THEORY OF COMPUTING, Volume 17 (6), 2021, pp. 1–57
www.theoryofcomputing.org

Almost Polynomial Hardness of Node-Disjoint Paths in Grids

Julia Chuzhoy* David Hong Kyun Kim† Rachit Nimavat‡

Received August 2, 2018; Revised April 9, 2020; Published September 22, 2021

Abstract. In the classical Node-Disjoint Paths (NDP) problem, we are given an n-vertex graph $G = (V,E)$, and a collection $\mathcal{M} = \{(s_1,t_1),\dots,(s_k,t_k)\}$ of pairs of its vertices, called source-destination, or demand pairs. The goal is to route as many of the demand pairs as possible, where to route a pair we need to select a path connecting it, so that all selected paths are disjoint in their vertices. The best current algorithm for NDP achieves an $O(\sqrt{n})$-approximation [Kolliopoulos, Stein, *Math. Progr.* 2004], while the best current negative result is a factor $2^{\Omega(\sqrt{\log n})}$-hardness of approximation unless $\mathsf{NP} \subseteq \mathsf{DTIME}(n^{O(\log n)})$ [Chuzhoy, Kim, Nimavat, STOC'17], even if the underlying graph is a **subgraph** of a grid graph; unfortunately, this result does not extend to grid graphs.

The approximability of the NDP problem on grid graphs has remained a tantalizing open question, with the best current upper bound of $\tilde{O}(n^{1/4})$, and the best current lower bound of APX-hardness [Chuzhoy, Kim, APPROX'15] only ruling out a $(1+\delta)$-approximation algorithm for some fixed $\delta > 0$, assuming that $\mathsf{P} \neq \mathsf{NP}$. In this paper we come close to

An extended abstract of this paper appeared in the Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC'18) [20].

*Supported in part by NSF grants CCF-1318242 and CCF-1616584.
†Supported in part by NSF grant CCF-1318242 and CCF-1616584.
‡Supported in part by NSF grant CCF-1318242 and CCF-1616584.

ACM Classification: F.2.2

AMS Classification: 68Q17, 68Q25, 68W25

Key words and phrases: disjoint paths, graph routing, hardness of approximation

DOI: 10.4086/toc.2021.v017a006

resolving the approximability of NDP in general, and of NDP in grids in particular. Our main result is that NDP is $2^{\Omega(\log^{1-\varepsilon} n)}$-hard to approximate for any constant ε, assuming that $\mathsf{NP} \not\subseteq \mathsf{DTIME}(n^{\mathrm{poly}\log n})$, and that it is $n^{\Omega(1/(\log\log n)^2)}$-hard to approximate, assuming that for some constant $\delta > 0$, $\mathsf{NP} \not\subseteq \mathsf{DTIME}(2^{n^\delta})$. These results hold even for grid graphs and wall graphs, and extend to the closely related Edge-Disjoint Paths problem, even in wall graphs.

Our hardness proof performs a reduction from the 3COL(5) problem to NDP, using a new graph partitioning problem as a proxy. Unlike the more standard approach of employing Karp reductions to prove hardness of approximation, our proof is a Cook-type reduction, where, given an input instance of 3COL(5), we produce a large number of instances of NDP, and apply an approximation algorithm for NDP to each of them.

Contents

1 Introduction

We study the Node-Disjoint Paths (NDP) problem: given an undirected n-vertex graph G and a collection $\mathcal{M} = \{(s_1,t_1),\ldots,(s_k,t_k)\}$ of pairs of its vertices, called *source-destination*, or *demand* pairs, we are interested in *routing* the demand pairs: in order to route a pair (s_i,t_i), we need to select a path connecting s_i to t_i. The goal is to route as many of the pairs as possible, subject to the constraint that the selected routing paths are mutually disjoint in their vertices and their edges. We let $S = \{s_1,\ldots,s_k\}$ be the set of the source vertices, $T = \{t_1,\ldots,t_k\}$ the set of the destination vertices, and we refer to the vertices of $S \cup T$ as *terminals*. We denote by NDP-Planar the special case of the problem where the graph G is planar; by NDP-Grid the special case where G is a square grid; and by NDP-Wall the special case where G is a wall (see Figure 1 for an illustration of a wall and Section 2 for its formal definition).

NDP is a fundamental problem in the area of graph routing, that has been studied extensively. Robertson and Seymour [49, 50] showed, as part of their famous Graph Minors Series, an efficient algorithm for solving the problem if the number k of the demand pairs is bounded by a constant. However, when k is a part of input, the problem becomes NP-hard [30, 23], and it remains NP-hard even for planar graphs [41], and for grid graphs [38]. The best current upper bound on the approximability of NDP is $O(\sqrt{n})$, obtained by a simple greedy algorithm [37]. Until recently, the best known lower bound was an $\Omega(\log^{1/2-\varepsilon} n)$-hardness of approximation for any constant ε, unless $\mathsf{NP} \subseteq \mathsf{ZPTIME}(n^{\mathrm{poly}\log n})$ [5, 4]. For the special cases of NDP-Planar and NDP-Grid [17], the best known lower bound was APX-hardness, so that they do not have a $(1+\delta)$-approximation algorithm, for some fixed $\delta > 0$, unless $\mathsf{P} = \mathsf{NP}$. In a recent paper [19], the authors have shown an improved $2^{\Omega(\sqrt{\log n})}$-hardness of approximation for NDP, assuming that $\mathsf{NP} \not\subseteq \mathsf{DTIME}(n^{O(\log n)})$. This result holds even for planar graphs with maximum vertex degree 3, where all source vertices lie on the boundary of a single face, and for **subgraphs** of grid graphs, with all source vertices lying on the boundary of the grid. We note that for general planar graphs, the $O(\sqrt{n})$-approximation algorithm of [37] was recently slightly improved to an $\tilde{O}(n^{9/19})$-approximation [18].

The approximability status of NDP-Grid—the special case of NDP where the underlying graph is a square grid—remained a tantalizing open question. The study of this problem dates back to the 70's, and was initially motivated by applications to VLSI design. As grid graphs are extremely well-structured, one would expect that good approximation algorithms can be designed for them, or that, at the very least, they should be easy to understand. However, establishing the approximability of NDP-Grid has been elusive so far. The simple greedy $O(\sqrt{n})$-approximation algorithm of [37] was only recently improved to a $\tilde{O}(n^{1/4})$-approximation for NDP-Grid [17], while on the negative side only APX-hardness is known. In a very recent paper [21], the authors designed a $2^{O(\sqrt{\log n}\cdot\log\log n)}$-approximation algorithm for a special case of NDP-Grid, where the source vertices appear on the grid boundary. This result can be seen as complementing the $2^{\Omega(\sqrt{\log n})}$-hardness of approximation of NDP on subgraphs of grids with all sources lying on the grid boundary [19].[1] Furthermore, this result can be seen as suggesting that subpolynomial approximation algorithms may be achievable for NDP-Grid.

In this paper we show that this is unlikely to be the case, and come close to resolving the approximability status of NDP-Grid, and of NDP in general, by showing that NDP-Grid is $2^{\Omega(\log^{1-\varepsilon} n)}$-hard to approximate for any constant ε, unless[2] $\mathsf{NP} \subseteq \mathsf{DTIME}(n^{\mathrm{poly}\log n})$. We further show that it is $n^{\Omega(1/(\log\log n)^2)}$-hard to approximate, assuming that for some constant $\delta > 0$, $\mathsf{NP} \not\subseteq \mathsf{DTIME}(2^{n^{\delta}})$. The same hardness results also extend to NDP-Wall. These hardness results are stronger than the best currently known hardness for the general NDP problem, and should be contrasted with the $2^{O(\sqrt{\log n}\cdot\log\log n)}$-approximation algorithm for NDP-Grid with all sources lying on the grid boundary [21].

Another basic routing problem that is closely related to NDP is Edge-Disjoint Paths (EDP). The input to this problem is the same as before: an undirected graph $G = (V, E)$ and a set $\mathcal{M} = \{(s_1, t_1), \ldots, (s_k, t_k)\}$ of demand pairs. The goal, as before, is to route the largest number of the demand pairs via paths. However, we now allow the paths to share vertices, and only require that they are mutually edge-disjoint. In general, it is easy to see that EDP is a special case of NDP. Indeed, given an EDP instance $(G, \mathcal{M})$, computing the line graph of the input graph G transforms it into an equivalent instance of NDP. However, this transformation may inflate the number of the graph vertices, and so approximation factors that depend on $|V(G)|$ may no longer be preserved. Moreover, this transformation does not preserve planarity, and no such relationship is known between NDP and EDP in planar graphs. The approximability status of EDP is very similar to that of NDP: the best current approximation algorithm achieves an $O(\sqrt{n})$-approximation factor [13], and the recent $2^{\Omega(\sqrt{\log n})}$-hardness of approximation of [19], under the assumption that $\mathsf{NP} \not\subseteq \mathsf{DTIME}(n^{O(\log n)})$, extends to EDP. Interestingly, EDP appears to be relatively easy on grid graphs, and has a constant-factor approximation for this special case [6, 36, 35]. The analogue of the grid graph in the setting of EDP seems to be the wall graph (see Figure 1): the approximability status of EDP on wall graphs is similar to that of NDP on grid graphs, with the best current upper bound of $\tilde{O}(n^{1/4})$, and the best lower bound of APX-hardness [17]. The results of [19] extend to a $2^{\Omega(\sqrt{\log n})}$-hardness of approximation for EDP on subgraphs of wall graphs, with all source vertices lying on the wall boundary, under the same complexity assumption. We denote by EDP-Wall the special case of the EDP problem where the underlying graph is a wall. We show that our new almost polynomial hardness-of-approximation results also hold for EDP-Wall and for NDP-Wall.

[1] Note that the results are not strictly complementary: the algorithm only works for grid graphs, while the hardness result is only valid for subgraphs of grids.

[2] The original version of this paper contained a randomized reduction, with a somewhat stronger complexity assumptions that $\mathsf{NP} \not\subseteq \mathsf{RTIME}(n^{\mathrm{poly}\log n})$ and $\mathsf{NP} \not\subseteq \mathsf{RTIME}(2^{n^{\delta}})$, respectively. The derandomization strategy that we have used was proposed to us by a reviewer.

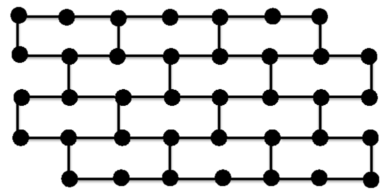

Figure 1: A wall graph.

Other related work. Several other special cases of EDP are known to have reasonably good approximation algorithms. For example, for the special case of Eulerian planar graphs, Kleinberg [33] showed an $O(\log^2 n)$-approximation algorithm, while Kawarabayashi and Kobayashi [31] provide an improved $O(\log n)$-approximation for both Eulerian and 4-connected planar graphs. Polylogarithmic approximation algorithms are also known for bounded-degree expander graphs [39, 10, 9, 34, 27], and constant-factor approximation algorithms are known for trees [28, 15], and grids and grid-like graphs [6, 7, 36, 35]. Rao and Zhou [47] showed an efficient randomized $O(\operatorname{poly}\log n)$-approximation algorithm for the special case of EDP where the value of the global minimum cut in the input graph is $\Omega(\log^5 n)$. Recently, Fleszar et al. [26] designed an $O(\sqrt{r} \cdot \log(kr))$-approximation algorithm for EDP, where r is the feedback vertex set number of the input graph $G = (V,E)$—the smallest number of vertices that need to be deleted from G in order to turn it into a forest.

A natural variation of NDP and EDP that relaxes the disjointness constraint by allowing a small vertex- or edge-congestion has been a subject of extensive study. In the NDP with Congestion (NDPwC) problem, the input consists of an undirected graph and a set of demand pairs as before, and additionally a non-negative integer c. The goal is to route a maximum number of the demand pairs with congestion c, that is, each vertex may participate in at most c paths in the solution. The EDP with Congestion problem (EDPwC) is defined similarly, except that now the congestion is measured on the graph edges and not vertices. The famous result of Raghavan and Thompson [44], that introduced the randomized LP-rounding technique, obtained a constant-factor approximation for NDPwC and EDPwC, for a congestion value $c = \Theta(\log n/\log\log n)$. A long sequence of work [12, 43, 3, 47, 16, 22, 11] has led to an $O(\operatorname{poly}\log k)$-approximation for EDPwC with congestion bound $c = 2$, and for NDPwC with constant congestion. These results are essentially optimal, since it is known that for every constant ε, and for every congestion value $c = o(\log\log n/\log\log\log n)$, both problems are hard to approximate to within a factor $\Omega((\log n)^{\frac{1-\varepsilon}{c+1}})$, unless $\mathsf{NP} \subseteq \mathsf{ZPTIME}(n^{\operatorname{poly}\log n})$ [4]. When the input graph is planar, Seguin-Charbonneau and Shepherd [51], improving on the result of Chekuri, Khanna and Shepherd [14], have shown a constant-factor approximation for EDPwC with congestion 2.

Our results and techniques. Our main result is the proof of the following two theorems.

Theorem 1.1. *For every constant $\varepsilon > 0$, there is no $2^{O(\log^{1-\varepsilon} n)}$-approximation polynomial-time algorithm for* NDP, *assuming that* $\mathsf{NP} \not\subseteq \mathsf{DTIME}(n^{\operatorname{poly}\log n})$. *Moreover, there is no $n^{O(1/(\log\log n)^2)}$-approximation polynomial-time algorithm for* NDP, *assuming that for some constant $\delta > 0$,* $\mathsf{NP} \not\subseteq \mathsf{DTIME}(2^{n^{\delta}})$. *These* *results hold even when the input graph is a grid graph or a wall graph.*

Theorem 1.2. *For every constant $\varepsilon > 0$, there is no $2^{O(\log^{1-\varepsilon} n)}$-approximation polynomial-time algorithm for* EDP*, assuming that* $\mathsf{NP} \not\subseteq \mathsf{DTIME}(n^{\mathrm{poly}\log n})$*. Moreover, there is no $n^{O(1/(\log\log n)^2)}$-approximation polynomial-time algorithm for* EDP*, assuming that for some constant $\delta > 0$,* $\mathsf{NP} \not\subseteq \mathsf{DTIME}(2^{n^\delta})$*. These results hold even when the input graph is a wall graph.*

We now provide a high-level overview of our techniques. The starting point of our hardness-of-approximation proof is 3COL(5)—a special case of the 3-coloring problem, where the underlying graph is 5-regular. We define a new graph partitioning problem, that we refer to as (r,h)-Graph Partitioning, and denote by (r,h)-GP. In this problem, we are given a bipartite graph $\tilde{G} = (V_1, V_2, E)$ and two integral parameters $r, h > 0$. A solution to the problem is a partition $(W_1, W_2, \dots, W_r)$ of $V_1 \cup V_2$ into r subsets, and for each $1 \le i \le r$, a subset $E_i \subseteq E(W_i)$ of edges, so that $|E_i| \le h$ holds, and the goal is to maximize $\sum_{i=1}^{r} |E_i|$. A convenient intuitive way to think about this problem is that we would like to partition $\tilde{G}$ into a large number of subgraphs, in a roughly balanced way (with respect to the number of edges), so as to preserve as many of the edges as possible. We show that NDP-Grid is at least as hard to approximate as the (r,h)-GP problem (to within polylogarithmic factors). Our reduction exploits the fact that routing in grids is closely related to graph drawing, and that graphs with small crossing number have small balanced separators. The (r,h)-GP problem itself appears similar in flavor to the Densest k-Subgraph problem (DkS). In the DkS problem, we are given a graph $G = (V, E)$ and a parameter k, and the goal is to find a subset $U \subseteq V$ of k vertices, that maximizes the number of edges in the induced graph $G[U]$. Intuitively, in the (r,h)-GP problem, the goal is to partition the graph into many dense subgraphs, and so in order to prove that (r,h)-GP is hard to approximate, it is natural to employ techniques used in hardness-of-approximation proofs for DkS. The best current approximation algorithm for DkS achieves a $n^{1/4+\varepsilon}$-approximation for any constant ε [8]. Even though the problem appears to be very hard to approximate, its hardness-of-approximation proof has been elusive until recently: only constant-factor hardness results were known for DkS under various worst-case complexity assumptions, and $2^{\Omega(\log^{2/3} n)}$-hardness under average-case assumptions [24, 1, 32, 45]. In a recent breakthrough, Manurangsi [42] has shown that for some constant c, DkS is hard to approximate to within a factor $n^{1/(\log\log n)^c}$, under the Exponential Time Hypothesis. Despite our feeling that (r,h)-GP is somewhat similar to DkS, we were unable to extend the techniques of [42] to this problem, or to prove its hardness of approximation via other techniques.

