arXiv:1111.6745v5 [cs.CC] 29 Jan 2013

Fast Balanced Partitioning is Hard Even on Grids and Trees*

Andreas Emil Feldmann

Combinatorics and Optimization Department,
University of Waterloo, Canada
andreas.feldmann@uwaterloo.ca

Abstract

Two kinds of approximation algorithms exist for the k-BALANCED PARTITIONING problem: those that are fast but compute unsatisfactory approximation ratios, and those that guarantee high quality ratios but are slow. In this article we prove that this tradeoff between *running time* and *solution quality* is unavoidable. For the problem a minimum number of edges in a graph need to be found that, when cut, partition the vertices into k equal-sized sets. We develop a general reduction which identifies some sufficient conditions on the considered graph class in order to prove the hardness of the problem. We focus on two combinatorially simple but very different classes, namely *trees* and *solid grid graphs*. The latter are finite connected subgraphs of the infinite two-dimensional grid without holes. We apply the reduction to show that for solid grid graphs it is NP-hard to approximate the optimum number of cut edges within any satisfactory ratio. We also consider solutions in which the sets may deviate from being equal-sized. Our reduction is applied to grids and trees to prove that no *fully polynomial time* algorithm exists that computes solutions in which the sets are arbitrarily close to equal-sized. This is true even if the number of edges cut is allowed to increase when the limit on the set sizes decreases. These are the first bicriteria inapproximability results for the k-BALANCED PARTITIONING problem.

Keywords: balanced partitioning; bicriteria approximation; inapproximability; grid graphs; trees.

*A preliminary version of this article appeared at the *37th International Symposium on Mathematical Foundations of Computer Science* [13]

1 Model and Setting

We consider the k-BALANCED PARTITIONING problem in which the n vertices of a graph need to be partitioned into k sets of size at most $\lceil n/k \rceil$ each. At the same time the *cut size*, which is the number of edges connecting vertices from different sets, needs to be minimised. This problem has many applications including VLSI circuit design [4], image processing [32], computer vision [24], route planning [6], and divide-and-conquer algorithms [26]. In our case the motivation (cf. [2, Section 4]) stems from parallel computations for finite element models (FEMs). In these a continuous domain of a physical model is discretised into a mesh of sub-domains (the elements). The mesh induces a graph in which the vertices are the elements and each edge connects neighbouring sub-domains. A vertex then corresponds to a computational task in the physical simulation, during which tasks that are adjacent in the graph need to exchange data. Since the model is usually very large, the computation is done in parallel. Hence the tasks need to be scheduled on to k machines (which corresponds to a partition of the vertices) so that the loads of the machines (the sizes of the sets in the partition) are balanced. At the same time the inter-processor communication (the cut size) needs to be minimised since this constitutes a bottleneck in parallel computing. Although we will give some conclusion for more general cases, in this article we mainly focus on 2D FEMs. For these the corresponding graph is a planar graph, typically given by a triangulation or a quadrilateral tiling of the plane [10]. We concentrate on the latter and consider so called *solid grid graphs* which model tessellations into squares. A *grid graph* is a finite subgraph of the infinite two-dimensional grid. An interior face of a grid graph is called a *hole* if more than four edges surround it. If a grid graph is connected and does not have any holes, it is called *solid*.

In general it is NP-hard to approximate the cut size of k-BALANCED PARTITIONING within any finite factor [1]. However the corresponding reduction relies on the fact that a general graph may not be connected and thus the optimal cut size can be zero. Since a 2D FEM always induces a connected planar graph, this strong hardness result may not apply. Yet even for trees [15] it is NP-hard to approximate the cut size within n^c, for any constant $c < 1$. The latter result however relies on the fact that the maximum degree of a tree can be arbitrarily large. Typically though, a 2D FEM induces a graph of constant degrees, as for instance in grid graphs. In fact, even though approximating the cut size in constant degree trees is APX-hard [15], there exists an $\mathcal{O}(\log(n/k))$ approximation algorithm [27] for these. This again raises the question of whether efficient approximation algorithms can be

found for graphs induced by 2D FEMs. In this article we give a negative answer to this question. We prove that it is NP-hard to approximate the cut size within n^c for any constant $c < 1/2$ for solid grid graphs. We also show that this is asymptotically tight by providing a corresponding approximation algorithm.

Hence when each set size is required to be at most $\lceil n/k \rceil$ (the *perfectly balanced* case), the achievable approximation factors are not satisfactory. To circumvent this issue, both in theory and practice it has proven beneficial to consider *bicriteria approximations*. Here additionally the sets may deviate from being perfectly balanced. The computed cut size is compared with the optimal perfectly balanced solution. Throughout this article we denote the approximation ratio on the cut size by α.

