected component induced by either W or $V \setminus W$. Assume w.l.o.g. that W consists of several connected components and let X be the vertex set of one of them. This means that $|W| = |X| + |W \setminus X|$ and $C(W) = C(X) + C(W \setminus X)$. Since $|W|(n-|W|) = n|W| - |W|^2$ the sparsity of W can thus be lower bounded by

$$\frac{C(X) + C(W \setminus X)}{n(|X| + |W \setminus X|) - (|X| + |W \setminus X|)^2} \geq \frac{C(X) + C(W \setminus X)}{n|X| - |X|^2 + n|W \setminus X| - |W \setminus X|^2}.$$

This means that either the sparsity of X or the sparsity of $W \setminus X$ is at most the sparsity of W. Since the segments in S are non-crossing, both X and $W \setminus X$ are induced by subsets of non-crossing segments from S. By repeating this argument for a sparsest cut among those induced by non-crossing sets of segments from $\mathcal{K}$, we can always find one that is induced by only a single segment from $\mathcal{K}$. □

The above observations can now be put together in order to show that an approximate solution to k-BALANCED PARTITIONING on solid grid graphs can be computed fast if the set sizes are allowed to deviate by a factor of two from being perfectly balanced.

Theorem 8. *There is an algorithm that will compute a partition of the n vertices of a solid grid graph into k sets of size at most $2\lceil n/k \rceil$ each, and for which $\alpha \in \mathcal{O}(\log k)$. Moreover the algorithm runs in time $\mathcal{O}(n^{1.5} \log k)$.*

Proof. By Lemma 7, for any solid grid graph we can compute a cut induced by non-crossing segments from the family $\mathcal{K}$ with lowest sparsity among these, by considering all segments in $\mathcal{K}$ and picking the one with smallest sparsity. For a disconnected grid graph for which the connected components are solid a sparsest cut can be computed by taking an arbitrary connected component and putting its vertices in the cut. Corollary 6 says that in both cases the computed cut is a constant approximation of the sparsest cut in the given grid.

The algorithm by Leighton and Rao [9] can use these approximate sparsest cuts in order to compute a constant approximation to an optimal edge separator for any grid graph for which the connected components are solid. This is true since by Theorem 5 the segments in $\mathcal{K}$ do not cut out vertex sets inside of the given grid, i.e. the segments are all connected to the boundary of the grid, since the corresponding cycles in the dual graph include the exterior face. This means that no recursion step of the algorithm can introduce a hole into the resulting grids induced by the two vertex sets defined by the separator. Hence Corollary 6 can be applied in every one of the recursion steps. We use the approximate sparsest cut to compute a separator using Theorem 4, where we will set $b' = 1/3$ and $b'' = 1/6$. The result is a constant approximation of an optimal separator for any grid graph in which the connected components are solid.

The algorithm by Simon and Teng [13] can then use this approximate separator to compute the sought after approximate solution to k-BALANCED PARTITIONING on solid grid graphs. This in turn is true since the vertex sets of the separator computed above induce two grid graphs which may be disconnected but for which the connected components do not contain holes. Hence in each recursion step of this algorithm the above procedure can be used and is guaranteed to produce a separator with the needed attributes. As already shown in [13] this can be used in Theorem 3 where we set $b = b''$ to compute a solution having the claimed properties.

It remains to be shown that the runtime is $\mathcal{O}(n^{1.5} \log k)$. The algorithm given by Theorem 4 will first use the sparsest cuts that are given by the vertex sets of the connected components in its recursive procedure. Only when the current graph is connected will it start cutting approximate sparsest cuts using the segments in $\mathcal{K}$. For a disconnected graph the vertex set of a connected component of size m can be found in time $\mathcal{O}(m)$. Hence the time needed by the algorithm in all recursive steps until the given graph is connected is $\mathcal{O}(n)$. We will argue that the rest of the time spent by the algorithm given by Theorem 4 to

cut vertices from a connected solid grid is $\mathcal{O}(n^{1.5})$. By Theorem 5 there are only $\mathcal{O}(n)$ many segments in $\mathcal{K}$ for a given solid grid graph with n vertices. Hence computing a constant approximation to a sparsest cut in a solid grid graph only needs linear time (see also [5] for more details). By Lemma 1 any number of vertices can be cut from a grid graph using at most $\mathcal{O}(\sqrt{n})$ edges. Hence there always exists a separator (for any b) with cut size $\mathcal{O}(\sqrt{n})$. Since the algorithm given by Theorem 4 computes a constant approximation to an optimal separator, the total cut size of the solution is also $\mathcal{O}(\sqrt{n})$. In any of the recursive steps of the algorithm the computed sparsest cut will cut at least one edge if the given graph is connected. Hence the time needed to compute the separator is $\mathcal{O}(n^{1.5})$.