We overcome this difficulty as follows. First, we define a graph partitioning problem that is slightly more general than (r,h)-GP. The definition of this problem is somewhat technical and is deferred to Section 3. This problem is specifically designed so that the reduction to NDP-Grid still goes through, but it is somewhat easier to control its solutions and prove hardness of approximation for it. Furthermore, instead of employing a standard Karp-type reduction (where an instance of 3COL(5) is reduced to a single instance of NDP-Grid, while using the graph partitioning problem as a proxy), we employ a sequential Cook-type reduction. We assume for contradiction that an α-approximation algorithm $\mathcal{A}$ for NDP-Grid exists, where α is the hardness-of-approximation factor we are trying to prove. Our reduction is iterative. In every iteration j, we reduce the 3COL(5) instance to a collection $\mathcal{I}_j$ of instances of NDP-Grid, and apply the algorithm $\mathcal{A}$ to each of them. If the 3COL(5) instance is a YES-INSTANCE, then we are guaranteed that each resulting instance of NDP-Grid has a large solution, and so all solutions returned by

$\mathcal{A}$ are large. If the 3COL(5) instance is a NO-INSTANCE, then unfortunately it is still possible that the resulting instances of NDP-Grid will have large solutions. However, we can use these resulting solutions in order to further refine our reduction, and construct a new collection $\mathcal{I}_{j+1}$ of instances of NDP-Grid. While in the YES-INSTANCE case we will continue to obtain large solutions to all NDP-Grid instances that we construct, we can show that in the NO-INSTANCE case, in some iteration of the algorithm, we will fail to find such a large solution. Our reduction is crucially sequential, and we exploit the solutions returned by algorithm $\mathcal{A}$ in previous iterations in order to construct new instances of NDP-Grid for subsequent iterations. It is interesting whether these techniques may be helpful in obtaining new hardness-of-approximation results for DkS.

We note that our approach is completely different from the previous hardness-of-approximation proof of [19]. The proof in [19] proceeded by performing a reduction from 3SAT(5). Initially, a simple reduction from 3SAT(5) to the NDP problem on a subgraph of the grid graph is used in order to produce a constant hardness gap. The resulting instance of NDP is called a level-1 instance. The reduction then employs a boosting technique, that, in order to obtain a level-i instance, combines a number of level-$(i-1)$ instances with a single level-1 instance. The hardness gap grows by a constant factor from iteration to iteration, until the desired hardness-of-approximation bound is achieved. All source vertices in the constructed instances appear on the grid boundary, and a large number of vertices are carefully removed from the grid in order to create obstructions to routing, and to force the routing paths to behave in a prescribed way. The reduction itself is a Karp-type reduction, and eventually produces a single instance of NDP with a large gap between the YES-INSTANCE and NO-INSTANCE solutions.

Organization. We start with Preliminaries in Section 2, and introduce the new graph partitioning problems in Section 3. We introduce two main tools that we use in the hardness-of-approximation proof in Section 4. The tools are a reduction from 3COL to the graph partitioning problem, and a reduction from the graph partitioning problem to NDP-Grid. The hardness-of-approximation proof for NDP-Grid appears in Section 5, with the details of the reduction from the graph partitioning problem to NDP-Grid deferred to Section 6. Finally, we extend our hardness results to NDP-Wall and EDP-Wall in Section 7.

2 Preliminaries

All graphs in this paper are simple. We use standard graph-theoretic notation. Given a graph G and a subset $W \subseteq V(G)$ of its vertices, $E(W)$ denotes the set of all edges of G that have both their endpoints in W. Given a path P and a subset U of vertices of G, we say that P is *internally disjoint* from U if every vertex in $P \cap U$ is an endpoint of P. Similarly, we say that P is internally disjoint from a subgraph G' of G if P is internally disjoint from $V(G')$. Given a subset $\mathcal{M}' \subseteq \mathcal{M}$ of the demand pairs in G, we denote by $S(\mathcal{M}')$ and $T(\mathcal{M}')$ the sets of the source and the destination vertices of the demand pairs in $\mathcal{M}'$, respectively. We let $\mathcal{T}(\mathcal{M}') = S(\mathcal{M}') \cup T(\mathcal{M}')$ denote the set of all terminals participating as a source or a destination in $\mathcal{M}'$. Given a graph G and a vertex $v \in V(G)$, we denote by $d_G(v)$ the degree of v in G. All logarithms in this paper are to the base of 2.

Grid graphs. For a pair $h, \ell > 0$ of integers, we let $G^{h,\ell}$ denote the grid of height h and length ℓ. The set of its vertices is $V(G^{h,\ell}) = \{v(i,j) \mid 1 \le i \le h, 1 \le j \le \ell\}$, and the set of its edges is the union of two subsets: the set $E^H = \{(v(i,j), v(i,j+1)) \mid 1 \le i \le h, 1 \le j < \ell\}$ of horizontal edges and the set $E^V = \{(v(i,j), v(i+1,j)) \mid 1 \le i < h, 1 \le j \le \ell\}$ of vertical edges. The subgraph of $G^{h,\ell}$ induced by the edges of E^H consists of h paths, that we call the *rows* of the grid; for $1 \le i \le h$, the ith row R_i is the row containing the vertex $v(i,1)$. Similarly, the subgraph induced by the edges of E^V consists of ℓ paths that we call the *columns* of the grid, and for $1 \le j \le \ell$, the jth column W_j is the column containing $v(1,j)$. We think of the rows as ordered from top to bottom and the columns as ordered from left to right. Given a vertex $v = v(i,j)$ of the grid, we denote by $\mathrm{row}(v)$ and $\mathrm{col}(v)$ the row and the column of the grid, respectively, that contain v. We say that $G^{h,\ell}$ is a *square grid* if $h = \ell$. The *boundary of the grid* is $R_1 \cup R_h \cup W_1 \cup W_\ell$. We sometimes refer to R_1 and R_h as the top and the bottom boundary edges of the grid respectively, and to W_1 and W_ℓ as the left and the right boundary edges of the grid.

Given a subset $\mathcal{R}'$ of consecutive rows of G and a subset $\mathcal{W}'$ of consecutive columns of G, the *subgrid of G spanned by the rows in $\mathcal{R}'$ and the columns in $\mathcal{W}'$* is the subgraph of G induced by the set $\{v \mid \mathrm{row}(v) \in \mathcal{R}', \mathrm{col}(v) \in \mathcal{W}'\}$ of its vertices.

Given two vertices $u = v(i,j)$ and $u' = v(i',j')$ of a grid G, the shortest-path distance between them is denoted by $d(u,u')$. Given two vertex subsets $X, Y \subseteq V(G)$, the distance between them is $d(X,Y) = \min_{u \in X, u' \in Y} \{d(u,u')\}$. When H, H' are subgraphs of G, we use $d(H,H')$ to denote $d(V(H), V(H'))$.

Wall graphs. Let $G = G^{\ell,h}$ be a grid of length ℓ and height h. Assume that $\ell > 0$ is an even integer, and that $h > 0$. For every column W_j of the grid, let $e^j_1, \dots, e^j_{h-1}$ be the edges of W_j indexed in their top-to-bottom order. Let $E^*(G) \subseteq E(G)$ contain all edges e^j_z, where $z \ne j \mod 2$, and let $\hat{G}$ be the graph obtained from $G \setminus E^*(G)$, by deleting all degree-1 vertices from it. Graph $\hat{G}$ is called a *wall of length $\ell/2$ and height h* (see Figure 1). Consider the subgraph of $\hat{G}$ induced by all horizontal edges of the grid G that belong to $\hat{G}$. This graph is a collection of h node-disjoint paths, that we refer to as the *rows* of $\hat{G}$, and denote them by $R_1, \dots, R_h$ in this top-to-bottom order; notice that R_j is also the jth row of the grid G for all j. Graph $\hat{G}$ contains a unique collection $\mathcal{W}$ of $\ell/2$ node-disjoint paths that connect vertices of R_1 to vertices of R_h and are internally disjoint from R_1 and R_h. We refer to the paths in $\mathcal{W}$ as the *columns* of $\hat{G}$, and denote them by $W_1, \dots, W_{\ell/2}$ in this left-to-right order. Paths $W_1, W_{\ell/2}, R_1$ and R_h are called the left, right, top and bottom boundary edges of $\hat{G}$, respectively, and their union is the boundary of $\hat{G}$.

Semiregular bipartite graphs. We will use the following notion of semiregular bipartite graphs.

Definition 2.1. Let $G = (A,B,E)$ be a bipartite graph. We say that G is (d,d')-*semiregular*, for integers $d, d' > 0$ if the following hold:

- For every vertex $v \in A$, $d \le d_G(v) < 2d$;
- For every vertex $u \in B$, $d_G(u) < 2d'$; and
- $|E| \ge d'|B|/(4\log|B|)$ if $|B| > 1$, and $|E| = d'$ otherwise.

We say that G is *semiregular* if it is (d,d')-semiregular for any integers $d,d' > 0$.

Claim 2.2. *There is a polynomial-time algorithm, that, given any bipartite graph $G = (A,B,E)$ with $|A|,|B| > 1$, computes a semiregular subgraph $G' = (A',B',E')$ of G, with $A' \subseteq A$, $B' \subseteq B$, and $|E'| \geq \Omega(|E|/(\log|A|\log|B|))$.*

(We note that, if $|A| = 1$ or $|B| = 1$, then the graph is semiregular by definition.)

Proof. We transform the graph G into a semiregular graph G' in two steps. In the first step, we regularize the degrees of the vertices of B. We partition the vertices in B into $r = \lceil \log|A| \rceil + 1$ groups $Y_1,\ldots,Y_r$, as follows: group Y_i contains all vertices $u \in B$ with $2^i \leq d_G(u) < 2^{i+1}$. If a vertex $u \in B$ belongs to group Y_i, then we say that all edges that are incident to u also belong to group Y_i. Clearly, there is some index $1 \leq i \leq r$, such that at least $|E|/r = \Omega(|E|/\log|A|)$ edges of E belong to group Y_i. We let $d' = 2^i$, and we define a new bipartite graph $\hat{G}$. The set of vertices consists of the set A, and of a set $B' \subseteq B$, containing all vertices that belong to group Y_i. The set of edges, that we denote by $\hat{E}$, contains all edges of E that are incident to the vertices of B'. Notice that $|\hat{E}| \geq d'|B'|$.

In the second step, we similarly regularize the degrees of the vertices in A. We partition the vertices of A into $r' = \lceil \log|B'| \rceil + 1$ groups $X_1,\ldots,X_{r'}$, as follows: group X_i contains all vertices $v \in A$ with $2^j \leq d_{\hat{G}}(v) < 2^{j+1}$. If a vertex $v \in A$ belongs to group X_j, then we say that all edges that are incident to v in $\hat{G}$ also belong to group X_j. Clearly, there is some index $1 \leq j \leq r'$, such that at least

$$\frac{|\hat{E}|}{r'} \geq \frac{d'|B'|}{\lceil \log|B'| \rceil + 1} \geq \frac{d'B'}{4\log|B'|} \geq \Omega\left(\frac{|E|}{\log|A|\log|B|}\right)$$

edges of $\hat{E}$ belong to group X_j. We set $d = 2^j$. We now define the final bipartite graph $G' = (A',B',E')$. The set B' of vertices remains the same as before. The set A' of vertices contains every vertex of A that belongs to class X_j. The set E' of edges contains every edge of $\hat{E}$ that is incident to a vertex in A'. It is now immediate to verify that for every vertex $v \in A'$, $d \leq d_{G'}(v) < 2d$, and for every vertex $u \in B'$, $d_{G'}(u) < 2d'$. From the above discussion, $|E'| \geq \Omega(|E|/\log|A|\log|B|)$. Moreover, $|E'| \geq d'|B'|/(4\log|B'|)$ as required. ☐

The 3COL(5) problem. The starting point of our reduction is the 3COL(5) problem. In this problem, we are given a 5-regular graph $G = (V,E)$. Note that, if $n = |V|$ and $m = |E|$, then $m = 5n/2$. We are also given a set $\mathcal{C} = \{r,b,g\}$ of 3 colors. A *coloring* $\chi : V \to \mathcal{C}$ is an assignment of a color in $\mathcal{C}$ to every vertex in V. We say that an edge $e = (u,v)$ is *satisfied* by the coloring χif $\chi(u) \neq \chi(v)$. We say that the coloring χ is *valid* if it satisfies every edge. We say that G is a YES-INSTANCE if there is a valid coloring $\chi : V \to \mathcal{C}$. We say that it is a NO-INSTANCE with respect to some given parameter ε if for every coloring $\chi : V \to \mathcal{C}$, at most a $(1-\varepsilon)$-fraction of the edges are satisfied by χ. We use the following theorem of Feige et al. [25]:

Theorem 2.3 (Proposition 15 in [25])**.** *There is some constant ε, such that distinguishing between the* YES-INSTANCES *and the* NO-INSTANCES *(with respect to ε) of 3COL(5) is* NP*-hard.*

A two-prover protocol. We use the following two-prover protocol. The two provers are called an *edge-prover* and a *vertex-prover.* Given a 5-regular graph G, the verifier selects an edge $e = (u, v)$ of G uniformly at random, and then selects a random endpoint (say v) of this edge. It then sends edge e to the edge-prover and vertex v to the vertex-prover. The edge-prover must return an assignment of colors from $\mathcal{C}$ to u and v, such that the two colors are distinct; the vertex-prover must return an assignment of a color from $\mathcal{C}$ to v. The verifier accepts iff both provers assign the same color to v. Given a 2-prover game $\mathcal{G}$, its *value* is the maximum acceptance probability of the verifier over all possible strategies of the provers.