For planar graphs the famous Klein-Plotkin-Rao Theorem [22] can be combined with spreading metric techniques [11] in order to compute a solution for which $\alpha \in \mathcal{O}(1)$ and each set has size at most $2\lceil n/k \rceil$. This needs $\widetilde{\mathcal{O}}(n^3)$ time or $\widetilde{\mathcal{O}}(n^2)$ expected time. For the same guarantee on the set sizes, a faster algorithm exists for solid grid graphs [12]. It runs in $\widetilde{\mathcal{O}}(n^{1.5})$ time but approximates the cut size within $\alpha \in \mathcal{O}(\log k)$. However it is not hard to see how set sizes that deviate by a factor of 2 from being perfectly balanced may be undesirable for practical applications. For instance in parallel computing this means a significant slowdown. This is why graph partitioning heuristics such as Metis [19] or Scotch [5] allow to compute *near-balanced* partitions. Here each set has size at most $(1+\varepsilon)\lceil n/k \rceil$, for arbitrary $\varepsilon > 0$. The heuristics used in practice however do not give any guarantees on the cut size. For general graphs the best algorithm [15] known that gives such a guarantee, will compute a near-balanced solution for which $\alpha \in \mathcal{O}(\log n)$. However the running time of this algorithm increases exponentially with decreasing εs. Therefore this algorithm is too slow for practical purposes. Do algorithms exist that are both *fast* and compute *near-balanced* solutions, and for which rigorous guarantees can be given on the computed cut size? Note that the factor α of the above algorithm does not depend on ε. It therefore suggests itself to devise an algorithm that will compensate the cost of being able to compute near-balanced solutions not in the running time but in the cut size (as long as it does not increase too much). In this article however, we show that no such algorithm exists that is reasonable for practical applications. More precisely, we consider *fully polynomial time* algorithms for which the running time is polynomial in n/ε, for a value $\varepsilon > 0$ that is part of the input. We show that, unless P=NP, for solid grid graphs there is no such algorithm for which the computed solution is near-balanced and $\alpha = n^c/\varepsilon^d$, for any constants c and d where $c < 1/2$.

Our main contribution is a general reduction with which hardness results such as the two described above can be generated. For it we identify some sufficient conditions on the considered graphs which make the problem hard. Intuitively these conditions entail that cutting vertices from a graph must be expensive in terms of the number of edges used. We also apply the proposed reduction to general graphs and trees, in order to complement the known results. For general (disconnected) graphs we can show that, unless P=NP, there is no finite value for α allowing a fully polynomial time algorithm that computes near-balanced partitions. For trees we can prove that this is true for any $\alpha = n^c/\varepsilon^d$, for arbitrary constants c and d where $c < 1$. These results demonstrate that the identified sufficient conditions capture a fundamental trait of the k-`BALANCED PARTITIONING` problem. In particular since we prove the hardness for two combinatorially simple graph classes which however are very dissimilar (as for instance documented by the high *tree-width* of solid grids [8]). For solid grid graphs we harness their isoperimetric properties in order to satisfy the conditions, while for trees we use their ability to have high vertex degrees instead. These are the first bicriteria inapproximability results for the problem. We also show that all of them are asymptotically tight by giving corresponding approximation algorithms.

Related Work. Apart from the results mentioned above, Simon and Teng [31] gave a framework with which bicriteria approximations to k-`BALANCED PARTITIONING` can be computed. It is a recursive procedure that repeatedly uses a given algorithm for *sparsest cuts*. If a sparsest cut can be approximated within a factor of β then their algorithm obtains ratios $\varepsilon = 1$ and $\alpha \in \mathcal{O}(\beta \log k)$. The best factor β for general graphs [3] is $\mathcal{O}(\sqrt{\log n})$. For planar graphs Park and Phillips [29] show how to obtain $\beta \in \mathcal{O}(t)$ in $\widetilde{\mathcal{O}}(n^{1.5+1/t})$ time, for arbitrary t. On solid grid graphs constant approximations to sparsest cuts can be computed in linear time [14]. For general graphs the best ratio α is achieved by Krauthgamer *et al.* [23]. For $\varepsilon = 1$ they give an algorithm for which $\alpha \in \mathcal{O}(\sqrt{\log n \log k})$.