The algorithm given by Theorem 3 uses the latter procedure recursively in order to compute the final approximate solution to k-`BALANCED PARTITIONING`. Consider the recursion tree given by the algorithm and its recursive computation steps. With each computation step there is a node associated with it in the tree. That is in the recursion step associated with node i of the recursion tree the algorithm needs $\mathcal{O}(n_i^{1.5})$ time to compute a separator in the grid graph of size n_i considered in that step. Clearly all grid graphs associated with the vertices of one level of the recursion tree, i.e. those graphs considered at the same recursion depth, are disjoint. Hence we can amortise the runtime of a given level by $\mathcal{O}(n^{1.5})$ using Jensen's inequality. Since the separator will in each recursion step i produce two vertex sets of size at least $n_i/6$ and a separator will only be computed for a graph of size at least $2\lceil n/k \rceil$, the height of the recursion tree is $\mathcal{O}(\log k)$. Hence the runtime of the cutting phase of this algorithm is $\mathcal{O}(n^{1.5} \log k)$. In the second phase of the algorithm the resulting vertex sets are merged until only k remain. This can be done in time $\mathcal{O}(n \log n)$ by first sorting the set sizes and then merging the two smallest ones repeatedly. Hence the claimed total runtime follows. □

3 Computing Near-Balanced Partitions in Fully Polynomial Time

In this section we show that the algorithms given in the previous section are the best one can hope for if the runtime is to be fully polynomial. In particular we prove that the k-`BALANCED PARTITIONING` problem on solid grid graphs is NP-hard to approximate within n^c for any constant $c < 1/2$. We will also show that, unless P=NP, no FPTAS exists that approximates the balance within $1+\varepsilon$, for any $\varepsilon > 0$, as well as the cut size within n^c/ε^d, for any constants c and d where $c < 1/2$. This means that even if the cut size is allowed to deviate farther from the optimum the tighter the ratio on the balance is, no fully polynomial approximation algorithms exist. The first algorithm of the previous section shows that these two results are asymptotically tight since it constitutes a (trivial) FPTAS matching the approximation guarantees. The second algorithm of the previous section shows that if ε is not allowed to be arbitrarily small then better approximations on the cut size are possible in fully polynomial time.

The reductions leading to the two hardness results will be stated in a way such that they can easily be adapted to other graph classes than solid grid graphs. In this way we are able give the first bicriteria inapproximability results for this problem. We thereby strengthen the hardness results for trees and general

graphs given in the literature.

3.1 A Reduction Framework

To derive the hardness results we give a reduction from the 3-PARTITION problem defined below. It is known that 3-PARTITION is strongly NP-hard [6] which means that it remains so even if all integers are polynomially bounded.

Definition 9 (3-PARTITION)**.** Given $3k$ integers $a_1, \ldots, a_{3k}$ and a threshold s such that $s/4 < a_i < s/2$ for each $i \in \{1, \ldots, 3k\}$, and $\sum_{i=1}^{3k} a_i = ks$, find a partition of the integers into k triples such that each triple sums up to exactly s.

We will set up a general framework for a reduction from 3-PARTITION to different graph classes. This will be achieved by identifying some structural properties that a graph constructed from a 3-PARTITION instance has to fulfil, in order to show the hardness of the k-BALANCED PARTITIONING problem. We will state a lemma which asserts that if the constructed graph has these properties then an algorithm which approximates the balance within $1 + \varepsilon$ and the cut size within some α will be able to decide the 3-PARTITION problem. Thereafter we will see that carefully choosing the involved parameters for each of the given graph classes yields the desired reductions. While describing the structural properties we will exemplify them for general graphs which constitute an easily understandable case. For these graphs Andreev and Räcke [1] already showed that it is NP-hard to approximate the cut size within any finite factor. We will show that no FPTAS exists even if the balance is approximated as well.