Notice that, if G is a YES-INSTANCE, then there is a strategy for both provers that guarantees acceptance with probability 1: the provers fix a valid coloring $\chi : V \to \mathcal{C}$ of G and respond to the queries according to this coloring.

We claim that if G is a NO-INSTANCE, then for any strategy of the two provers, the verifier accepts with probability at most $(1-\varepsilon/2)$. Note first that we can assume without loss of generality that the strategies of the provers are deterministic. Indeed, if the provers have a probability distribution over the answers to each query Q, then the edge-prover, given a query Q', can return an answer that maximizes the acceptance probability of the verifier under the random strategy of the vertex-prover. This defines a deterministic strategy for the edge-prover that does not decrease the acceptance probability of the verifier. The vertex-prover in turn, given any query Q, can return an answer that maximizes the acceptance probability of the verifier, under the new deterministic strategy of the edge-prover. The acceptance probability of this final deterministic strategy of the two provers is at least as high as that of the original randomized strategy. The deterministic strategy of the vertex-prover defines a coloring of the vertices of G. This coloring must dissatisfy at least εm edges. The probability that the verifier chooses one of these edge is at least ε. The response of the edge-prover on such an edge must differ from the response of the vertex-prover on at least one endpoint of the edge. The verifier chooses this endpoint with probability at least $1/2$, and so overall the verifier rejects with probability at least $\varepsilon/2$. Therefore, if G is a YES-INSTANCE, then the value of the corresponding game is 1, and if it is a NO-INSTANCE, then the value of the game is at most $(1-\varepsilon/2)$.

Parallel repetition. We perform ℓ rounds of parallel repetition of the above protocol, for some integer $\ell > 0$, that may depend on $n = |V(G)|$. Specifically, the verifier chooses a sequence $(e_1, \dots, e_\ell)$ of ℓ edges, where each edge e_i is selected independently uniformly at random from $E(G)$. For each chosen edge e_i, one of its endpoints v_i is then chosen independently at random. The verifier sends $(e_1, \dots, e_\ell)$ to the edge-prover, and $(v_1, \dots, v_\ell)$ to the vertex-prover. The edge-prover returns a coloring of both endpoints of each edge e_i. This coloring must satisfy the edge (so the two endpoints must be assigned different colors), but it need not be consistent across different edges. In other words, if two edges e_i and e_j share the same endpoint v, the edge-prover may assign different colors to each occurrence of v. The vertex-prover returns a coloring of the vertices in $(v_1, \dots, v_\ell)$. Again, if some vertex v_i repeats twice, the coloring of the two occurrences need not be consistent. The verifier accepts iff for each $1 \le i \le \ell$, the coloring of the vertex v_i returned by both provers is consistent. (No verification of consistency is performed across different i's. So, for example, if v_i is an endpoint of e_j for $i \neq j$, then it is possible that the two colorings do not agree and the verifier still accepts.) We say that a pair (A, A') of answers to the two queries $(e_1, \dots, e_\ell)$ and $(v_1, \dots, v_\ell)$ is *matching*, or *consistent*, if it causes the verifier to accept. We let $\mathcal{G}^\ell$ denote this 2-prover protocol with ℓ repetitions.

Theorem 2.4 (Parallel Repetition, [48, 29, 46])**.** *There is some constant $0 < \gamma' < 1$, such that for each 2-prover game $\tilde{\mathcal{G}}$, if the value of $\tilde{\mathcal{G}}$ is x, then the value of the game $\tilde{\mathcal{G}}^\ell$, obtained from $\ell > 0$ parallel repetitions of $\tilde{\mathcal{G}}$, is at most $x^{\gamma'\ell}$.*

Corollary 2.5. *There is some constant $0 < \gamma < 1$, such that, if G is a* YES-INSTANCE*, then $\mathcal{G}^\ell$ has value 1, and if G is a* NO-INSTANCE*, then $\mathcal{G}^\ell$ has value at most $2^{-\gamma\ell}$.*

We now summarize the parameters and introduce some basic notation.

- Let $\mathcal{Q}^E$ denote the set of all possible queries to the edge-prover, so each query is an ℓ-tuple of edges. Then $|\mathcal{Q}^E| = m^\ell = (5n/2)^\ell$. Each query has 6^ℓ possible answers—6 colorings per edge. The set of feasible answers is the same for each edge-query, and we denote it by $\mathcal{A}^E$.
- Let $\mathcal{Q}^V$ denote the set of all possible queries to the vertex-prover, so each query is an ℓ-tuple of vertices, and $|\mathcal{Q}^V| = n^\ell$. Each query has 3^ℓ feasible answers – 3 colorings per vertex. The set of feasible answers is the same for each vertex-query, and we denote it by $\mathcal{A}^V$.
- We think about the verifier as choosing a number of random bits, that determine the choices of the queries $Q \in \mathcal{Q}^E$ and $Q' \in \mathcal{Q}^V$ that it sends to the provers. We sometimes call each such random choice a "random string." The set of all such choices is denoted by $\mathcal{R}$, where for each $R \in \mathcal{R}$, we denote $R = (Q, Q')$, with $Q \in \mathcal{Q}^E$, $Q' \in \mathcal{Q}^V$—the two queries sent to the two provers when the verifier chooses R. Then $|\mathcal{R}| = (2m)^\ell = (5n)^\ell$, and each random string $R \in \mathcal{R}$ is chosen with the same probability.
- It is important to note that each query $Q = (e_1, \dots, e_\ell) \in \mathcal{Q}^E$ of the edge-prover participates in exactly 2^ℓ random strings (one random string for each choice of one endpoint per edge of $\{e_1, \dots, e_\ell\}$), while each query $Q' = (v_1, \dots, v_\ell)$ of the vertex-prover participates in exactly 5^ℓ random strings (one random string for each choice of an edge incident to each of $v_1, \dots, v_\ell$).

A function $f : \mathcal{Q}^E \cup \mathcal{Q}^V \to \mathcal{A}^E \cup \mathcal{A}^V$ is called a *global assignment of answers to queries* if for every query $Q \in \mathcal{Q}^E$ to the edge-prover, $f(Q) \in \mathcal{A}^E$, and for every query $Q' \in \mathcal{Q}^V$ to the vertex-prover, $f(Q') \in \mathcal{A}^V$. We say that f is a *perfect global assignment* if for every random string $R = (Q^E, Q^V)$, $(f(Q^E), f(Q^V))$ is a matching pair of answers. The following simple theorem, whose proof appears in Section A of the Appendix, shows that in the YES-INSTANCE case, there are many perfect global assignments, that neatly partition all answers to the edge-queries.

Theorem 2.6. *Assume that G is a* YES-INSTANCE*. Then there are 6^ℓ perfect global assignments $f_1, \dots, f_{6^\ell}$ of answers to queries, such that:*

- *for each query $Q \in \mathcal{Q}^E$ to the edge-prover, for each possible answer $A \in \mathcal{A}^E$, there is exactly one index $1 \le i \le 6^\ell$ with $f_i(Q) = A$; and*
- *for each query $Q' \in \mathcal{Q}^V$ to the vertex-prover, for each possible answer $A' \in \mathcal{A}^V$ to Q', there are exactly 2^ℓ indices $1 \le i \le 6^\ell$, for which $f_i(Q') = A'$.*

Constraint graph H. Given a 3COL(5) instance G with $|V(G)| = n$, and an integer $\ell > 0$, we associate a graph H, that we call the *constraint graph*, with it. For every query $Q \in \mathcal{Q}^E \cup \mathcal{Q}^V$, there is a vertex $v(Q)$ in H, while for each random string $R = (Q, Q')$, there is an edge $e(R) = (v(Q), v(Q'))$. Notice that H is a bipartite graph. We denote by U^E the set of its vertices corresponding to the edge-queries, and by U^V the set of its vertices corresponding to the vertex-queries. Recall that $|U^E| = (5n/2)^\ell$, $|U^V| = n^\ell$; the degree of every vertex in U^E is 2^ℓ; the degree of every vertex in U^V is 5^ℓ, and $|E(H)| = |\mathcal{R}| = (5n)^\ell$.

Assignment graphs $L(H')$. Assume now that we are given some subgraph $H' \subseteq H$ of the constraint graph. We build a bipartite graph $L(H')$ associated with it (this graph takes into account the answers to the queries; it may be convenient for now to think that $H' = H$, but later we will use smaller subgraphs of H). The vertices of $L(H')$ are partitioned into two subsets:

- For each edge-query $Q \in \mathcal{Q}^E$ with $v(Q) \in H'$, for each possible answer $A \in \mathcal{A}^E$ to Q, we introduce a vertex $v(Q,A)$. We denote by $S(Q)$ the set of these 6^ℓ vertices corresponding to Q, and we call them a *group representing* Q. We denote by $\hat{U}^E$ the resulting set of vertices:
$$\hat{U}^E = \left\{ v(Q,A) \mid (Q \in \mathcal{Q}^E \text{ and } v(Q) \in H'), A \in \mathcal{A}^E \right\} .$$
- For each vertex-query $Q' \in \mathcal{Q}^V$ with $v(Q') \in H'$, for each possible answer $A' \in \mathcal{A}^V$ to Q', we introduce 2^ℓ vertices $v_1(Q',A'), \dots, v_{2^\ell}(Q',A')$. We call all these vertices *the copies of answer A' to query Q'*. We denote by $S(Q')$ the set of all vertices corresponding to Q':
$$S(Q') = \left\{ v_i(Q',A') \mid A' \in \mathcal{A}^V, 1 \le i \le 2^\ell \right\} ,$$
so $|S(Q')| = 6^\ell$. We call $S(Q')$ *the group representing Q'*. We denote by $\hat{U}^V$ the resulting set of vertices:
$$\hat{U}^V = \left\{ v_i(Q',A') \mid (Q' \in \mathcal{Q}^V \text{ and } v(Q') \in H'), A' \in \mathcal{A}^V, 1 \le i \le 2^\ell \right\} .$$

The final set of vertices of $L(H')$ is $\hat{U}^E \cup \hat{U}^V$. We define the set of edges of $L(H')$ as follows. For each random string $R = (Q^E, Q^V)$ whose corresponding edge $e(R)$ belongs to H', for every answer $A \in \mathcal{A}^E$ to Q^E, let $A' \in \mathcal{A}^V$ be the unique answer to Q^V consistent with A. For each copy $v_i(Q^V, A')$ of answer A' to query Q^V, we add an edge $(v(Q^E,A), v_i(Q^V,A'))$. Let
$$E(R) = \left\{ (v(Q^E,A), v_i(Q^V,A')) \;\middle|\; A \in \mathcal{A}^E, A' \in \mathcal{A}^V,\ A \text{ and } A' \text{ are consistent answers to } R, 1 \le i \le 2^\ell \right\}$$
be the set of the resulting edges, so $|E(R)| = 6^\ell \cdot 2^\ell = 12^\ell$. We denote by $\hat{E}$ the set of all edges of $L(H')$—the union of the sets $E(R)$ for all random strings R with $e(R) \in H'$.

Recall that we have defined a partition of the set $\hat{U}^E$ of vertices into groups $S(Q)$—one group for each query $Q \in \mathcal{Q}^E$ with $v(Q) \in H'$. We denote this partition by $\mathcal{U}_1$. Similarly, we have defined a partition of $\hat{U}^V$ into groups, that we denote by $\mathcal{U}_2 = \left\{ S(Q') \mid Q' \in \mathcal{Q}^V \text{ and } v(Q') \in H' \right\}$. Recall that for each group $U \in \mathcal{U}_1 \cup \mathcal{U}_2$, $|U| = 6^\ell$.

Finally, we need to define bundles of edges in the graph $L(H')$. For every vertex $v \in \hat{U}^E \cup \hat{U}^V$, we define a partition $\mathcal{B}(v)$ of the set of all edges incident to v in $L(H')$ into bundles, as follows. Fix some group $U \in \mathcal{U}_1 \cup \mathcal{U}_2$ that we have defined. If there is at least one edge of $L(H')$ connecting v to the vertices of U, then we define a bundle containing all edges connecting v to the vertices of U, and add this bundle to $\mathcal{B}(v)$. Therefore, if $v \in S(Q)$, then for each random string R in which Q participates, with $e(R) \in H'$, we have defined one bundle of edges in $\mathcal{B}(v)$. For each vertex $v \in \hat{U}^E \cup \hat{U}^V$, the set of all edges incident to v is thus partitioned into a collection of bundles, that we denote by $\mathcal{B}(v)$, and we denote $\beta(v) = |\mathcal{B}(v)|$. Note that, if $v \in S(Q)$ for some query $Q \in \mathcal{Q}^E \cup \mathcal{Q}^V$, then $\beta(v)$ is exactly the degree of the vertex $v(Q)$ in graph H'. Note also that $\bigcup_{v \in V(H')} \mathcal{B}(v)$ does not define a partition of the edges of $\hat{E}$, as each such edge belongs to two bundles. However, each of $\bigcup_{v \in \hat{U}^E} \mathcal{B}(v)$ and $\bigcup_{v \in \hat{U}^V} \mathcal{B}(v)$ does define a partition of $\hat{E}$. It is easy to verify that every bundle that we have defined contains exactly 2^ℓ edges. We use this fact later.

3 The (r,h)-Graph Partitioning problem

We will use a graph partitioning problem as a proxy in order to reduce the 3COL(5) problem to NDP-Grid. The specific graph partitioning problem is somewhat complex. We first define a simpler variant of this problem, and then provide the intuition and the motivation for the more complex variant that we eventually use.

In the basic (r,h)-Graph Partitioning problem, that we denote by (r,h)-GP, we are given a bipartite graph $\tilde{G} = (V_1, V_2, E)$, and two integral parameters $h, r > 0$. A solution consists of a partition $(W_1, \ldots, W_r)$ of $V_1 \cup V_2$ into r subsets, and for each $1 \leq i \leq r$, a subset $E_i \subseteq E(W_i)$ of edges, such that $|E_i| \leq h$. The goal is to maximize $\sum_i |E_i|$.