Near-balanced partitions were considered by Andreev and Räcke [1] who showed that a ratio of $\alpha \in \mathcal{O}(\log^{1.5}(n)/\varepsilon^2)$ is possible. This was later improved [15] to $\alpha \in \mathcal{O}(\log n)$, making α independent of ε. In the latter paper also a PTAS is given for trees. For perfectly balanced solutions, there is an approximation algorithm achieving $\alpha \in \mathcal{O}(\Delta \log_\Delta(n/k))$ for trees [27], where Δ is the maximum degree.

The special case when $k = 2$ (the `BISECTION` problem) has been thoroughly studied. The problem is NP-hard in general [18] and can be approx-

imated within $\mathcal{O}(\log n)$ [30]. Assuming the Unique Games Conjecture, no constant approximations are possible in polynomial time [21]. Also, unless NP$\subseteq \cap_{\epsilon>0}$ BPTIME(2^{n^ϵ}), no PTAS exists for this problem [20]. Leighton and Rao [25] show how near-balanced solutions for which $\alpha \in \mathcal{O}(\beta/\varepsilon^3)$ can be computed, where β is as above. In contrast to the case of arbitrary k, the `BISECTION` problem can be computed optimally in $\mathcal{O}(n^4)$ time for solid grid graphs [16], and in $\mathcal{O}(n^2)$ time for trees [27]. For planar graphs the complexity of `BISECTION` is unknown, but a PTAS exists [7].

2 A General Reduction

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 [17] which means that it remains so even if all integers are polynomially bounded in the size of the input.

Definition 1 (`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 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 computing near-balanced partitions and approximating the cut size within some α, is able to decide the `3-PARTITION` problem. 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 (disconnected) graphs which constitute an easily understandable case. For these graphs it is NP-hard to approximate the cut size within any finite factor [1]. We will show that, unless P=NP, no fully polynomial time algorithm exists for any α when near-balanced solutions are desired.

For any `3-PARTITION` instance we construct $3k$ 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 $2a_i$ vertices. This assures that the gadgets can be constructed in polynomial time since `3-PARTITION` is strongly NP-hard. In

general we will assume that we can construct $3k$ gadgets for the given graph class such that each gadget G_i has pa_i vertices for some parameter p specific to 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 we chose $p = 2$ and we let $m = 0$, i.e. the gadgets are disconnected. In order to show that the given gadgets can be used in a reduction, we require 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. For approximation ratios α and ε, the property we need is that cutting the graph containing n vertices into k sets using at most αm edges, produces less than $p - \varepsilon n$ minority vertices in total. Clearly ε needs to be sufficiently small so that the graph exists. When considering fully polynomial time algorithms, ε should however also not be too small since otherwise the running time may not be polynomial. For general graphs we achieve this by choosing $\varepsilon = (2ks)^{-1}$. This means that $p - \varepsilon n = 1$ since $n = \sum_{i=1}^{3k} pa_i = 2ks$. Simultaneously the running time of a corresponding algorithm is polynomial in the size of the `3-PARTITION` instance since `3-PARTITION` is strongly NP-hard. Additionally the desired condition is met for this graph class since no gadget can be cut using $\alpha m = 0$ edges. The following definition formalises the needed properties.

Definition 2. For each instance I of `3-PARTITION` with integers a_1 to a_{3k} and threshold s, a *reduction set for k-`BALANCED PARTITIONING`* contains a graph determined by some given parameters $m \geq 0$, $p \geq 1$, $\varepsilon \geq 0$, and $\alpha \geq 1$ which may depend on I. Such a graph constitutes $3k$ (disjoint) gadgets connected through m edges. Each gadget G_i, where $i \in \{1, \ldots, 3k\}$, has pa_i vertices. Additionally, if a partition of the n vertices of the graph into k sets has a cut size of at most αm, 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 reduction set to exist. For instance p must be an integer and ε must be sufficiently small compared to p and n. Since however the values will vary with the considered graph class we fix them only later. In the following lemma we will assume that the reduction set exists and therefore all parameters were chosen appropriately. It assures that given a reduction set, a bicriteria approximation algorithm for k-`BALANCED PARTITIONING`

can decide the 3-PARTITION problem. For general graphs we have seen above that a reduction set exists for any finite α and $\varepsilon = (2ks)^{-1}$. This means that a fully polynomial time algorithm for k-BALANCED PARTITIONING computing near-balanced partitions and approximating the cut size within α, can decide the 3-PARTITION problem in polynomial time. Such an algorithm can however not exist, unless P=NP.