For any 3-PARTITION instance we construct a set of $3k$ graphs, which we will call *gadgets*, with a number of vertices proportional to the integers a_1 to a_{3k}. In particular, for general graphs each gadget G_i, where $i \in \{1, \ldots 3k\}$, is a connected graph on $(\varepsilon n + 1)a_i$ vertices, where n is the total number of vertices in all gadgets. Clearly we need to assume that ε is sufficiently small in order for such a set of gadgets to exist. This can be achieved by letting $\varepsilon = (2ks)^{-1}$ since the total number of vertices is $n = \sum_{i=1}^{3k}(\varepsilon n+1)a_i = (\varepsilon n+1)ks$. Solving this for n and setting the value of ε accordingly gives $n = 2ks$. At the same time this assures that the gadgets can be constructed in polynomial time since 3-PARTITION is strongly NP-hard. For the same reason an FPTAS for k-BALANCED PARTITIONING will run in polynomial time on this set of gadgets for this value of ε. We will show that such an FPTAS can decide the 3-PARTITION problem for any approximation ratio α on the cut size.

In general we will assume that we can construct $3k$ gadgets for the given graph class such that each gadget has pa_i vertices for some p specific for the graph class. These gadgets will then be connected using some number m of edges. The parameters p and m may depend on the values of the given 3-PARTITION instance. For the case of general graphs $m = 0$, i.e. they are disconnected. In order to show that the given gadgets can be used in a reduction, we will assume that they have the property that an upper bound can be given on the number of vertices that can be cut out using a limited number of edges. More precisely given any colouring of the vertices of all gadgets into k colours, by a *minority vertex* in a gadget G_i we mean a vertex that has the same colour as less than half of G_i's vertices. Any partition of the vertices of all gadgets into k sets induces a colouring of the vertices into k colours. The property we need is that using at most αm edges, for some approximation ratio α, to cut the graph into k sets

produces less than $p - \varepsilon n$ minority vertices in total. For general graphs this is easily established since $\alpha m = 0$ for any finite α and hence no gadget can be cut in this case. The following definition formalises the properties needed for our reductions.

Definition 10. An *(α, ε)-reduction set* contains a graph for every instance of `3-PARTITION`, for some given parameters m, p, ε, and α which may depend on the values of the instances. Given such an instance the corresponding graph constitutes $3k$ gadgets connected through m edges such that a gadget G_i, where $i \in \{1, \ldots, 3k\}$, has pa_i vertices. Additionally if at most αm edges are used to partition the n vertices of such a graph into k sets then in total there are less than $p - \varepsilon n$ minority vertices in the induced colouring.

Obviously the involved parameters have to be set to appropriate values in order for the (α, ε)-reduction set to exist. However the values will vary with the considered graph class and we therefore fix them only later. The following lemma assures that given an (α, ε)-reduction set an algorithm for k-`BALANCED PARTITIONING` approximating the cut size within α and the balance within $1 + \varepsilon$ can decide the `3-PARTITION` problem. For general graphs we have seen above that an $(\alpha, (2ks)^{-1})$-reduction set exists for any finite α. This means that an FPTAS for k-`BALANCED PARTITIONING` approximating the cut size within any α can decide the `3-PARTITION` problem in polynomial time. This however is a contradiction unless P=NP.

Lemma 11. *Let for $\varepsilon \geq 0$ an algorithm be given that computes a partition of the vertices for any graph in an (α, ε)-reduction set into k sets of size at most $(1 + \varepsilon)\lceil n/k \rceil$ each. If the cut size of the computed solution deviates by at most α from the optimal cut size of a perfectly balanced solution, then the algorithm can decide the `3-PARTITION` problem.*

Proof. Let k be the value given by the `3-PARTITION` instance corresponding to a graph G in the (α, ε)-reduction set. Assume that the instance has a solution. Then obviously cutting the m edges connecting the gadgets of G gives a perfectly balanced solution to the k-`BALANCED PARTITIONING` problem. Hence the optimal solution in this case has a cut size of at most m. Accordingly the algorithm which approximates the cut size by a factor of α will cut at most αm edges. We will show that in the other case when the `3-PARTITION` instance does not have a solution, the algorithm will cut more than αm edges. Hence the algorithm can decide the `3-PARTITION` problem and thus the lemma follows.