One intuitive way to think about the (r,h)-GP problem is that we would like to partition the vertices of $\tilde{G}$ into r clusters, that are roughly balanced (in terms of the number of edges in each cluster). However, unlike the standard balanced partitioning problems, that attempt to minimize the number of edges connecting the different clusters, our goal is to maximize the total number of edges that remain in the clusters. We suspect that the (r,h)-GP problem is very hard to approximate; in particular it appears to be somewhat similar to the Densest k-Subgraph problem (DkS). Like in the DkS problem, we are looking for dense subgraphs of $\tilde{G}$ (the subgraphs $\tilde{G}[W_i]$), but unlike DkS, where we only need to find one such dense subgraph, we would like to partition all vertices of $\tilde{G}$ into a prescribed number of dense subgraphs. We can prove that NDP-Grid is at least as hard as (r,h)-GP (to within polylogarithmic factors; see below), but unfortunately we could not prove strong hardness-of-approximation results for (r,h)-GP. In particular, known hardness proofs for DkS do not seem to generalize to this problem. To overcome this difficulty, we define a slightly more general problem, and then use it as a proxy in our reduction. Before defining the more general problem, we start with intuition.

Intuition: Given a 3COL(5) instance G, we can construct the constraint graph H, and the assignment graph $L(H)$, as described in Section 2. We can then view $L(H)$ as an instance of (r,h)-GP, with $r = 6^\ell$ and $h = |\mathcal{R}|$. Assume that G is a YES-INSTANCE. Then we can use the perfect global assignments $f_1, \ldots, f_r$ of answers to the queries, given by Theorem 2.6, in order to partition the vertices of $L(H)$ into $r = 6^\ell$

clusters $W_1,\dots,W_r$, as follows. Fix some $1 \le i \le r$. For each query $Q \in \mathcal{Q}^E$ to the edge-prover, set W_i contains a single vertex $v(Q,A) \in S(Q)$, where $A = f_i(Q)$. For each query $Q' \in \mathcal{Q}^V$ to the vertex-prover, set W_i contains a single vertex $v_j(Q',A')$, where $A' = f_i(Q')$, and the indices j are chosen so that every vertex $v_j(Q',A')$ participates in exactly one cluster W_i. From the construction of the assignment graph $L(H)$ and the properties of the assignments f_i guaranteed by Theorem 2.6, we indeed obtain a partition $W_1,\dots,W_r$ of the vertices of $L(H)$. For each $1 \le i \le r$, we then set $E_i = E(W_i)$. Notice that for every query $Q \in \mathcal{Q}^E \cup \mathcal{Q}^V$, exactly one vertex of $S(Q)$ participates in each cluster W_i. Therefore, for each group $U \in \mathcal{U}_1 \cup \mathcal{U}_2$, each cluster W_i contains exactly one vertex from this group. It is easy to verify that for each $1 \le i \le r$, for each random string $R \in \mathcal{R}$, set E_i contains exactly one edge of $E(R)$, and so $|E_i| = |\mathcal{R}| = h$, and the solution value is $h \cdot r$. Unfortunately, in the NO-INSTANCE, we may still obtain a solution of a high value, as follows: instead of distributing, for each query $Q \in \mathcal{Q}^E \cup \mathcal{Q}^V$, the vertices of $S(Q)$ to different clusters W_i, we may put all vertices of $S(Q)$ into a single cluster. While in our intended solution to the (r,h)-GP problem instance each cluster can be interpreted as an assignment of answers to the queries, and the number of edges in each cluster is bounded by the number of random strings satisfied by this assignment, we may no longer use this interpretation with this new type of solutions.[3] Moreover, unlike in the YES-INSTANCE solutions, if we now consider some cluster W_i, and some random string $R \in \mathcal{R}$, we may add several edges of $E(R)$ to E_i, which will further allow us to accumulate a high solution value. One way to get around this problem is to impose additional restrictions on the feasible solutions to the (r,h)-GP problem, which are consistent with our YES-INSTANCE solution, and thereby obtain a more general (and hopefully more difficult) problem. But while doing so we still need to ensure that we can prove that NDP-Grid remains at least as hard as the newly defined problem. Recall the definition of bundles in graph $L(H)$. It is easy to verify that in our intended solution to the YES-INSTANCE, every bundle contributes at most one edge to the solution. This motivates our definition of a slight generalization of the (r,h)-GP problem, that we call (r,h)-Graph Partitioning with Bundles, or (r,h)-GPwB.

The input to (r,h)-GPwB problem is almost the same as before: we are given a bipartite graph $\tilde{G} = (V_1,V_2,E)$, and two integral parameters $h,r > 0$. Additionally, we are given a partition $\mathcal{U}_1$ of V_1 into groups, and a partition $\mathcal{U}_2$ of V_2 into groups, so that for each group $U \in \mathcal{U}_1 \cup \mathcal{U}_2$, $|U| = r$. Using these groups, we define bundles of edges as follows: for every vertex $v \in V_1$, for each group $U \in \mathcal{U}_2$, such that some edge of E connects v to a vertex of U, the set of all edges that connect v to the vertices of U defines a single bundle. Similarly, for every vertex $v \in V_2$, for each group $U \in \mathcal{U}_1$, such that some edge of E connects v to a vertex of U, all edges that connect v to the vertices of U define a bundle. We denote, for each vertex $v \in V_1 \cup V_2$, by $\mathcal{B}(v)$ the set of all bundles into which the edges incident to v are partitioned, and we denote by $\beta(v) = |\mathcal{B}(v)|$ the number of such bundles. We also denote by $\mathcal{B} = \bigcup_{v \in V_1 \cup V_2} \mathcal{B}(v)$—the set of all bundles. Note that as before, $\mathcal{B}$ is not a partition of E, but every edge of E belongs to exactly two bundles: one bundle in $\bigcup_{v \in V_1} \mathcal{B}(v)$, and one bundle in $\bigcup_{v \in V_2} \mathcal{B}(v)$. As before, we need to compute a partition $(W_1,\dots,W_r)$ of $V_1 \cup V_2$ into r subsets, and for each $1 \le i \le r$, select a subset $E_i \subseteq E(W_i)$ of edges, such that $|E_i| \le h$. But now there is an additional restriction: we require that for each $1 \le i \le r$, for every bundle $B \in \mathcal{B}$, E_i contains at most one edge $e \in B$. As before, the goal is to maximize $\sum_i |E_i|$.

[3] We note that a similar problem arises if one attempts to design naive hardness-of-approximation proofs for DkS.

Valid instances, regular instances, and perfect solutions. Given an instance

$$\mathcal{I} = (\tilde{G} = (V_1, V_2, E), \mathcal{U}_1, \mathcal{U}_2, h, r)$$

of (r,h)-GPwB, let $\beta^*(\mathcal{I}) = \sum_{v \in V_1} \beta(v)$. Note that for any solution to $\mathcal{I}$, the solution value must be bounded by $\beta^*(\mathcal{I})$, since for every vertex $v \in V_1$, for every bundle $B \in \mathcal{B}(v)$, at most one edge from the bundle may contribute to the solution value. In all instances of (r,h)-GPwB that we consider, we always set $h = \beta^*(\mathcal{I})/r$. Next, we define valid instances; they are defined so that the instances that we obtain when reducing from 3COL(5) are always valid, as we show later.

Definition 3.1. We say that instance $\mathcal{I}$ of (r,h)-GPwB is *valid* if $h = \beta^*(\mathcal{I})/r$ and $h \geq \max_{v \in V_1 \cup V_2} \{\beta(v)\}$.

Recall that for every group $U \in \mathcal{U}_1 \cup \mathcal{U}_2$, $|U| = r$. We now define perfect solutions to the (r,h)-GPwB problem. We will ensure that our intended solutions in the YES-INSTANCE are always perfect, as we show later.

Definition 3.2. We say that a solution $((W_1, \ldots, W_r), (E_1, \ldots, E_r))$ to a valid (r,h)-GPwB instance $\mathcal{I}$ is *perfect* iff:

- for each group $U \in \mathcal{U}_1 \cup \mathcal{U}_2$, exactly one vertex of U belongs to each cluster W_i; and
- for each $1 \leq i \leq r$, $|E_i| = h$.

Note that the value of a perfect solution to a valid instance $\mathcal{I}$ is $h \cdot r = \beta^*(\mathcal{I})$, and this is the largest value that any solution can achieve.

Lastly, we say that an instance $\mathcal{I} = (\tilde{G} = (V_1, V_2, E), \mathcal{U}_1, \mathcal{U}_2, h, r)$ of the (r,h)-GPwB problem is (d_1, d_2, b)-*regular* if the graph $\tilde{G}$ is (d_1, d_2)-semiregular, and for every vertex $v \in V_1 \cup V_2$, every bundle $B \in \mathcal{B}(v)$ has cardinality exactly b.

4 Tools for the hardness proof

In this section we provide two basic tools that will be used in the hardness-of-approximation proof: a reduction from the 3COL(5) problem to the (r,h)-GPwB problem, and a reduction from the (r,h)-GPwB problem to the NDP-Grid problem. We also establish some properties of these reductions that will be used later.

4.1 From 3COL(5) to (r,h)-GPwB

Suppose we are given an instance G of the 3COL(5) problem, and an integral parameter $\ell > 0$ (the number of repetitions). Consider the corresponding constraint graph H, and suppose we are given some connected subgraph $H' \subseteq H$. We define an instance $\mathcal{I}(H')$ of (r,h)-GPwB, as follows. First, we use Claim 2.2

in order to compute a semiregular subgraph $H'' \subseteq H'$, with $|E(H'')| \geq \Omega(|E(H')|/\log^2|E(H'|)$. In our reduction, we will work with the graph H'' instead of the graph H' from now on. We now define the instance $\mathcal{I}(H')$ of (r,h)-GPwB.

- The underlying graph is $L(H'') = (\hat{U}^E, \hat{U}^V, \hat{E})$;
- the parameters are $r = 6^\ell$ and $h = |E(H'')|$;
- the partition $\mathcal{U}_1$ of $\hat{U}^E$ is the same as before: the vertices of $\hat{U}^E$ are partitioned into groups $S(Q)$—one group for each query $Q \in \mathcal{Q}^E$ with $v(Q) \in V(H'')$. Similarly, the partition $\mathcal{U}_2$ of $\hat{U}^V$ into groups is also defined exactly as before, and contains, for each query $Q' \in \mathcal{Q}^V$ with $v(Q') \in V(H'')$, a group $S(Q')$. (Recall that for all $Q \in \mathcal{Q}^E \cup \mathcal{Q}^V$ with $v(Q) \in H'$, $|S(Q)| = 6^\ell$.)

Claim 4.1. *Let G be an instance of the 3COL(5) problem, $\ell > 0$ an integral parallel repetition parameter, and $H' \subseteq H$ a connected subgraph of the corresponding constraint graph. Consider the corresponding instance $\mathcal{I}(H')$ of* (r,h)-GPwB. *Then $\mathcal{I}(H')$ is a valid instance, it is a (d', d'', b)-regular instance for some $d', d'' > 0$ and $b = 2^\ell$, and moreover, if G is a* YES-INSTANCE, *then there is a perfect solution to $\mathcal{I}(H')$.*

Proof. We first verify that $\mathcal{I}(H')$ is a valid instance of (r,h)-GPwB. Recall that for a query $Q \in \mathcal{Q}^E$ to the edge-prover and an answer $A \in \mathcal{A}^E$, the number of bundles incident to vertex $v(Q,A)$ in $L(H')$ is exactly the degree of the vertex $v(Q)$ in graph H''. The total number of bundles incident to the vertices of $S(Q)$ is then the degree of $v(Q)$ in H'' times $|\mathcal{A}^E|$. Therefore, $\beta^*(\mathcal{I}) = \sum_{v(Q,A) \in \hat{U}^E} |\beta(v)| = |E(H'')| \cdot |\mathcal{A}^E| = h \cdot r$. It is now immediate to verify that $h = \beta^*(\mathcal{I})/r$. Similarly, for a vertex $v = v_j(Q', A') \in U^V$, the number of bundles incident to v is exactly the degree of $v(Q')$ in H''. Since $h = |E(H'')|$, we get that $h \geq \max_{v \in V_1 \cup V_2}\{\beta(v)\}$, and so $\mathcal{I}(H')$ is a valid instance.

Next, we show that $\mathcal{I}(H')$ is a (d', d'', b)-regular instance, for some $d', d'' > 0$ and $b = 2^\ell$. Denote $\mathcal{I}(H') = (\tilde{G} = (V_1, V_2, E), \mathcal{U}_1, \mathcal{U}_2, h, r)$. Recall that we have defined a graph $H'' \subseteq H'$, that is (d_1, d_2)-semiregular, for some $d_1, d_2 > 0$. Recall also that, from the discussion at the end of Section 2, for every vertex $v \in V_1 \cup V_2$, every bundle $B \in \mathcal{B}(v)$ contains exactly 2^ℓ edges. Moreover, if $v \in S(Q)$ for any query Q, then the number of its bundles $\beta(v)$ is exactly the degree of the vertex $v(Q)$ in graph H''. Therefore, $d_{\tilde{G}}(v) = 2^\ell \cdot d_{H''}(v(Q))$. It follows that for every vertex $v \in V_1$, $d_1 \cdot 2^\ell \leq d_{\tilde{G}}(v) < 2d_1 \cdot 2^\ell$, and for every vertex $v' \in V_2$, $d_{\tilde{G}}(v') < 2d_2 \cdot 2^\ell$. Moreover, $|E(\tilde{G})| = 12^\ell \cdot |E(H'')|$.

Recall that $V(H') = U^E \cup U^V$, where the vertices of U^E represent the edge-queries, and the vertices of U^V represent the vertex-queries. Since, from the definition of semiregular graphs, we are guaranteed that $|E(H'')| \geq d_2|U^V \cap V(H'')|/(4\log|U^V \cap V(H'')|)$, and $|V_2| = 6^\ell \cdot |U^V \cap V(H'')|$, we get that:

$$\begin{aligned}|E(\tilde{G})| = 12^\ell \cdot |E(H'')| &\geq \frac{12^\ell \cdot d_2|U^V \cap V(H'')|}{4\log|U^V \cap V(H'')|} \\ &\geq \frac{(d_2 \cdot 2^\ell) \cdot |V_2|}{4\log|U^V \cap V(H'')|} \geq \frac{(d_2 \cdot 2^\ell) \cdot |V_2|}{4\log|V_2|}.\end{aligned}$$

We conclude that instance $\mathcal{I}(H')$ is (d', d'', b)-regular, for $d' = 2^\ell \cdot d_1$, $d'' = 2^\ell \cdot d_2$, and $b = 2^\ell$.

Assume now that G is a YES-INSTANCE. We define a perfect solution $((W_1,\dots,W_r),(E_1,\dots,E_r))$ to instance $\mathcal{I}(H')$ of the (r,h)-GPwB problem. Let $\{f_1,f_2,\dots,f_{6^\ell}\}$ be the collection of perfect global assignments of answers to the queries, given by Theorem 2.6. Recall that $\mathcal{U}_1=\left\{S(Q)\mid Q\in\mathcal{Q}^E, v(Q)\in H''\right\}$ and $\mathcal{U}_2=\left\{S(Q')\mid Q'\in\mathcal{Q}^V, v(Q')\in H''\right\}$, where each group in $\mathcal{U}_1\cup\mathcal{U}_2$ has cardinality $r=6^\ell$. We now fix some $1\le i\le r$, and define the set W_i of vertices. For each query $Q\in\mathcal{Q}^E$ to the edge-prover with $v(Q)\in V(H'')$, if $A=f_i(Q)$, then we add the vertex $v(Q,A)$ to W_i. For each query $Q'\in\mathcal{Q}^V$ to the vertex-prover with $v(Q')\in V(H'')$, if $A'=f_i(Q')$, then we select some index $1\le j\le 2^\ell$, and add the vertex $v_j(Q',A')$ to W_i. The indices j are chosen so that every vertex $v_j(Q',A')$ participates in at most one cluster W_i. From the construction of the graph $L(H'')$ and the properties of the assignments f_i guaranteed by Theorem 2.6, it is easy to verify that $W_1,\dots,W_r$ partition the vertices of $L(H'')$, and moreover, for each group $S(Q)\in\mathcal{U}_1\cup\mathcal{U}_2$, each set W_i contains exactly one vertex of $S(Q)$.