Lemma 3. *Let an algorithm $\mathcal{A}$ be given that for any graph in a reduction set for k-BALANCED PARTITIONING computes a partition of the n vertices 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 a 3-PARTITION instance I, and let G be the graph corresponding to I in the reduction set. Assume that I has a solution. Then obviously cutting the m edges connecting the gadgets of G gives a perfectly balanced solution to I. Hence in this case the optimal solution has cut size at most m. Accordingly algorithm $\mathcal{A}$ will cut at most αm edges since it approximates the cut size by a factor of α. We will show that in the other case when I does not have a solution, the algorithm cuts more than αm edges. Hence $\mathcal{A}$ can decide the 3-PARTITION problem and the lemma follows.

For the sake of deriving a contradiction, assume that algorithm $\mathcal{A}$ cuts at most αm edges in case the 3-PARTITION instance I does not permit a solution. Since the corresponding graph G is from a reduction set for k-BALANCED PARTITIONING, by Definition 2 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 $\mathcal{A}$. Each gadget G_i, where $i \in \{1, \dots 3k\}$, of G has a *majority colour*, i.e. a colour that more than 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 instance I is trivial due to $s/4 < a_i < s/2$). The majority colours of the gadgets induce a partition $\mathcal{P}$ of the integers a_i of I 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 equals the colour of the set.

Since we assume that I does not admit a solution, if every set in $\mathcal{P}$ contains exactly three integers there must be some set 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 $i \in \{1, \dots, 3k\}$, mean that in case not every set in $\mathcal{P}$ contains exactly three elements, there must also exist a set 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 $p(s-1)$. Let w.l.o.g. the colour of T be 1. Apart from the vertices in these gadgets having majority colour 1, all vertices in G that also have colour 1 must be minority vertices. Hence there must be less than $p(s-1)+p-\varepsilon n$ many vertices with colour 1. 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 $n = pks$ 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 1. Thus the assumption that the algorithm cuts less than αm edges if I does not have a solution is wrong. □

3 Consequences for Grids and Trees

We will now consider solid grid graphs and trees to show the hardness of the k-`BALANCED PARTITIONING` problem when restricted to these. For grids we establish our results by considering a set of *rectangular* grid graphs which are connected in a row (Figure 1). By a rectangular grid graph we mean the Cartesian product of two paths. That is, given two paths P_1 and P_2 with vertex sets V_1 and V_2 respectively, their Cartesian product is a solid grid graph with vertex set $V_1 \times V_2$. Two vertices (v_1, v_2) and (w_1, w_2) of the grid are adjacent if for some $i \in \{1,2\}$ there is an edge between v_i and w_i in the path P_i, while $v_j = w_j$ for the other index $j \in \{1,2\} \setminus \{i\}$. Consider the natural planar embedding of a grid graph where vertices are coordinates in $\mathbb{N}^2$ and edges have unit length. The *width* of a rectangular grid graph is the number of vertices sharing the same y-coordinate in this embedding. Accordingly the *height* is the number sharing the same x-coordinate. We first prove that such topologies can be used for reduction sets. We satisfy the conditions by observing that a grid graph resembles a discretised polygon and hence shares their isoperimetric properties. This fact was already used in [28] and we harness these results in the following lemma.

Lemma 4. *Let $\varepsilon \geq 0$ and $\alpha \geq 1$. For any `3-PARTITION` instance, let a solid grid graph G with n vertices be given that consists of $3k$ rectangular grids*

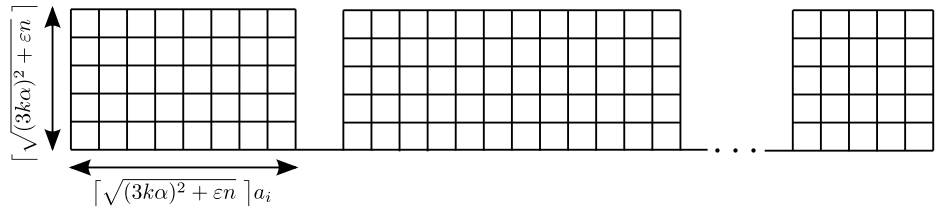


Figure 1: The solid grid constructed for the reduction from 3-PARTITION. The gadgets which are rectangular grids are connected through the bottom left and right vertices.

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 $i \in \{1, \dots 3k\}$, be $\lceil\sqrt{(3k\alpha)^2 + \varepsilon n}\rceil$ and $\lceil\sqrt{(3k\alpha)^2 + \varepsilon n}\rceil a_i$, respectively. If ε and α are values for which these grids exist, they form a reduction set for k-BALANCED PARTITIONING.