For the sake of deriving a contradiction assume that the algorithm cuts at most αm edges in case the `3-PARTITION` instance corresponding to the given graph G does not permit a solution. Since G is from an (α, ε)-reduction set, by definition this means that from its n vertices, in total less than $p - \varepsilon n$ are minority vertices in the colouring induced by the computed solution of the algorithm. In particular this means that each gadget G_i, where $i \in \{1, \ldots 3k\}$, of G has a *majority colour*, i.e. a colour that at least half the vertices in G_i share. This is because the size of G_i is pa_i and we can safely assume that $a_i \geq 2$ (otherwise the `3-PARTITION` instance is trivial). The majority colours of the gadgets induce a partition $\mathcal{P}$ of the integers a_i of the `3-PARTITION` instance into k sets. That is, we introduce a set in $\mathcal{P}$ for each colour and put an integer a_i in a set if the majority colour of G_i corresponds the colour of the set.

Since we assume that the 3-PARTITION instance does not admit a solution, if all sets in $\mathcal{P}$ contain exactly three integers there must be some sets for which the contained integers do not sum up to exactly the threshold s. On the other hand the bounds on the integers, assuring that $s/4 < a_i < s/2$ for each $i \in \{1, \ldots, 3k\}$, mean that in case not all sets in $\mathcal{P}$ contain exactly three elements, there must also exist sets for which the contained numbers do not sum up to s. By the pigeonhole principle and the fact that the sum over all a_i equals ks, there must thus be some set T among the k in $\mathcal{P}$ for which the sum of the integers is strictly less than s. Since the involved numbers are integers we can conclude that the sum of the integers in T is in fact at most $s-1$. Therefore the number of vertices in the gadgets corresponding to the integers in T is at most $p(s-1)$. Apart from the vertices in these gadgets, all vertices in G that have the same colour as the majority colour c of these gadgets must be minority vertices. Hence there must be less than $p(s-1) + p - \varepsilon n$ many vertices with colour c. Since $\sum_{i=1}^{3k} a_i = ks$ and thus $ps = n/k$, these are less than $n/k - \varepsilon n$.

At the same time the algorithm computes a solution inducing a colouring in which each colour has at most $(1+\varepsilon)n/k$ vertices, since n is divisible by k. This means we can give a lower bound of $n - (k-1)(1+\varepsilon)n/k$ on the number of vertices of a colour by assuming that all other colours have the maximum number of vertices. Since this lower bound equals $(1+\varepsilon)n/k - \varepsilon n$, for any $\varepsilon \geq 0$ we get a contradiction on the upper bound derived above for colour c. Thus the assumption that the algorithm cuts less than αm edges if the 3-PARTITION instance does not have a solution is wrong. □

3.2 Consequences for Grids and Other Graph Classes

We will now consider some specific graph classes and show their hardness with respect to the k-BALANCED PARTITIONING problem. In particular we will first consider trees and thereafter solid grid graphs. For trees it was already shown in [4] that approximating the cut size is NP-hard within n^c for any constant $c < 1$. We will additionally show that no FPTAS exists that approximates the balance within $1+\varepsilon$ and the cut size within n^c/ε^d for any constants c and d where $c < 1$. This result is asymptotically tight since a trivial FPTAS can cut all edges in the tree and thereby approximate the cut size within a factor of n.

Theorem 12. *There is no FPTAS for the k-BALANCED PARTITIONING problem on trees that for any $\varepsilon > 0$ computes a solution in which each of the k sets has size at most $(1+\varepsilon)\lceil n/k \rceil$ and for which $\alpha = n^c/\varepsilon^d$, for any constants c and d where $c < 1$, unless P=NP.*

