Fast Balanced Partitioning Is Hard
Even on Grids and Trees

Andreas Emil Feldmann
Institute of Theoretical Computer Science, ETH Zürich, Switzerland
feldmann@inf.ethz.ch

Abstract. Two kinds of approximation algorithms exist for the \textit{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 paper we prove that this tradeoff between runtime 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 \textit{trees} and \textit{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 \textit{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 \textit{k-BALANCED PARTITIONING} problem.

1 Model and Setting

We consider the \textit{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 \textit{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 [29], route planning [6], and divide-and-conquer algorithms [23]. 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

\* The author is supported by the Swiss National Science Foundation under grant 200021_125201/1.
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 interprocessor communication (the cut size) needs to be minimised since this constitutes a bottleneck in parallel computing. In this paper we 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 correspond to tessellations into squares. A grid graph is a finite subgraph of the infinite 2D 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 [14] 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 [14], there exists an \( O(\log(n/k)) \) approximation algorithm [24] for these. This again raises the question of whether efficient approximation algorithms can be found for graphs induced by 2D FEMs. In this paper 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 paper we denote the approximation ratio on the cut size by \( \alpha \).

For planar graphs the famous Klein-Plotkin-Rao Theorem [20] can be combined with spreading metric techniques [11] in order to compute a solution for which \( \alpha \in O(1) \) and each set has size at most \( 2\lceil n/k \rceil \). This needs \( \tilde{O}(n^3) \) time or \( \tilde{O}(n^2) \) expected time. For the same guarantee on the set sizes, a faster algorithm exists for solid grid graphs [12]. It runs in \( \tilde{O}(n^{1.5}) \) time but approximates the cut size within \( \alpha \in O(\log k) \). 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 [18] or Scotch [5] allow to