Proof. Consider one of the described graphs G 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 [28, Lemma 2] it follows 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. Such a cut will cut out at most $(\alpha m/2)^2$ vertices. Hence if the vertices of the grid graph G are cut into k sets using at most αm edges, then the induced colouring contains 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 first used in the following theorem to show that no satisfying fully polynomial time algorithm exists.

Theorem 5. *Unless P=NP, there is no fully polynomial time algorithm for the k-BALANCED PARTITIONING problem on solid grid graphs that for any $\varepsilon > 0$ computes a solution in which each set has size at most $(1+\varepsilon)\lceil n/k\rceil$ and where $\alpha = n^c/\varepsilon^d$, for any constants c and d where $c < 1/2$.*

Proof. In order to prove the claim we need to show that a reduction set as suggested by Lemma 4 exists and can be constructed in polynomial time. We first prove the existence by showing that the construction given by Lemma 4

is feasible for any 3-PARTITION instance. Since α and therefore the sizes of the gadgets depend on n, we need to determine the number of vertices n prior to the construction of the grid. The algorithm for k-BALANCED PARTITIONING can compute a near-balanced partition for any $\varepsilon > 0$. Hence we can set $\varepsilon = (2ks)^{-1}$ so that $\alpha = n^c(2ks)^d$. For a solid grid graph as suggested by Lemma 4 the parameter p is determined by the width and height of the gadgets. Hence $p = \lceil \sqrt{(3k\alpha)^2 + \varepsilon n} \rceil^2$ and the number of vertices is

$$n = \sum_{i=1}^{3k} pa_i = \left\lceil \sqrt{(3kn^c(2ks)^d)^2 + n/(2ks)} \right\rceil^2 \cdot ks. \tag{1}$$

Determining the non-zero solution for n in this equation will give us the number of vertices. For this we analyse the right-hand side of the equation as a function of n while ignoring the ceiling function. That is, we are interested in the function

$$f(n) = \left(\sqrt{(3kn^c(2ks)^d)^2 + n/(2ks)} \right)^2 \cdot ks = 9k^3 s(2ks)^{2d} n^{2c} + n/2.$$

It is easy to see that the points for which $f(n) = n$ are $n_0 = 0$ and $n_1 = (18k^3 s(2ks)^{2d})^{1/(1-2c)}$. Setting n to n_1 in the right-hand side of Equation (1) gives an upper bound $u \in \mathbb{N}$ on $f(n_1)$, which is integer valued due to the ceiling function and the fact that $k, s \in \mathbb{N}$. Let n_2 be the value for which $f(n_2) = u$, which is well-defined and greater than n_1 since $f(n)$ is strictly increasing. Note that since $c < 1/2$ the second derivative of the function $f(n)$ is negative. That is, $f(n)$ is concave which, by definition of n_0 and n_1, means that $f(n) \geq n$ for any $n \in [n_0, n_1]$ and also $f(n) \leq n$ for any $n \notin [n_0, n_1]$. Consequently $f(n_2) \leq n_2$ from which we can conclude that

$$n_1 = f(n_1) \leq u = f(n_2) \leq n_2.$$

Hence $u \in [n_1, n_2]$. Since $f(n)$ is strictly increasing, the right-hand side of Equation (1) equals u for any $n \in [n_1, n_2]$. This means that the non-zero solution to Equation (1) is the value u.

By first calculating u, the construction of Lemma 4 can be carried out given a 3-PARTITION instance I. Since c and d are constants and 3-PARTITION is strongly NP-hard, n_1 is polynomially bounded in the size of I. Clearly u is therefore also polynomially bounded. Hence a grid graph as suggested by Lemma 4 can be constructed in polynomial time. For $\varepsilon = (2ks)^{-1}$ a fully polynomial time algorithm has a running time that is polynomial in the size of the instance I when executed on the corresponding

grid. However, unless P=NP, this algorithm cannot exist since it decides the 3-PARTITION problem due to Lemma 3. □

Next we consider computing perfectly balanced partitions. Note that we may set $\varepsilon = 0$ in Lemma 3 for this purpose. The following theorem shows the NP-hardness of k-BALANCED PARTITIONING on solid grids. It therefore considers running times that are polynomial solely in the number of vertices and do not depend on some input parameter ε.