Finally, for each $1\le i\le r$, we set $E_i=E(W_i)$. We claim that for each bundle $B\in\mathcal{B}$, set E_i may contain at most one edge of B. Indeed, let $v\in W_i$ be some vertex, let $U\in\mathcal{U}_1\cup\mathcal{U}_2$ be some group, and let B be the bundle containing all edges that connect v to the vertices of U. Since W_i contains exactly one vertex of U, at most one edge of B may belong to E_i.

It now remains to show that $|E_i|=h$ for all i. Fix some $1\le i\le r$. It is easy to verify that for each random string $R=(Q,Q')$ with $e(R)\in H''$, set W_i contains a pair of vertices $v(Q,A)$, $v_j(Q',A')$, where A and A' are matching answers to Q and Q' respectively, and so the corresponding edge connecting this pair of vertices in $L(H'')$ belongs to E_i. Therefore, $|E_i|=|E(H'')|=h$. □

Claim 4.1 shows that, if an instance G of the 3COL(5) problem is a YES-INSTANCE, then the corresponding instance $\mathcal{I}(H)$ of (r,h)-GPwB has a solution of a high value (a perfect solution). Ideally, we would like to show that, if G is a NO-INSTANCE, then any solution to the corresponding (r,h)-GPwB instance has a low value. Unfortunately, we cannot do this directly, and this is the reason that our hardness-of-approximation proof, provided in Section 5, employs a Cook reduction and not a Karp reduction. As we will show in Section 5, if instance G of 3COL(5) is a NO-INSTANCE, but the corresponding instance of (r,h)-GPwB has a high solution value, then we can exploit the solution to the (r,h)-GPwB problem instance in order to compute a partition of the constraint graph H into smaller subgraphs, and then employ the same reduction on each one of the resulting subgraphs; see Section 5 for more details.

4.2 From (r,h)-GPwB to NDP

The following definition will be useful for us later, when we extend our results to NDP and EDP on wall graphs.

Definition 4.2. Let $\mathcal{P}$ be a set of paths in a grid $\hat{G}$. We say that $\mathcal{P}$ is a *spaced-out* set if for each pair $P,P'\in\mathcal{P}$ of paths, $d(V(P),V(P'))\ge 2$, and all paths in $\mathcal{P}$ are internally disjoint from the boundaries of the grid $\hat{G}$.

Note that if $\mathcal{P}$ is a set of paths that is spaced-out, then all paths in $\mathcal{P}$ are mutually node-disjoint. The following theorem is central to our hardness-of-approximation proof.

Theorem 4.3. *There is a constant $c^* > 0$, and there is an algorithm, that, given a valid and a (d,d',b)-regular instance $\mathcal{I} = (\tilde{G}, \mathcal{U}_1, \mathcal{U}_2, h, r)$ of* (r,h)-GPwB*, for some $d,d',b > 0$, with $|V(\tilde{G})| = N$ and $|E(\tilde{G})| = M$, in time* $\mathrm{poly}(NM)$ *constructs an instance $\hat{\mathcal{I}} = (\hat{G}, \mathcal{M})$ of* NDP-Grid *with $|V(\hat{G})| = O(M^4 \log^2 M)$, such that the following hold:*

- *if $\mathcal{I}$ has a perfect solution (of value $\beta^* = \beta^*(\mathcal{I})$), then instance $\hat{\mathcal{I}}$ has a solution $\mathcal{P}$ that routes at least $\frac{\beta^*}{c^* \log^2 M}$ demand pairs, such that the paths in $\mathcal{P}$ are spaced-out; and*
- *there is an algorithm with running time* $\mathrm{poly}(NM)$*, that, given a solution $\mathcal{P}^*$ to the* NDP-Grid *problem instance $\hat{\mathcal{I}}$, constructs a solution to the* (r,h)-GPwB *instance $\mathcal{I}$, of value at least $\frac{|\mathcal{P}^*|}{c^* \cdot \log^3 M}$.*

We note that the theorem is slightly stronger than what is needed in order to prove hardness of approximation of NDP-Grid: if $\mathcal{I}$ has a perfect solution, then it is sufficient to ensure that the set $\mathcal{P}$ of paths in the corresponding NDP-Grid instance $\hat{\mathcal{I}}$ is node-disjoint. But we will use the stronger guarantee that it is spaced-out when extending our results to NDP and EDP in wall graphs. Note that in the second assertion we are only guaranteed that the paths in $\mathcal{P}^*$ are node-disjoint. The proof of the theorem is somewhat technical and is deferred to Section 6.

4.3 Combining the tools for the YES-INSTANCE

Assume now that we are given an instance G of 3COL(5), an integral parallel repetitions parameter $\ell > 0$, and a connected subgraph $H' \subseteq H$ of the corresponding constraint graph. Recall that we have constructed a corresponding instance $\mathcal{I}(H')$ of (r,h)-GPwB that is valid and (d,d',b)-regular for some $d,d',b > 0$. We can then use Theorem 4.3 to construct an instance of NDP-Grid, that we denote by $\hat{\mathcal{I}}(H')$. Note that $|E(L(H''))| \leq 2^{O(\ell)} \cdot |E(H)| \leq n^{O(\ell)}$. Let $\hat{c}$ be a constant, such that $|E(L(H''))| \leq n^{\hat{c}\ell}$ for any connected subgraph $H' \subseteq H$; we can assume w.l.o.g. that $\hat{c} > 1$. We can also assume w.l.o.g. that $c^* \geq 1$, where c^* is the constant from Theorem 4.3, and we assume that the algorithm from Claim 2.2 returns a subgraph $G' = (A', B', E')$ of G with $|E'| \geq |E(G)|/(c^* \log^2 |E(G)|)$, if G is a connected graph. Lastly, we denote $c_{\mathrm{YI}} = (\hat{c} \cdot c^*)^4$. We obtain the following immediate corollary of Theorem 4.3:

Corollary 4.4. *Suppose we are given 3COL(5) instance G that is a* YES-INSTANCE*, an integral parallel repetition parameter $\ell > 0$, and a connected subgraph $H' \subseteq H$ of the corresponding constraint graph. Then instance $\hat{\mathcal{I}}(H')$ of* NDP-Grid *has a solution of value at least*

$$\frac{|E(H')| \cdot 6^\ell}{c_{\mathrm{YI}} \ell^4 \log^4 n},$$

where $n = |V(G)|$.

Proof. From Claim 4.1, instance $\mathcal{I}(H') = (L(H''), \mathcal{U}_1, \mathcal{U}_2, r, h)$ of (r,h)-GPwB is a valid instance, it is a (d', d'', b)-regular instance for some $d', d'', b > 0$, and it has a perfect solution, whose value is $\beta^* = \beta^*(\mathcal{I}(H'')) = h \cdot r = |E(H'')| \cdot 6^\ell$. Recall that H'' is a semiregularized graph, that was obtained from

H' by applying Claim 2.2 to it, so $|E(H'')| \geq |E(H')|/(c^* \log^2 |E(H')|)$. From Theorem 4.3, instance $\hat{\mathcal{I}}(H')$ of NDP-Grid has a solution of value at least

$$\frac{|E(H'')| \cdot 6^\ell}{c^* \log^2 |E(H'')|} \geq \frac{|E(H')| \cdot 6^\ell}{(c^*)^2 \log^4 M},$$

where $M = |E(L(H''))|$. Since $\log M \leq \hat{c}\ell \log n$, the corollary follows. □

As already mentioned before, if G is a NO-INSTANCE, it is still possible that the corresponding instance of (r,h)-GPwB, and hence the resulting instance of NDP-Grid, will have a high solution value. We get around this problem by employing a Cook reduction, as shown in Section 5.

5 The hardness proof

Let G be an input instance of 3COL(5). Recall that γ is the absolute constant from the Parallel Repetition Theorem (Corollary 2.5). We will set the value of the parallel repetition parameter ℓ later, ensuring that $\ell > \log^2 n$, where $n = |V(G)|$. Let $\alpha^* = 2^{\Theta(\ell/\log n)}$ be the hardness-of-approximation factor that we are trying to achieve.

Given the tools developed in the previous sections, a standard way to prove hardness of NDP-Grid would work as follows. Given an instance G of 3COL(5) and the chosen parameter ℓ, construct the corresponding graph H (the constraint graph), together with the assignment graph $L(H)$. We then construct an instance $\mathcal{I}(H)$ of (r,h)-GPwB as described in the previous section, and convert it into an instance $\hat{\mathcal{I}}(H)$ of NDP-Grid.

Note that, if G is a YES-INSTANCE, then from Corollary 4.4, there is a solution to $\hat{\mathcal{I}}(H)$ of value at least $\frac{|\mathcal{R}| \cdot 6^\ell}{c_{\mathrm{YI}} \ell^4 \log^4 n}$. Assume now that G is a NO-INSTANCE. If we could show that any solution to the corresponding (r,h)-GPwB instance $\mathcal{I}(H)$ has value less than $\frac{|\mathcal{R}| \cdot 6^\ell}{c_{\mathrm{YI}}^2 \cdot \alpha^* \ell^7 \log^7 n}$, we would be done. Indeed, in such a case, from Theorem 4.3, every solution to the NDP-Grid instance $\hat{\mathcal{I}}(H)$ routes fewer than $\frac{|\mathcal{R}| \cdot 6^\ell}{c_{\mathrm{YI}} \alpha^* \ell^4 \log^4 n}$ demand pairs. If we assume for contradiction that an α^*-approximation algorithm exists for NDP-Grid, then, if G is a YES-INSTANCE, the algorithm would have to return a solution to $\hat{\mathcal{I}}(H)$ routing at least $\frac{|\mathcal{R}| \cdot 6^\ell}{c_{\mathrm{YI}} \alpha^* \ell^4 \log^4 n}$ demand pairs, while, if G is a NO-INSTANCE, no such solution would exist. Therefore, we could use the α^*-approximation algorithm for NDP-Grid to distinguish between the YES-INSTANCES and the NO-INSTANCES of 3COL(5).

Unfortunately, we are unable to prove this directly. For simplicity, assume that H is a semiregular graph. Our intended solution to the (r,h)-GPwB instance $\mathcal{I}(H)$, defined over the graph $L(H)$, for each query $Q \in \mathcal{Q}^E \cup \mathcal{Q}^V$, places every vertex of $S(Q)$ into a distinct cluster. Any such solution will indeed have a low value in the NO-INSTANCE. But a cheating solution may place many vertices from the same set $S(Q)$ into some cluster W_j. Such a solution may end up having a high value, but it may not translate into a good strategy for the two provers, that satisfies a large fraction of the random strings. In an extreme case, for each query Q, we may place all vertices of $S(Q)$ into a single cluster W_j. The main idea in our reduction is to overcome this difficulty by noting that such a cheating solution can be used to compute a partition of the constraint graph H. The graph is partitioned into as many as 6^ℓ pieces, each of which is significantly smaller than the original graph H. At the same time, a large fraction of the edges of H will survive the partitioning procedure. Intuitively, if we now restrict ourselves to only those random strings $R \in \mathcal{R}$, whose corresponding edges have survived the partitioning procedure, then the problem does not become significantly easier, and we can recursively apply the same argument to the resulting subgraphs of H. We make a significant progress in each such iteration, since the sizes of the resulting subgraphs of H decrease very fast. The main tool that allows us to execute this plan is the following theorem.

Theorem 5.1. *Suppose we are given an instance G of the 3COL(5) problem with $|V(G)| = n$, and an integral parallel repetition parameter $\ell > \log^2 n$, together with some connected subgraph $H' \subseteq H$ of the corresponding constraint graph H. Consider the corresponding instance $\mathcal{I}(H')$ of* (r,h)-GPwB*, and assume that we are given a solution to this instance of value at least $|E(H')| \cdot 6^\ell/\alpha$, where $\alpha = c_{\text{YI}}^2 \cdot \alpha^* \cdot \ell^7 \log^7 n$. Then there is an algorithm, whose running time is $O\left(n^{O(\ell)}\right)$, that returns one of the following:*

- *Either a strategy for the two provers that satisfies more than a $2^{-\gamma\ell/2}$-fraction of the constraints $R \in \mathcal{R}$ with $e(R) \in E(H')$; or*
- *a collection $\mathcal{H}$ of disjoint connected subgraphs of H', such that for each $\tilde{H} \in \mathcal{H}$, $|E(\tilde{H})| \leq |E(H')|/2^{\gamma\ell/16}$, and $\sum_{\tilde{H}\in\mathcal{H}} |E(\tilde{H})| \geq c'|E(H')|/(\ell^2\alpha^2)$, for some universal constant c'.*

We postpone the proof of the theorem to the following subsection, after we complete the hardness proof for NDP-Grid. We assume for contradiction that we are given a factor-α^* approximation algorithm $\mathcal{A}$ for NDP-Grid (recall that $\alpha^* = 2^{\Theta(\ell/\log n)}$). We will use this algorithm to distinguish between the YES-INSTANCES and the NO-INSTANCES of 3COL(5) with $|V(G)| = n$. Suppose we are given an instance G of 3COL(5).

For an integral parallel repetition parameter $\ell > \log^2 n$, let H be the constraint graph corresponding to G and ℓ. We next show an algorithm, that uses $\mathcal{A}$ as a subroutine, in order to determine whether G is a YES-INSTANCE or a NO-INSTANCE. The running time of the algorithm is $n^{O(\ell)}$.

Throughout the algorithm, we maintain a collection $\mathcal{H}$ of disjoint connected subgraphs of H, that we sometimes call *clusters*. Set $\mathcal{H}$ is in turn partitioned into two subsets: set $\mathcal{H}_1$ of *active* clusters, and set $\mathcal{H}_2$ of *inactive* clusters. Consider now some inactive cluster $H' \in \mathcal{H}_2$. This cluster defines a 2-prover game $\mathcal{G}(H')$, where the queries to the two provers are $\left\{Q^E \in \mathcal{Q}^E \mid v(Q^E) \in V(H')\right\}$, and $\left\{Q^V \in \mathcal{Q}^V \mid v(Q^V) \in V(H')\right\}$ respectively, and the constraints of the verifier are:

$$\mathcal{R}(H') = \left\{R \in \mathcal{R} \mid e(R) \in E(H')\right\}.$$

For each inactive cluster $H' \in \mathcal{H}_2$, we will store a strategy of the two provers for game $\mathcal{G}(H')$, that satisfies at least a $2^{-\gamma\ell/2}$-fraction of the constraints in $\mathcal{R}(H')$.