Proof. We need to identify an (α, ε)-reduction set for trees. We use the same topology for the gadgets as in the reduction given in [4]. That is, each gadget is a star and these are connected in a path through their centre vertices using $m = 3k - 1$ edges. The number of vertices in the stars is proportional to the integers a_i by the factor $p = 3k\alpha + \varepsilon n$. Using at most αm, i.e. less than $3k\alpha$, edges to cut off vertices from a single star, will cut off less than $3k\alpha$ leaves so that more than half of the star is still connected to the centre vertex. This is because we can safely assume that $a_i \geq 2$ for each $i \in \{1, \ldots, 3k\}$ and thus each star contains at least $6k\alpha$ vertices. Therefore partitioning the vertices of all gadgets into k sets using at most αm edges will in total produce less than $3k\alpha$ minority

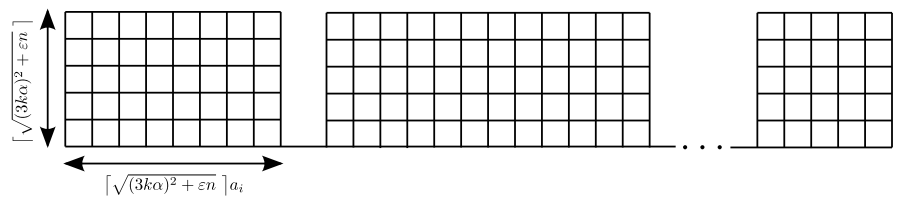


Figure 1: The solid grid constructed for the reduction from `3-PARTITION`. The gadgets which are rectangular grids are connected through the bottom left and right vertices.

vertices in the induced colouring. This establishes the desired upper bound on the number of minority vertices for the (α, ε)-reduction set since $3k\alpha = p - \varepsilon n$.

By Lemma 11 an FPTAS computing a solution on the above (α, ε)-reduction set can decide the `3-PARTITION` problem. In order to prove the claim we need to show that the above (α, ε)-reduction set exists and can be constructed in polynomial time for $\alpha = n^c/\varepsilon^d$. This will be achieved by letting $\varepsilon = (2ks)^{-1}$ in which case the FPTAS also runs in polynomial time since the `3-PARTITION` problem is strongly NP-hard. For these values of α and ε the number of vertices n in a tree of the (α, ε)-reduction set is

$$n = \sum_{i=1}^{3k} pa_i = (3k\alpha + \varepsilon n)ks = 3k^2 s(2ks)^d n^c + n/2.$$

Solving this equation for n gives $n = (6k^2 s(2ks)^d)^{\frac{1}{1-c}}$. Hence if c and d are constants and $c < 1$, any tree in the (α, ε)-reduction set can be constructed in polynomial time. □

For solid grid graphs we establish that it is NP-hard to approximate the cut size within n^c for any constant $c < 1/2$. We also show that no FPTAS exists which for any $\varepsilon > 0$ approximates the balance within $1 + \varepsilon$ and the cut size within n^c/ε^d for any constants c and d where $c < 1/2$. For both of these results we will use the same topology for the grid graphs, namely a set of *rectangular* grid graphs which are connected in a row (Figure 1). By a rectangular grid graph we mean a solid grid graph with the following properties. There exists a planar embedding such that the vertices are coordinates in $\mathbb{N}^2$ and the edges have unit length. Additionally the straight line edges of this embedding touching the exterior face form an orthogonal rectangle or a straight line segment (in the trivial case when there are no edges the grid graph is also rectangular). The *width* of a rectangular grid graph is the number of vertices sharing the same y-coordinate in the defining embedding. Accordingly the *height* is the number sharing the same x-coordinate. We first prove that such topologies can be used as (α, ε)-reduction sets and thereafter choose the parameters according to the considered result.

Lemma 13. *Let $\varepsilon \geq 0$ and $\alpha \geq 1$. For any `3-PARTITION` instance, let a solid grid graph G be given that consists of $3k$ rectangular grids which are connected in a row using their lower left and lower right vertices by $m = 3k - 1$ edges. Moreover let the height and width of a rectangular grid G_i, where $i \in \{1, \ldots 3k\}$, be $\lceil\sqrt{(3k\alpha)^2 + \varepsilon n}\,\rceil$ and $\lceil\sqrt{(3k\alpha)^2 + \varepsilon n}\,\rceil a_i$ respectively. The set of all these graphs form an (α, ε)-reduction set.*