Theorem 6. *There is no polynomial time 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 first need to show that a reduction set as proposed in Lemma 4 exists in order to use it together with Lemma 3. To prove the existence we determine the number of vertices of a grid as suggested by Lemma 4. If this number is finite the construction of Lemma 4 is feasible. 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 $\lceil 3k\alpha \rceil$ and $\lceil 3k\alpha \rceil a_i$, respectively, for a gadget G_i. Hence the number of vertices of the resulting grid is

$$n = \sum_{i=1}^{3k} pa_i \leq \lceil 3kn^c \rceil^2 \cdot ks. \tag{2}$$

Similar to the proof of Theorem 5, the non-zero solution of n in this equation can be determined by considering the function $f(n) = 9k^3sn^{2c}$, i.e. the right-hand side of Equation (2) ignoring the ceiling function. The non-zero point n_1 at which $f(n) = n$ is $n_1 = (9k^3s)^{1/(1-2c)}$. The function $f(n)$ is again strictly increasing and concave if $c < 1/2$. Hence the same arguments as in the proof of Theorem 5 can be used to show that the number of vertices is the integer value u obtained by setting n to n_1 in the right-hand side of Equation (2). Additionally it is polynomial in the size of the 3-PARTITION instance since c is a constant and 3-PARTITION is strongly NP-hard. Hence the grids can be constructed in polynomial time given the value u and the integers of a 3-PARTITION instance. By Lemma 3, a polynomial time algorithm which computes a perfectly balanced partition on any grid given by the reduction set, and which approximates the cut size within α, can decide the 3-PARTITION problem in polynomial time. This gives a contradiction unless P=NP, which concludes the proof. □

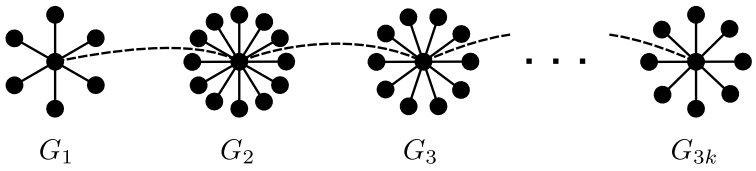


Figure 2: The tree constructed for the reduction from 3-PARTITION. The gadgets which are stars are connected through their centre vertices.

Lemma 4 shows that for solid grid graphs the hardness derives from their isoperimetric properties. Trees do not experience such qualities. However they may have high vertex degrees, which grids cannot. The following theorem shows that this property also leads to a similar hardness as for solid grid graphs.

Theorem 7. *Unless P=NP, there is no fully polynomial time algorithm for the k-BALANCED PARTITIONING problem on trees that for any $\varepsilon > 0$ computes a solution in which each set has size at most $(1+\varepsilon)\lceil n/k \rceil$ and where $\alpha = n^c/\varepsilon^d$, for any constants c and d where $c < 1$.*

Proof. We need to identify a reduction set for k-BALANCED PARTITIONING containing trees. We use very similar gadgets to those used in [15, Theorem 2], despite the fact that the proof idea in the latter paper is different from the one employed in Lemma 3. Each gadget of a tree in our reduction set is a star (Figure 2) and these are connected in a path through their centre vertices using $m = 3k - 1$ edges. The number of vertices in each star G_i is pa_i where $p = \lceil 3k\alpha + \varepsilon n \rceil$. Using at most αm, i.e. less than $3k\alpha$, edges to cut off vertices from a single star, less than $3k\alpha$ leaves will be cut off. At the same time more than half the vertices of the star are still connected to the centre vertex. This is because each star contains at least $6k\alpha$ vertices since we can safely assume that $a_i \geq 2$ for each $i \in \{1, \ldots, 3k\}$ (otherwise the 3-PARTITION instance is trivial due to $s/4 < a_i < s/2$). Therefore partitioning the vertices of all gadgets into k sets using at most αm edges will in total produce less than $3k\alpha$ minority vertices in the induced colouring. This establishes the desired upper bound on the number of minority vertices for the reduction set since $3k\alpha \leq p - \varepsilon n$.

In order to prove the claim we need to show that for the given parameters a reduction set as suggested above exists and can be constructed in polynomial

time. We first prove the existence by determining the number of vertices in a tree of the reduction set. Since the algorithm for k-BALANCED PARTITIONING can compute a near-balanced partition for any $\varepsilon > 0$ we set $\varepsilon = (2ks)^{-1}$. Thus the number of vertices in a tree of the reduction set is

$$n = \sum_{i=1}^{3k} pa_i = \lceil 3kn^c(2ks)^d + n/(2ks) \rceil \cdot ks. \tag{3}$$