At the beginning, the clusters in $\mathcal{H}$ are the connected components of H, and every cluster is active. The algorithm is executed while $\mathcal{H}_1 \neq \emptyset$, and its execution is partitioned into phases. In every phase, we process each of the clusters that belongs to $\mathcal{H}_1$ at the beginning of the phase. Each phase is then in turn is partitioned into iterations, where in every iteration we process a distinct active cluster $H' \in \mathcal{H}_1$. We describe an iteration when an active cluster $H' \in \mathcal{H}_1$ is processed in Figure 2 (see also the flowchart in Figure 3).

ITERATION FOR PROCESSING A CLUSTER $H' \in \mathcal{H}_1$

1. Construct an instance $\mathcal{I}(H')$ of (r,h)-GPwB.
2. Use Theorem 4.3 to construct an instance $\hat{\mathcal{I}}(H')$ of the NDP-Grid problem.
3. Run the α^*-approximation algorithm $\mathcal{A}$ on this instance. If the resulting solution routes fewer than $\frac{|E(H')|6^\ell}{c_{\mathrm{YI}}\alpha^*\ell^4\log^4 n}$ demand pairs, halt and return "G is a NO-INSTANCE."
4. Otherwise, consider the returned solution routing at least $\frac{|E(H')|6^\ell}{c_{\mathrm{YI}}\alpha^*\ell^4\log^4 n}$ demand pairs. Denote $|E(L(H''))| = M$, and recall that $M \leq n^{\hat{c}\ell}$. Use Theorem 4.3 to compute a solution $((W_1,\dots,W_r),(E_1,\dots,E_r))$ to the instance $\mathcal{I}(H')$ of the (r,h)-GPwB problem, of value at least:
$$\frac{|E(H')|\cdot 6^\ell}{(c_{\mathrm{YI}}\alpha^*\ell^4\log^4 n)(c^*\log^3 M)} \geq \frac{|E(H')|\cdot 6^\ell}{c_{\mathrm{YI}}c^*\hat{c}^3\alpha^*\ell^7\log^7 n} \geq \frac{|E(H')|\cdot 6^\ell}{c_{\mathrm{YI}}^2\alpha^*\ell^7\log^7 n} = \frac{|E(H')|\cdot 6^\ell}{\alpha}.$$
5. Apply the algorithm from Theorem 5.1 to this solution.
   (a) If the outcome is a strategy for the provers satisfying more than a $2^{-\gamma\ell/2}$-fraction of constraints $R \in \mathcal{R}$ with $e(R) \in E(H')$, then declare cluster H' inactive and move it from $\mathcal{H}_1$ to $\mathcal{H}_2$. Store the resulting strategy of the provers.
   (b) Otherwise, let $\tilde{\mathcal{H}}$ be the collection of subgraphs of H' returned by the algorithm. Remove H' from $\mathcal{H}_1$ and add all graphs of $\tilde{\mathcal{H}}$ to $\mathcal{H}_1$.

Figure 2: Description of an iteration

If the algorithm terminates with $\mathcal{H}$ containing only inactive clusters, then we return "G is a YES-INSTANCE."

Correctness. We establish the correctness of the algorithm in the following two lemmas.

Lemma 5.2. *If G is a* YES-INSTANCE*, then the algorithm always returns "G is a* YES-INSTANCE*."*

Proof. Consider an iteration of the algorithm when an active cluster H' is processed. Notice that the algorithm may only determine that G is a NO-INSTANCE in Step (3).

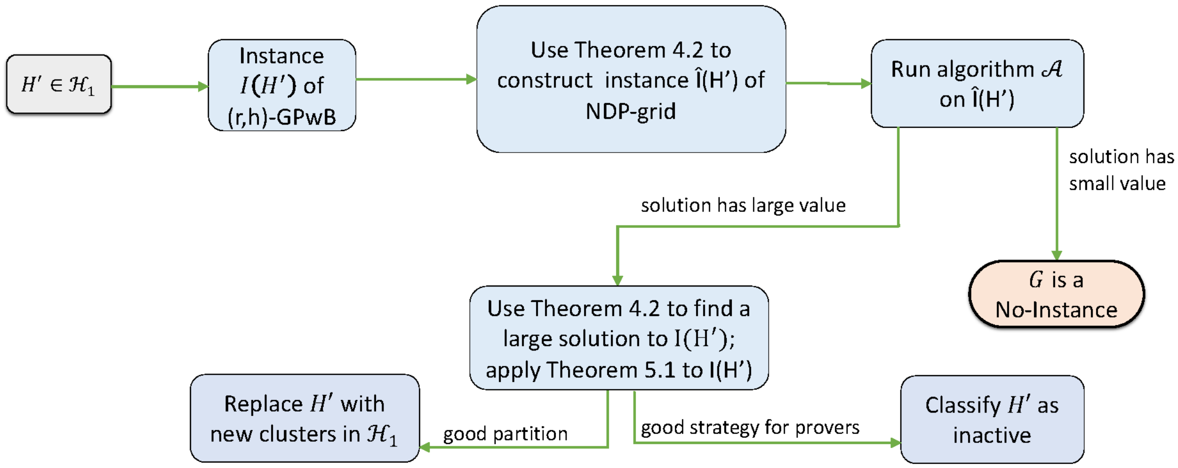

Figure 3: Flowchart for the execution of an iteration

From Corollary 4.4, an instance $\hat{\mathcal{I}}(H')$ of NDP-Grid is guaranteed to have a solution of value at least

$$\psi := \frac{|E(H')| \cdot 6^{\ell}}{c_{\mathrm{YI}} \ell^4 \log^4 n},$$

and our α^*-approximation algorithm to NDP-Grid must then return a solution of value at least ψ/α^*. Therefore, our algorithm will never return "G is a NO-INSTANCE." □

Lemma 5.3. *If G is a* NO-INSTANCE, *then the algorithm always returns "G is a* NO-INSTANCE."

Proof. From Corollary 2.5, it is enough to show that, whenever the algorithm classifies G as a YES-INSTANCE, there is a strategy for the two provers, that satisfies more than a $2^{-\gamma\ell}$-fraction of the constraints in $\mathcal{R}$.

Note that the original graph H has at most $n^{\hat{c}\ell}$ edges. In every phase, the number of edges in each active graph decreases by a factor of at least $2^{\gamma\ell/16}$. Therefore, the number of phases is bounded by $O(\log n)$. If the algorithm classifies G as a YES-INSTANCE, then it must terminate when no active clusters remain. In every phase, the number of edges in $\bigcup_{H' \in \mathcal{H}} E(H')$ goes down by at most a factor $\ell^2\alpha^2/c'$. Therefore, at the end of the algorithm:

$$\sum_{H' \in \mathcal{H}_2} |E(H')| \geq \frac{|\mathcal{R}|}{(\ell^2\alpha^2/c')^{O(\log n)}} = \frac{|\mathcal{R}|}{(\ell^{16} \cdot (\alpha^*)^2 \cdot \log^{14} n)^{O(\log n)}} = \frac{|\mathcal{R}|}{(\alpha^*)^{O(\log n)}}.$$

By appropriately setting $\alpha^* = 2^{\Theta(\ell/\log n)}$, we will ensure that the number of edges remaining in the inactive clusters $H' \in \mathcal{H}_2$ is at least $|\mathcal{R}|/2^{\gamma\ell/4}$. Each such edge corresponds to a distinct random string $R \in \mathcal{R}$. Recall that for each inactive cluster H', there is a strategy for the provers in the corresponding game $\mathcal{G}(H')$ that satisfies at least $|E(H')|/2^{\gamma\ell/2}$ of its constraints. Taking the union of all these strategies, we can satisfy more than $|\mathcal{R}|/2^{\gamma\ell}$ constraints of $\mathcal{R}$, contradicting the fact that G is a NO-INSTANCE. $\square$

Running time and the hardness factor. It is easy to see that the algorithm has at most $n^{O(\ell)}$ iterations, where in every iteration it processes a distinct active cluster $H' \subseteq H$. The corresponding graph $L(H'')$ has at most $n^{O(\ell)}$ edges, and so the resulting instance of NDP-Grid contains at most $n^{O(\ell)}$ vertices. Therefore, the overall running time of the algorithm is $n^{O(\ell)}$. From the above analysis, the algorithm always correctly classifies G as a YES-INSTANCE or as a NO-INSTANCE. The hardness factor that we obtain is $\alpha^* = 2^{\Theta(\ell/\log n)}$, while we only apply our approximation algorithm to instances of NDP-Grid containing at most $N = n^{O(\ell)}$ vertices.

Setting $\ell = \log^p n$ for a large enough integer p, we obtain $\alpha^* = 2^{\Theta((\log N)^{1-2/(p+1)})}$, giving us a $2^{(\log n)^{(1-\varepsilon)}}$-hardness of approximation for NDP-Grid for any constant ε, assuming $\mathsf{NP} \not\subseteq \mathsf{DTIME}(n^{\operatorname{poly}\log n})$.

Setting $\ell = n^\delta$ for some constant δ, we get that $N = 2^{O(n^\delta \log n)}$ and $\alpha^* = 2^{\Theta(n^\delta/\log n)}$, giving us a $n^{\Omega(1/(\log\log n)^2)}$-hardness of approximation for NDP-Grid, assuming that $\mathsf{NP} \not\subseteq \mathsf{DTIME}(2^{n^\delta})$ for some constant $\delta > 0$.

Note that the approximation factor α^* is in fact eventually expressed as a function of the size of the NDP-Grid instance, and that we apply the approximation algorithm to instances of NDP-Grid of various sizes. However, the size of each resulting instance of NDP-Grid is bounded by N, and so the approximation factor achieved for each such instance should be at most α^* (as defined with respect to N).

5.1 Proof of Theorem 5.1

Recall that each edge of graph H' corresponds to some constraint $R \in \mathcal{R}$. Let $\mathcal{R}' \subseteq \mathcal{R}$ be the set of all constraints R with $e(R) \in E(H')$. Let $\mathcal{R}'' \subseteq \mathcal{R}'$ be the set of all constraints R with $e(R) \in E(H'')$. Denote the solution to the (r,h)-GPwB instance $\mathcal{I}(H')$ by $((W_1,\ldots,W_r),(E_1,\ldots,E_r))$, and let $E' = \bigcup_{i=1}^r E_i$. Recall that for each random string $R \in \mathcal{R}''$, there is a set $E(R)$ of 12^ℓ edges in graph $L(H'')$ representing R. Due to the way these edges are partitioned into bundles, at most 6^ℓ edges of $E(R)$ may belong to E'. We say that a random string $R \in \mathcal{R}''$ is *good* if E' contains at least $6^\ell/(2\alpha)$ edges of $E(R)$, and we say that it is bad otherwise.

Observation 5.4. At least $|\mathcal{R}'|/(2\alpha)$ random strings of $\mathcal{R}''$ are good.

Proof. A good random string contributes at most 6^ℓ edges to E', while a bad random string contributes at most $6^\ell/(2\alpha)$ edges. If the number of good random strings is less than $|\mathcal{R}'|/(2\alpha)$, then a simple accounting shows that $|E'| < |\mathcal{R}'| \cdot 6^\ell/\alpha = |E(H')| \cdot 6^\ell/\alpha$, a contradiction. $\square$

Consider some random string $R \in \mathcal{R}''$, and assume that $R = (Q^E, Q^V)$. We denote by $E'(R) = E(R) \cap E'$.

Intuitively, say that a cluster W_i is a *terrible cluster* for R if the number of edges of $E(R)$ that lie in E_i is much smaller than $|Q^E \cap W_i|$ or $|Q^V \cap W_i|$. We now give a formal definition of a terrible cluster.

Definition 5.5. Given a random string $R \in \mathcal{R}''$ and an index $1 \le i \le 6^\ell$, we say that a cluster W_i is a *terrible cluster* for R, if:

- either $|E'(R) \cap E_i| < |W_i \cap S(Q^V)|/(8\alpha)$; or
- $|E'(R) \cap E_i| < |W_i \cap S(Q^E)|/(8\alpha)$.

We say that an edge $e \in E'(R)$ is a *terrible edge* if it belongs to the set E_i, where W_i is a terrible cluster for R.

Observation 5.6. For each good random string $R \in \mathcal{R}''$, at most $6^\ell/(4\alpha)$ edges of $E'(R)$ are terrible.

Proof. Assume for contradiction that more than $6^\ell/(4\alpha)$ edges of $E'(R)$ are terrible. Denote $R = (Q^E, Q^V)$. Consider some such terrible edge $e \in E'(R)$, and assume that $e \in E_i$ for some cluster W_i, that is terrible for R. We say that e is a *type*-1 *terrible edge* if $|E'(R) \cap E_i| < |W_i \cap S(Q^V)|/(8\alpha)$, and it is a *type*-2 *terrible edge* otherwise, in which case $|E'(R) \cap E_i| < |W_i \cap S(Q^E)|/(8\alpha)$ must hold. Let $E^1(R)$ and $E^2(R)$ be the sets of all terrible edges of $E'(R)$ of types 1 and 2, respectively. Then either $|E^1(R)| > 6^\ell/(8\alpha)$, or $|E^2(R)| > 6^\ell/(8\alpha)$ must hold.

Assume first that $|E^1(R)| > 6^\ell/(8\alpha)$. Fix some index $1 \le i \le 6^\ell$, such that W_i is a cluster that is terrible for R, and $|E(R) \cap E_i| < |W_i \cap S(Q^V)|/(8\alpha)$. We assign, to each edge $e \in E_i \cap E^1(R)$, a set of 8α vertices of $W_i \cap S(Q^V)$ arbitrarily, so that every vertex is assigned to at most one edge; we say that the corresponding edge is *responsible* for the vertex. Every edge of $E_i \cap E^1(R)$ is now responsible for 8α distinct vertices of $W_i \cap S(Q^V)$. Once we finish processing all such clusters W_i, we will have assigned, to each edge of $E^1(R)$, a set of 8α distinct vertices of $S(Q^V)$. We conclude that $|S(Q^V)| \ge 8\alpha|E^1(R)| > 6^\ell$. But $|S(Q^V)| = 6^\ell$, a contradiction.

The proof for the second case, where $|E^2(R)| > 6^\ell/(8\alpha)$ is identical, and relies on the fact that $|S(Q^E)| = 6^\ell$. □

We will use the following simple observation.

Observation 5.7. Let $R \in \mathcal{R}''$ be a good random string, with $R = (Q^E, Q^V)$, and let $1 \le i \le 6^\ell$ be an index, such that W_i is not terrible for R. Then $|W_i \cap S(Q^V)| \ge |W_i \cap S(Q^E)|/(8\alpha)$ and $|W_i \cap S(Q^E)| \ge |W_i \cap S(Q^V)|/(8\alpha)$.