Proof. Consider one of the described graphs for a 3-PARTITION instance. Since both the height and the width of each rectangular grid G_i is greater than αm, using at most αm edges it is not possible to cut across a gadget G_i, neither in horizontal nor in vertical direction. Due to the isoperimetric properties of a rectangular grid this means that with this limited amount of edges the maximum number of vertices can be cut out from the gadgets by using a square shaped cut in one corner of a single gadget. This also follows from Lemma 2 in [10]. Such a cut will contain at most $(\alpha m/2)^2$ vertices. Hence in any colouring induced by a partition of the vertices of the grid graph into k sets using at most αm edges, there are at most $(\alpha m/2)^2$ minority vertices in total. Since the size of each gadget is its height times its width the parameter p is greater than $(\alpha m)^2 + \varepsilon n$. Hence the number of minority vertices is less than $p - \varepsilon n$. □

The above topology is used with varying sizes of the gadgets in order to prove the desired results. We first show that it is NP-hard to approximate the cut size within n^c for any constant $c < 1/2$.

Theorem 14. *There is no polynomial time approximation algorithm for the k-BALANCED PARTITIONING problem on solid grid graphs that approximates the cut size within $\alpha = n^c$ for any constant $c < 1/2$, unless P=NP.*

Proof. We need to show that an (α, ε)-reduction set as proposed in Lemma 13 exists in order to use it together with Lemma 11. Since the balance of the solution is not to be approximated we set $\varepsilon = 0$. The parameter p is determined by the height and width of the gadgets which in this case are $3k\alpha$ and $3k\alpha a_i$ for a gadget G_i respectively. That is $p = (3k\alpha)^2$. By Lemmas 11 and 13 an algorithm that computes a perfectly balanced solution on these grids and approximates the cut size within α can decide the 3-PARTITION problem. Hence to prove the claim for solid grid graphs we need to show that such a grid can be computed in polynomial time whenever $\alpha = n^c$ for a constant $c < 1/2$. The number of vertices of the resulting grid is

$$n = \sum_{i=1}^{3k} pa_i = (3kn^c)^2 ks.$$

Solving this equation for n gives $n = (9k^3 s)^{\frac{1}{1-2c}}$ which is polynomial if c is a constant smaller than $1/2$ since 3-PARTITION is strongly NP-hard. □

To show that no FPTAS exists that approximates the cut size within any reasonable factor we need to take into account that α may depend on ε and $\varepsilon > 0$. The next theorem shows that this is possible.

Theorem 15. *There is no FPTAS for the k-BALANCED PARTITIONING problem on solid grid graphs that for any $\varepsilon > 0$ computes a solution in which each of the k sets has a size at most $(1+\varepsilon)\lceil n/k \rceil$ and where $\alpha = n^c/\varepsilon^d$, for any constants c and d where $c < 1/2$, unless P=NP.*

Proof. As in the proof of Theorem 14 we need to show the existence of the (α, ε)-reduction set given in Lemma 13 in order to apply Lemma 11. We will set $\varepsilon = (8ks)^{-1}$ which means that the runtime of any FPTAS is polynomial since 3-PARTITION is strongly NP-hard. Clearly $\left\lceil \sqrt{(3k\alpha)^2 + \varepsilon n} \right\rceil \leq 2\sqrt{(3k\alpha)^2 + \varepsilon n}$ and hence the parameter p, which is determined by the width and height of the

gadgets, is at most $4(3k\alpha)^2 + 4\varepsilon n$. This means that the number of vertices can be upper bounded by

$$n = \sum_{i=1}^{3k} pa_i \leq (4(3kn^c/\varepsilon^d)^2 + 4\varepsilon n)ks = 36k^3 s(8ks)^{2d} n^{2c} + n/2.$$

Solving this inequality for n yields $n \leq (72k^3 s(8ks)^{2d})^{\frac{1}{1-2c}}$ which is polynomial if c and d are constants and $c < 1/2$ since `3-PARTITION` is strongly NP-hard. □

4 Conclusions

In this paper we were concerned with the question of whether fast algorithms can be found for the k-`BALANCED PARTITIONING` problem on graphs resulting from 2D finite element models. We were able to show that on solid grid graphs perfectly balanced solutions can be computed in $\mathcal{O}(kn)$ time. However the cut size in this solution is approximated by a factor of $\alpha \in \mathcal{O}(\sqrt{n})$. If the set sizes are allowed to deviate by a factor of two from being perfectly balanced, we showed that there is an algorithm for solid grid graphs that achieves a factor of $\alpha \in \mathcal{O}(\log k)$. We harnessed structural properties of cuts in these graphs in order to show that the runtime of this algorithm is $\mathcal{O}(n^{1.5} \log k)$. This improves on the best previously known runtime of this algorithm for solid grid graphs.