As in the proof of Theorem 5 we can determine the non-zero solution of n to this equality by considering the right-hand side and ignoring the ceiling function. This gives the function $f(n) = 3k^2s(2ks)^d n^c + n/2$. The non-zero point n_1 at which $f(n) = n$ in this case is $n_1 = (6k^2s(2ks)^d)^{1/(1-c)}$. The function $f(n)$ is strictly increasing and concave since $c < 1$. Hence, similar to the proof of Theorem 5, we can determine the number of vertices by setting n to n_1 in the right-hand side of Equation (3). Given a 3-PARTITION instance the construction of a tree of the reduction set can thus be carried out by first calculating the number of vertices in the tree. Additionally, if c and d are constants such a construction can be done in polynomial time since 3-PARTITION is strongly NP-hard.

For $\varepsilon = (2ks)^{-1}$ a fully polynomial time algorithm has a running time that is polynomial in the 3-PARTITION instance when executed on the corresponding tree in the reduction set. However, unless P=NP, this algorithm cannot exist since it decides the 3-PARTITION problem due to Lemma 3. □

4 Conclusions

Are there algorithms for the k-BALANCED PARTITIONING problem that are both *fast* and compute *near-balanced solutions*, even when allowing the cut size to increase when ε decreases? We gave a negative answer to this question. This means that the tradeoff between fast running time and good solution quality, as provided by the algorithms mentioned in the introduction, is unavoidable. In particular the running time to compute near-balanced solutions [15] has to increase exponentially with ε. Also our results draw a frontier of the hardness of the problem by showing how the cut size needs to increase with decreasing ε for fully polynomial time algorithms. These are the first bicriteria inapproximability results for the k-BALANCED PARTITIONING problem.

To show the tightness of the achieved results, for grid graphs we harness results by Diks *et al.* [9]. They provide a polynomial time algorithm to cut out any number of vertices using at most $\mathcal{O}(\sqrt{\Delta n})$ edges from a planar graph

with maximum degree Δ. Since $\Delta = 4$ in a grid graph, it is possible to repeatedly cut out $\lceil n/k \rceil$ vertices from the grid using $\mathcal{O}(k\sqrt{n})$ edges in total. A perfectly balanced partition needs at least $k-1$ edges in a connected graph. Hence we obtain an $\alpha \in \mathcal{O}(\sqrt{n})$ approximation algorithm for grids. This shows that both the hardness results we gave for these graphs are asymptotically tight, since the algorithm runs in (fully) polynomial time. For trees a trivial approximation algorithm can cut all edges in the graph and thereby obtain $\alpha = n$. This shows that also the achieved result for trees is asymptotically tight.

We were able to show that both trees and grids experience similar hardness. This is remarkable since these graphs have entirely different combinatorial properties. On the other hand, it emphasises the ability of the given reduction framework to capture a fundamental trait of the k-`BALANCED PARTITIONING` problem. It remains to be seen what other structural properties can be harnessed for our framework, in order to prove the hardness for entirely different graph classes. Another interesting approach would be to take the opposite view and identify properties of graphs that in fact lead to good approximation algorithms.

It is interesting to note though that the isoperimetric properties that we used to establish the hardness for grid graphs, can be further exploited to show the hardness for related graph classes. For instance a graph class that is independent of solid grid graphs (in that neither class contains the other) are non-solid grids of simple shape. However even for the case when both the holes and the grids are restricted to rectangular shapes, our reduction can be used to give similar hardness results as for solid grid graphs. This can easily be seen by adding edges to the gadgets used in our reduction set (Figure 1) to connect the top most left and right vertices, respectively. The constants of the respective parameters can readily be adapted to compensate for the additional edges.

Another graph class for which some additional interesting observations can be made are δ-dimensional grids. Clearly the presented results are also valid for these since solid grid graphs are a special case. However one can show that the hardness in fact grows with the dimension δ. For example in a 3-dimensional cuboid shaped grid, it is easy to see that the maximum number of vertices that can be cut out using αm edges is in the order of $(\alpha m)^{3/2}$ if the grid is large enough. Such cuboid grids can be used as gadgets for a reduction set. This leads to hardness results where the constant c can take values up to $2/3$, i.e. the inverse of the former exponent. By exploiting the isoperimetric properties of grids in higher dimensions accordingly, one can show that the corresponding constant c can take values up to $1 - 1/\delta$.

Note that the respective ratios α of the bicriteria inapproximability results can in each case be amplified arbitrarily due to the unrestricted constant d. Also, we are able to provide reduction sets for graphs that resemble those resulting from 2D FEMs (solid grid graphs, or grid graphs with holes of simple shape) and even 3D FEMs (3-dimensional grids) which are widely used in practice. For these reasons our results imply that completely different methods (possibly randomness or fixed parameter tractability) must be employed in order to find fast practical algorithms with rigorously bounded approximation guarantees.