Proof. Assume first for contradiction that $|W_i \cap S(Q^V)| < |W_i \cap S(Q^E)|/(8\alpha)$. Consider the edges of $E(R) \cap E_i$. Each such edge must be incident to a distinct vertex of $S(Q^V)$. Indeed, if two edges $e, e' \in E(R) \cap E_i$ are incident to the same vertex $v_j(Q^V, A) \in S(Q^V)$, then, since the other endpoint of each such edge lies in $S(Q^E)$, the two edges belong to the same bundle, a contradiction. Therefore, $|E_i \cap E(R)| \le |W_i \cap S(Q^V)| < |W_i \cap S(Q^E)|/(8\alpha)$, contradicting the fact that W_i is not a terrible cluster for R.

The proof for the second case, where $|W_i \cap S(Q^E)| < |W_i \cap S(Q^V)|/(8\alpha)$ is identical. As before, each edge of $E(R) \cap E_i$ must be incident to a distinct vertex of $S(Q^E)$, as otherwise, a pair $e, e' \in E(R)$ of edges that are incident on the same vertex $v(Q^E, A) \in S(Q^E)$ belong the same bundle. Therefore, $|E_i \cap E(R)| \leq |W_i \cap S(Q^E)| < |W_i \cap S(Q^V)|/(8\alpha)$, contradicting the fact that W_i is not a terrible cluster for R. □

For each good random string $R \in \mathcal{R}''$, we discard the terrible edges from set $E'(R)$, so $|E'(R)| \geq 6^\ell/(4\alpha)$ still holds.

Let $z = 2^{\gamma\ell/8}$. We say that cluster W_i is *heavy* for a random string $R = (Q^E, Q^V) \in \mathcal{R}''$ if $|W_i \cap S(Q^E)|, |W_i \cap S(Q^V)| > z$. We say that an edge $e \in E'(R)$ is *heavy* if it belongs to set E_i, where W_i is a heavy cluster for R. Finally, we say that a random string $R \in \mathcal{R}''$ is *heavy* if at least half of the edges in $E'(R)$ are heavy. Random strings and edges that are not heavy are called *light*. We now consider two cases. The first case happens if at least half of the good random strings are light. In this case, we compute a strategy for the provers to choose assignments to the queries, so that at least a $2^{-\gamma\ell/2}$-fraction of the constraints in $\mathcal{R}'$ are satisfied. In the second case, at least half of the good random strings are heavy. We then compute a partition $\mathcal{H}$ of H' as desired. We now analyze the two cases. Note that if $|E(H')| < z/(8\alpha)$, then Case 2 cannot happen. This is since $h = |E(H')| < z/(8\alpha)$ in this case. Therefore, for all $1 \leq i \leq r$, $|E'_i| \leq h < z/(8\alpha)$ must hold, and so no random strings may be heavy. Therefore, if H' is small enough, we will return a strategy of the provers that satisfies a large fraction of the constraints in $\mathcal{R}'$.

Case 1. This case happens if at least half of the good random strings are light. Let $\mathcal{L} \subseteq \mathcal{R}''$ be the set of the good light random strings, so $|\mathcal{L}| \geq |\mathcal{R}'|/(4\alpha)$. For each such random string $R \in \mathcal{L}$, we let $E^L(R) \subseteq E'(R)$ be the set of all light edges corresponding to R, so $|E^L(R)| \geq 6^\ell/(8\alpha)$. We now describe a randomized algorithm to choose an answer to every query $Q \in \mathcal{Q}^E \cup \mathcal{Q}^V$ with $v(Q) \in H'$. Our algorithm chooses a random index $1 \leq i \leq r$. For every query $Q \in \mathcal{Q}^E \cup \mathcal{Q}^V$ with $v(Q) \in H'$, we consider the set $\mathcal{A}(Q)$ of all answers A, such that some vertex $v(Q, A)$ belongs to W_i (for the case where $Q \in \mathcal{Q}^V$, the vertex is of the form $v_j(Q, A)$). We then choose one of the answers from $\mathcal{A}(Q)$ uniformly at random, and assign it to Q. If $\mathcal{A}(Q) = \emptyset$, then we choose an arbitrary answer to Q.

We claim that the expected number of satisfied constraints of $\mathcal{R}'$ is at least $|\mathcal{R}'|/2^{\gamma\ell/2}$. Since $\mathcal{L} \geq |\mathcal{R}'|/(4\alpha)$, it is enough to show that the expected fraction the good light constraints that are satisfied is at least $4\alpha|\mathcal{L}|/2^{\gamma\ell/2}$, and for that it is sufficient to show that each light constraint $R \in \mathcal{L}$ is satisfied with probability at least $4\alpha/2^{\gamma\ell/2}$.

Fix one such constraint $R = (Q^E, Q^V) \in \mathcal{L}$, and consider an edge $e \in E^L(R)$. Assume that e connects a vertex $v(Q^E, A)$ to a vertex $v_j(Q^V, A')$, and that $e \in E_i$. We say that edge e is happy if our algorithm chose the index i, the answer A to query Q^E, and the answer A' to query Q^V. Notice that due to our construction of bundles, at most one edge $e \in E^L(R)$ may be happy with any choice of the algorithm; moreover, if any edge $e \in E^L(R)$ is happy, then the constraint R is satisfied. The probability that a fixed edge e is happy is at least $1/(8 \cdot 6^\ell z^2 \alpha)$. Indeed, we choose the correct index i with probability $1/6^\ell$. Since e belongs to E_i, W_i is a light cluster for R, and so either $|S(Q^E) \cap W_i| \leq z$, or $|S(Q^V) \cap W_i| \leq z$. Assume without loss of generality that it is the former; the other case is symmetric. Then, since e is not terrible, from Observation 5.7,

$|S(Q^V)\cap W_i| \le 8\alpha z$, and so $|\mathcal{A}(Q^V)| \le 8\alpha z$, while $|\mathcal{A}(Q^E)| \le z$. Therefore, the probability that we choose answer A to Q^E and answer A' to A^V is at least $1/(8\alpha z^2)$, and overall, the probability that a fixed constraint $R \in \mathcal{L}$ is satisfied is at least $|E^L(R)|/(8\cdot 6^\ell z^2\alpha) \ge 1/(64z^2\alpha^2) \ge 4\alpha/2^{\gamma\ell/2}$, since $z = 2^{\gamma\ell/8}$, and $\alpha < 2^{\gamma\ell/32}$.

So far, we have defined a randomized strategy for the provers that satisfies at least a $2^{-\gamma\ell/2}$-fraction of the constraints in $\mathcal{R}'$ in expectation. We can turn this strategy into a deterministic one, that satisfies at least a $2^{-\gamma\ell/2}$-fraction of the constraints in $\mathcal{R}'$, using standard techniques: first, for every cluster W_i, we can compute the expected number of satisfied constraints if the index i is chosen, and then fix the cluster W_i that maximizes this expectation. The randomized strategy of the provers outlined above can then be turned into a deterministic one using the method of conditional expectation, as sketched in Section 2.

Case 2. This case happens if at least half of the good random strings are heavy. Let $\mathcal{R}^H \subseteq \mathcal{R}''$ be the set of all random strings that are good and heavy, so $|\mathcal{R}^H| \ge |\mathcal{R}'|/(4\alpha)$. For each such random string $R \in \mathcal{R}^H$, we let $E^H(R) \subseteq E'(R)$ be the set of all heavy edges corresponding to R. Recall that $|E^H(R)| \ge 6^\ell/(8\alpha)$.

Fix some heavy random string $R \in \mathcal{R}^H$ and assume that $R = (Q^E, Q^V)$. For each $1 \le i \le r$, let $E_i(R) = E^H(R) \cap E_i$. Recall that, if $E_i(R) \ne \emptyset$, then $|W_i \cap S(Q^E)|, |W_i \cap S(Q^V)| \ge z$ must hold, and, from the definition of terrible clusters, $|E_i(R)| \ge z/(8\alpha)$. It is also immediate that $|E_i(R)| \le |E'(R)| \le 6^\ell$.

In the remainder of the algorithm and its analysis, it would be convenient for us if the values $|E_i(R)|$ were roughly the same for all indices $1 \le i \le r$ and random strings $R \in \mathcal{R}^H$ with $E_i(R) \ne \emptyset$. This can be achieved by discarding all but a polylogarithmic fraction of edges in $\bigcup_{R\in\mathcal{R}^H} E^H(R)$, using standard geometric grouping and averaging techniques, which we do next.

Consider some heavy random string $R \in \mathcal{R}^H$. We partition the set $\{1,\dots,6^\ell\}$ of indices into at most $\log(|E^H(R)|) \le \log(6^\ell)$ classes, where index $1 \le y \le 6^\ell$ belongs to class $\mathcal{C}_j(R)$ iff $2^{j-1} < |E^H(R)\cap E_y| \le 2^j$. Then there is some index j_R, so that $\sum_{y\in\mathcal{C}_{j_R}(R)} |E^H(R)\cap E_y| \ge |E^H(R)|/\log(6^\ell)$. We say that R *chooses* the index j_R. Notice that:

$$\sum_{y\in\mathcal{C}_{j_R}(R)} |E^H(R)\cap E_y| \ge \frac{|E^H(R)|}{\log(6^\ell)} \ge \frac{6^\ell}{8\ell\alpha\log 6}.$$

Moreover,

$$|\mathcal{C}_{j_R}(R)| \ge \frac{|E^H(R)|}{\log(6^\ell)\cdot 2^{j_R}} \ge \frac{6^\ell}{8\cdot 2^{j_R}\cdot \ell\alpha\log 6}. \tag{5.1}$$

Let j^* be the index that was chosen by at least $|\mathcal{R}^H|/\log(6^\ell)$ random strings, and let $\mathcal{R}^* \subseteq \mathcal{R}^H$ be the set of all random strings that chose j^*.

We now proceed in two steps. In the first step, we show a randomized algorithm for computing a collection $\mathcal{H}$ of subgraphs of H'. In the second step, we derandomize this algorithm using the method of conditional expectation.

Randomized algorithm. We now show a randomized algorithm to construct a collection

$$\mathcal{H} = \{H_1, \dots, H_{6^\ell}\}$$

of subgraphs of H'. We first define the sets of vertices in these subgraphs, and then the sets of edges. Choose a random ordering of the clusters $W_1, \dots, W_{6^\ell}$; re-index the clusters according to this ordering. For each query $Q \in \mathcal{Q}^E \cup \mathcal{Q}^V$ with $v(Q) \in H''$, add the vertex $v(Q)$ to set $V(H_i)$, where i is the smallest index for which W_i contains at least 2^{j^*-1} vertices of $S(Q)$; if no such index i exists, then we do not add $v(Q)$ to any set.

In order to define the edges of each graph H_i, for every random string $R = (Q^E, Q^V) \in \mathcal{R}^*$, if $i \in \mathcal{C}_{j^*}(R)$, and both $v(Q^E)$ and $v(Q^V)$ belong to $V(H_i)$, then we add the corresponding edge $e(R)$ to $E(H_i)$. This completes the definition of the family $\mathcal{H} = \{H_1, \dots, H_{6^\ell}\}$ of subgraphs of H'. We now analyze their properties. It is immediate to verify that the graphs in $\mathcal{H}$ are disjoint.

Claim 5.8. *For each* $1 \le i \le 6^\ell$, $|E(H_i)| \le |E(H')|/2^{\gamma\ell/16}$.

Proof. Fix some index $1 \le i \le 6^\ell$. An edge $e(R)$ may belong to H_i only if $R \in \mathcal{R}^*$, and $i \in \mathcal{C}_{j^*}(R)$. In that case, E_i contained at least $z/(8\alpha)$ edges of $E(R)$ (since W_i must be heavy for R and it is not terrible for R). Therefore, the number of edges in H_i is bounded by $|E_i| \cdot 8\alpha/z \le 8\alpha h/z \le 8\alpha|E(H'')|/2^{\gamma\ell/8} \le 8\alpha|E(H')|/2^{\gamma\ell/8} \le |E(H')|/2^{\gamma\ell/16}$, since $\alpha \le 2^{\gamma\ell/32}$. ☐

Claim 5.9. $\mathbf{E}\left[\sum_{i=1}^r |E(H_i)|\right] \ge \frac{|E(H')|}{128\ell^2\alpha^2\log^2 6}$.

Proof. Recall that $|\mathcal{R}^*| \ge |\mathcal{R}^H|/\log(6^\ell) \ge |\mathcal{R}'|/(4\alpha\log(6^\ell))$. We now fix $R \in \mathcal{R}^*$ and analyze the probability that $e(R) \in \bigcup_{i=1}^r E(H_i)$. Assume that $R = (Q^E, Q^V)$. Let J be the set of indices $1 \le y \le 6^\ell$, such that $|W_y \cap S(Q^V)| \ge 2^{j^*-1}$. Clearly, $|J| \le 6^\ell/2^{j^*-1}$, and $v(Q^V)$ may only belong to graph H_i if $i \in J$. Similarly, let J' be the set of indices $1 \le y \le 6^\ell$, such that $|W_y \cap S(Q^E)| \ge 2^{j^*-1}$. As before, $|J'| \le 6^\ell/2^{j^*-1}$, and $v(Q^E)$ may only belong to graph H_i if $i \in J$. Observe that every index $y \in \mathcal{C}_{j^*}(R)$ must belong to $J \cap J'$, and, since $j^* = j_R$, from Equation (5.1),

$$|\mathcal{C}_{j^*}(R)| \ge \frac{6^\ell}{8 \cdot 2^{j^*} \cdot \ell\alpha\log 6}.$$

Let $y \in J \cup J'$ be the first index that occurs in our random ordering. If $y \in \mathcal{C}_{j^*}(R)$, then edge $e(R)$ is added to H_y. The probability of this happening is:

$$\frac{|\mathcal{C}_{j^*}(R)|}{|J \cup J'|} \ge \frac{6^\ell/(8 \cdot 2^{j^*} \cdot \ell\alpha\log 6)}{2 \cdot 6^\ell/2^{j^*-1}} = \frac{1}{32\ell\alpha\log 6}.$$

Overall, the expectation of $\sum_{i=1}^r |E(H_i)|$ is at least:

$$\frac{|\mathcal{R}^*|}{32\ell\alpha\log 6} \ge \frac{|\mathcal{R}'|}{128\ell^2\alpha^2\log^2 6} = \frac{|E(H')|}{128\ell^2\alpha^2\log^2 6}.$$ ☐

Deterministic algorithm. We obtain a deterministic algorithm for computing a collection $\mathcal{H}$ of subgraphs of H' with the desired properties, by applying the method of conditional expectation. Specifically, we perform r iterations, where in the ith iteration we choose one of the clusters W_j, for $1 \leq j \leq r$, to be the ith cluster in the ordering. Notice that Claim 5.8 holds regardless of the ordering of the clusters that we choose. Therefore, it is sufficient to choose the ordering in such a way that, for the resulting subgraphs $H_1, \ldots, H_r$,

$$\sum_{i=1}^{r} |E(H_i)| \geq \frac{|E(H')|}{128\ell^2\alpha^2 \log^2 6}\,.$$

Consider the first iteration. We process every cluster W_j in turn, and compute the expected value of $\sum_{i=1}^{r} |E(H_i)|$, if W_j is chosen to be the first cluster in the ordering, and the order of the remaining clusters is random. Once W_j is chosen to be the first cluster, the graph H_1 if fixed, so we can compute $|E(H_1)|$ directly. For every random string $R \in \mathcal{R}^*$ with $e(R) \notin E(H_1)$, we can calculate the probability that $e(R)$ is included in $\bigcup_{i>1} E(H_i)$ using the same analysis as in Claim 5.9. We then choose a cluster W_j that gives the largest expectation of $\sum_{i=1}^{r} |E(H_i)|$, make W_j the first cluster in the ordering, and continue to the next iteration. It is easy to verify that, if $\{H_1, \ldots, H_r\}$ is the final collection of subgraphs of H', then

$$\sum_{i=1}^{r} |E(H_i)| \geq \frac{|E(H')|}{128\ell^2\alpha^2 \log^2 6}\,.$$

Lastly, the final set $\mathcal{H}$ of clusters contains all connected components of every graph in $\{H_1, \ldots, H_r\}$. It is now easy to verify that $\mathcal{H}$ has all required properties.