Our main contribution however was a framework with which hardness of approximation results can be achieved by identifying some structural properties of the considered graph class. Using this method we were able to show that for solid grid graphs it is NP-hard to approximate the cut size within $\alpha = n^c$, for any constant $c < 1/2$. We also proved that no FPTAS exists that computes a near-balanced solution for these graphs and achieves a factor of $\alpha = n^c/\varepsilon^d$, for any constants c and d where $c < 1/2$, unless P=NP. The algorithm computing perfectly balanced solutions from above shows that these two results are asymptotically tight. We were also able to use our framework on other graph classes like general graphs and trees. To the best of our knowledge these gave the first results on the hardness of bicriteria approximation for the k-`BALANCED PARTITIONING` problem. It is worth noticing that the results for solid grids can easily be applied to other grid graphs that have a simple structure but are not solid. For instance one might consider grid graphs that have rectangular shapes having only rectangular shaped holes. By adding an edge between the top most corner vertices in the gadgets used in Figure 1 and multiplying the respective parameters in the reductions by two, one can show that this class of grid graphs experiences the same complexity.

In face of these harsh complexity results that rule out reasonable algorithms even for very simple graph classes, the question remains how provably good algorithms can be found. Possibly, randomised algorithms might circumvent the presented complexity results. Or some more sophisticated tools analysing the complexity space of the problem, such as given by smoothed analysis techniques, may reveal that the hard instances are isolated and therefore not relevant in practice.

References

[1] K. Andreev and H. Räcke. Balanced graph partitioning. *Theor. Comp. Sys.*, 39(6):929–939, 2006.

[2] K. Diks, H. N. Djidjev, O. Sykora, and I. Vrto. Edge separators of planar and outerplanar graphs with applications. *Journal of Algorithms*, 14(2):258 – 279, 1993.

[3] A. E. Feldmann, S. Das, and P. Widmayer. Restricted cuts for bisections in solid grids: A proof via polygons. In *Proceedings of the 37th International Workshop on Graph-Theoretic Concepts in Computer Science*, 2011.

[4] A. E. Feldmann and L. Foschini. Balanced partitions of trees and applications. Technical Report 733, Institute of Theoretical Computer Science, ETH Zürich, July 2011.

[5] A. E. Feldmann and P. Widmayer. An $O(n^4)$ time algorithm to compute the bisection width of solid grid graphs. In *Proceedings of the 19th Annual European Symposium on Algorithms*, 2011.

[6] M. R. Garey and D. S. Johnson. *Computers and Intractability: A guide to the Theory of NP-completeness*. W.H. Freeman and Co., San Fransisco, California, 1979.

[7] P. Klein, S. Plotkin, and S. Rao. Excluded minors, network decomposition, and multicommodity flow. In *Proceedings of the 25th annual ACM symposium on Theory of computing*, pages 682–690. ACM, 1993.

[8] R. Krauthgamer, J. Naor, and R. Schwartz. Partitioning graphs into balanced components. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 942–949. Society for Industrial and Applied Mathematics, 2009.

[9] T. Leighton and S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM*, 46(6):787–832, 1999.

[10] C. Papadimitriou and M. Sideri. The bisection width of grid graphs. *Theory of Computing Systems*, 29:97–110, 1996.

[11] J. K. Park and C. A. Phillips. Finding minimum-quotient cuts in planar graphs. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, STOC '93, pages 766–775, New York, NY, USA, 1993. ACM.

[12] S. Rao. Faster algorithms for finding small edge cuts in planar graphs. In *Proceedings of the 24th annual ACM symposium on Theory of computing*, pages 229–240, 1992.

[13] H. D. Simon and S. H. Teng. How good is recursive bisection? *SIAM J. Sci. Comput.*, 18(5):1436–1445, 1997.