References

[1] K. Andreev and H. Räcke. Balanced graph partitioning. *Theory of Computing Systems*, 39(6):929–939, 2006.

[2] P. Arbenz, G. van Lenthe, U. Mennel, R. Müller, and M. Sala. Multi-level μ-finite element analysis for human bone structures. In *Proceedings of the 8th Workshop on State-of-the-art in Scientific and Parallel Computing (PARA)*, pages 240–250, 2007.

[3] S. Arora, S. Rao, and U. Vazirani. Expander flows, geometric embeddings and graph partitioning. In *Proceedings of the 26th annual ACM symposium on Theory of computing (STOC)*, pages 222–231, 2004.

[4] S. Bhatt and F. T. Leighton. A framework for solving VLSI graph layout problems. *Journal of Computer and System Sciences*, 28(2):300–343, 1984.

[5] C. Chevalier and F. Pellegrini. PT-Scotch: A tool for efficient parallel graph ordering. *Parallel Computing*, 34(68):318–331, 2008.

[6] D. Delling, A. Goldberg, T. Pajor, and R. Werneck. Customizable route planning. *Experimental Algorithms*, pages 376–387, 2011.

[7] J. Díaz, M. J. Serna, and J. Torán. Parallel approximation schemes for problems on planar graphs. *Acta Informatica*, 33(4):387–408, 1996.

[8] R. Diestel, T. R. Jensen, K. Y. Gorbunov, and C. Thomassen. Highly connected sets and the excluded grid theorem. *Journal of Combinatorial Theory, Series B*, 75(1):61–73, 1999.

[9] 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.

[10] H. Elman, D. Silvester, and A. Wathen. *Finite Elements and Fast Iterative Solvers: with Applications in Incompressible Fluid Dynamics*. Oxford University Press, USA, 2005.

[11] G. Even, J. Naor, S. Rao, and B. Schieber. Fast approximate graph partitioning algorithms. *SIAM Journal on Computing*, 28(6):2187–2214, 1999.

[12] A. E. Feldmann. *Balanced Partitioning of Grids and Related Graphs: A Theoretical Study of Data Distribution in Parallel Finite Element Model Simulations*. PhD thesis, ETH Zurich, April 2012. Diss.-Nr. ETH: 20371.

[13] A. E. Feldmann. Fast balanced partitioning is hard, even on grids and trees. In *Proceedings of the 37th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 372–382, 2012.

[14] 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 (WG)*, pages 143–154, 2011.

[15] A. E. Feldmann and L. Foschini. Balanced partitions of trees and applications. In *29th International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 100–111, 2012.

[16] 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 (ESA)*, pages 143–154, 2011.

[17] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Co., 1979.

[18] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976.

[19] G. Karypis and V. Kumar. METIS-unstructured graph partitioning and sparse matrix ordering system, version 2.0. Technical report, University of Minnesota, 1995.

[20] S. Khot. Ruling out PTAS for graph min-bisection, dense k-subgraph, and bipartite clique. *SIAM Journal on Computing*, 36(4):1025–1071, 2006.

[21] S. A. Khot and N. K. Vishnoi. The Unique Games Conjecture, integrality gap for cut problems and embeddability of negative type metrics into ℓ_1. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 53–62, 2005.

[22] 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 (STOC)*, pages 682–690, 1993.

[23] R. Krauthgamer, J. Naor, and R. Schwartz. Partitioning graphs into balanced components. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 942–949, 2009.

[24] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick. Graphcut textures: Image and video synthesis using graph cuts. *ACM Transactions on Graphics*, 22(3):277–286, 2003.

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

[26] R. Lipton and R. Tarjan. Applications of a planar separator theorem. *SIAM Journal on Computing*, 9:615–627, 1980.

[27] R. M. MacGregor. *On Partitioning a Graph: a Theoretical and Empirical Study.* PhD thesis, University of California, Berkeley, 1978.

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

[29] J. K. Park and C. A. Phillips. Finding minimum-quotient cuts in planar graphs. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing (STOC)*, pages 766–775, 1993.

[30] H. Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 255–264, 2008.

[31] H. D. Simon and S. H. Teng. How good is recursive bisection? *SIAM Journal on Scientific Computing*, 18(5):1436–1445, 1997.

[32] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1101–1113, 1993.