6 Reduction from (r,h)-GPwB to NDP-Grid

In this section we prove Theorem 4.3, by providing a reduction from (r,h)-GPwB to NDP-Grid. We assume that we are given an instance $\mathcal{I} = (\tilde{G} = (V_1 \cup V_2, E), \mathcal{U}_1, \mathcal{U}_2, h, r)$ of (r,h)-GPwB. Let $|V_1| = N_1, |V_2| = N_2$, $|E| = M$, and $N = N_1 + N_2$. We assume that $\mathcal{I}$ is a valid instance, so, if we denote by $\beta^* = \beta^*(\mathcal{I}) = \sum_{v \in V_1} \beta(v)$, then $h = \beta^*/r$, and $h \geq \max_{v \in V_1 \cup V_2} \{\beta(v)\}$. We also assume that $\mathcal{I}$ is a (d, d', b)-regular instance, for some $d, d', b > 0$, so for every vertex $v \in V_1 \cup V_2$, every bundle $B \in \mathcal{B}(v)$ has cardinality b, and graph $\tilde{G}$ is (d, d')-semiregular. We start by describing the construction of the NDP-Grid instance from the instance $\mathcal{I}$ of (r,h)-GPwB.

6.1 Proof intuition

In this section, we assume that we are given an instance $\mathcal{I}$ of (r,h)-GPwB as described above, and we show how to construct a corresponding instance of the NDP-Grid problem. Before we provide a formal description of the construction, we provide intuition for it. In our construction, the graph $\hat{G}$ associated with the NDP-Grid instance that we are constructing is simply a large enough square grid. We select two rows of this grid, denoted by R' and R'', that are far enough from each other and from the top and bottom grid boundaries. Consider now the graph $\tilde{G} = (V_1 \cup V_2, E)$ associated with the input instance $\mathcal{I}$ of the (r,h)-GPwB problem. We will associate, with every vertex $v_i \in V_1$, a contiguous subpath K_i of the row R', that we call a *block representing* v_i, such that all resulting subpaths K_i are far from each other, and far from the grid boundaries. Similarly, we associate, with every vertex $v'_j \in V_2$, a subpath K'_j of row R'', such that all resulting subpaths K'_j are far from each other and far from the grid boundaries. Intuitively, for every edge $e = (v_i, v'_j) \in E$ of graph $\tilde{G}$, with $v_i \in V_1$ and $v'_j \in V_2$, we would like to create a demand pair $s(e), t(e)$, where $s(e)$ is a vertex lying on K_i, and $t(e)$ is a vertex lying on K'_j. Let us assume for now that we create these demand pairs in such a way that they do not share source or destination vertices. If we now consider some block K_i corresponding to some vertex $v_i \in V_1$, then we need to designate some of its vertices as the source vertices $s(e)$ for all edges $e \in E(\tilde{G})$ that are incident to v_i. For convenience, denote the set of all edges incident to v_i in $\tilde{G}$ by $\{e_1, \dots, e_z\}$. Notice that for each such edge $e_{z'}$, its destination vertex $t(e_{z'})$ lies in a distinct block $K'_{j_{z'}}$. Therefore, the ordering of the destination vertices is fixed, once we fix the ordering of the blocks $\{K'_j\}_{j=1}^{N_2}$ along the row R''. In order to make the routing of these demand pairs easier, it is convenient to place the source vertices $s(e_1), \dots, s(e_z)$ on the path K_i in exactly the same order as the order of their destination vertices on the row R''. Also, we need to select the spacing between these source vertices carefully: if the sources are too far apart, then it is easy to show that we can always route all demand pairs, regardless of whether the corresponding (r,h)-GPwB instance has a high-value solution or not. On the other hand, if the source vertices are too close to each other, then routing the demand pairs, in the case where (r,h)-GPwB has a perfect solution can be challenging. The placement of the destination vertices in blocks K'_j is done similarly.

At a high level, if we assume that the resulting NDP-Grid instance has a large solution value, then we can define a large subgraph $G' \subseteq \tilde{G}$ that can be drawn in the plane with relatively few crossings. Graph G' will contain all edges $e \in E(\tilde{G})$, whose corresponding demand pair is routed by the solution to the NDP-Grid problem instance. It is well known that graphs that can be drawn in the plane with few crossings have balanced cuts of a small value (that is, a partition of the vertices of the graph into two subsets of similar size with only a small number of vertices in each subset that have a neighbor in the other subset). We exploit this in order to iteratively compute balanced cuts in G', until we obtain a partition of G' into r clusters, thus obtaining a solution to the underlying (r,h)-GPwB problem. There are two subtleties with this part of the proof. First, we can only ensure that a drawing of G' with few crossings exists if the distances between the source vertices within each block K_i are small enough, and the same holds for the destination vertices. Second, we are not guaranteed that the resulting solution to the (r,h)-GPwB problem obeys the bundle condition: that is, it is possible that several edges that belong to the same bundle are added to the solution. In order to overcome the latter problem, we slightly modify the construction of the NDP-Grid instance, by unifying some source vertices and some destination vertices, so that, whenever two edges lie in the same bundle, their corresponding demand pairs share their source or destination vertex. This ensures that no two edges of the graph G' induced by the solution to the NDP-Grid problem instance lie in the same bundle, and that the final solution to the (r,h)-GPwB problem instance is indeed a feasible solution.

Lastly, we show that, if the instance $\mathcal{I}$ of (r,h)-GPwB has a perfect solution, then there is a solution to the NDP-Grid problem that routes a large enough number of demand pairs. Intuitively, given a perfect solution $((W_1, \dots, W_r), (E_1, \dots, E_r))$ to the (r,h)-GPwB problem instance, we will route a large subset of the demand pairs corresponding to the edges in the sets $E_1, \dots, E_r$. Due to the large distances between the blocks $\{K_i\}_{v_i \in V_1}$ and between the blocks $\{K'_j\}_{v'_j \in V_2}$, in a sense, we can think about routing r different sets of demand pairs (where each set corresponds to the edges in a distinct set E_i) independently, and exploit the large distances between the blocks in order to combine these routings together. There are two crucial properties that the routing uses. First, we use the fact that within each block K_i, the source vertices are ordered in the same order as their corresponding destination vertices on row R'', and the same holds for each block $K'_j \subseteq R''$. Second, we exploit a "distance property," that, intuitively (though somewhat imprecisely) says that, if $\mathcal{M}'$ is the set of the demand pairs that we attempt to route, then for any pair $(s(e),t(e)),(s(e'),t(e')) \in \mathcal{M}'$ of demand pairs, the total number of destination vertices of the pairs in $\mathcal{M}'$ lying between $t(e)$ and $t(e')$ is at most the distance between $s(e)$ and $s(e')$ on row R'. In order to achieve this distance property, we will carefully select an ordering of the blocks $K'_1,\dots,K'_{N_2}$ that correspond to the vertices of V_2 along the row R''. More precisely, we will fix an arbitrary ordering ρ of the groups in $\mathcal{U}_1$, and we will carefully select an ordering ρ' of the groups in $\mathcal{U}_2$. The former will be used in order to select an ordering of the blocks K_i along the row R', while the latter will be used to determine an ordering of the blocks K'_j along the row R''.

6.2 The construction

We now proceed to describe the reduction formally. We assume for now that we are given some ordering ρ of the groups in $\mathcal{U}_1$, and some ordering ρ' of the groups in $\mathcal{U}_2$; we show later how to select these orderings. Using the ordering ρ, we define an ordering σ of the vertices of V_1, as follows. The vertices that belong to the same group $U \in \mathcal{U}_1$ are placed consecutively in the ordering σ, in an arbitrary order. The ordering between the groups in $\mathcal{U}_1$ is the same as their ordering in ρ. We assume that $V_1 = \{v_1, v_2, \dots, v_{N_1}\}$, where the vertices are indexed according to their order in σ. We then define an ordering σ' of the vertices of V_2 in exactly the same manner, using the ordering ρ' of $\mathcal{U}_2$. We assume that $V_2 = \left\{v'_1, v'_2, \dots, v'_{N_2}\right\}$, where the vertices are indexed according to their ordering in σ'.

Consider some vertex $v \in V_1$. Recall that $\mathcal{B}(v)$ denotes the partition of the edges incident to v in $\tilde{G}$ into bundles, where every bundle is a non-empty subsets of edges, and that $\beta(v) = |\mathcal{B}(v)|$. Recall also that, since graph $\tilde{G}$ is (d,d')-semiregular, $d \le d_{\tilde{G}}(v) < 2d$. Moreover, since every bundle has cardinality b, we get that for every vertex $v \in V_1$:

$$d/b \le \beta(v) < 2d/b. \tag{6.1}$$

Recall also that, since instance $\mathcal{I}$ is valid, $\sum_{v \in V_1} \beta(v) = \beta^*(\mathcal{I}) = hr$. As every bundle contains exactly b edges, we get that:

$$|E(\tilde{G})| = \sum_{v \in V_1} d(v) = \sum_{v \in V_1} \beta(v) \cdot b = hrb. \tag{6.2}$$

Consider again some vertex $v \in V_1$. Recall that each bundle $B \in \mathcal{B}(v)$ corresponds to a single group $U(B) \in \mathcal{U}_2$, and contains all edges that connect v to the vertices of $U(B)$. For every bundle $B \in \mathcal{B}(v)$, we say that the corresponding group $U(B) \in \mathcal{U}_2$ *covers* v.

The ordering ρ' of the groups in $\mathcal{U}_2$ naturally induces an ordering of the bundles in $\mathcal{B}(v)$, where B appears before B' in the ordering iff $U(B)$ appears before $U(B')$ in ρ'. We denote

$$\mathcal{B}(v) = \left\{B_1(v), B_2(v), \dots, B_{\beta(v)}(v)\right\},$$

where the bundles are indexed according to this ordering.

Similarly, for a vertex $v' \in V_2$, every bundle $B \in \mathcal{B}(v')$ corresponds to a group $U(B) \in \mathcal{U}_1$, and contains all edges that connect v' to the vertices of $U(B)$. As before, the ordering ρ of the groups in $\mathcal{U}_1$ naturally defines an ordering of the bundles in $\mathcal{B}(v')$. We denote $\mathcal{B}(v') = \left\{B_1(v'), B_2(v'), \dots, B_{\beta(v')}(v')\right\}$, and we assume that the bundles are indexed according to this ordering.

Lastly, since graph $\tilde{G}$ is (d,d')-semiregular, we get that $|E(\tilde{G})| \geq d'N_2/(4\log N_2) \geq d'N_2/(4\log M)$. Since every group in $\mathcal{U}_2$ contains exactly r vertices of V_2, we get that:

$$|\mathcal{U}_2| = \frac{N_2}{r} \leq \frac{4|E(\tilde{G})|\log M}{d'r}. \tag{6.3}$$

We are now ready to define the instance $\hat{\mathcal{I}} = (\hat{G}, \mathcal{M})$ of NDP-Grid, from the input instance $(\tilde{G} = (V_1, V_2, E), \mathcal{U}_1, \mathcal{U}_2, h, r)$ of (r,h)-GPwB. Let $\lambda = 2048 \cdot \left\lceil M^2 \cdot \log M \right\rceil$. The graph $\hat{G}$ is simply the $(\lambda \times \lambda)$-grid, so $V(\hat{G}) = O(M^4 \log^2 M)$ as required. We now turn to define the set $\mathcal{M}$ of the demand pairs. We first define the set $\mathcal{M}$ itself, without specifying the locations of the corresponding vertices in $\hat{G}$, and later specify a mapping of all vertices participating in the demand pairs to $V(\hat{G})$.

Consider the underlying graph $\tilde{G} = (V_1, V_2, E)$ of the (r,h)-GPwB problem instance. Initially, for every edge $e = (u,v) \in E$, with $u \in V_1, v \in V_2$, we define a demand pair $(s(e), t(e))$ representing e, and add it to $\mathcal{M}$, so that the vertices participating in the demand pairs are all distinct. Next, we process the vertices $v \in V_1 \cup V_2$ one-by-one. Consider first some vertex $v \in V_1$, and some bundle $B \in \mathcal{B}(v)$. Assume that $B = \{e_1, \dots, e_z\}$. Recall that for each $1 \leq i \leq z$, set $\mathcal{M}$ currently contains a demand pair $(s(e_i), t(e_i))$ representing e_i. We unify all vertices $s(e_1), \dots, s(e_z)$ into a single vertex s_B. We then replace the demand pairs $(s(e_1), t(e_1)), \dots, (s(e_z), t(e_z))$ with the demand pairs $(s_B, t(e_1)), \dots, (s_B, t(e_z))$. Once we finish processing all vertices in V_1, we perform the same procedure for every vertex of V_2: given a vertex $v' \in V_2$, for every bundle $B' \in \mathcal{B}(v')$, we unify all destination vertices $t(e)$ with $e \in B'$ into a single destination vertex, that we denote by $t_{B'}$, and we update $\mathcal{M}$ accordingly. This completes the definition of the set $\mathcal{M}$ of the demand pairs.

Observe that each edge of $e \in E$ still corresponds to a unique demand pair in $\mathcal{M}$, that we will denote by $(s_{B(e)}, t_{B'(e)})$, where $B(e)$ and $B'(e)$ are the two corresponding bundles containing e. Given a subset $E' \subseteq E$ of edges of $\tilde{G}$, we denote by $\mathcal{M}(E') = \left\{(s_{B(e)}, t_{B'(e)}) \mid e \in E'\right\}$ the set of all demand pairs corresponding to the edges of E'.

In order to complete the reduction, we need to show a mapping of all source and all destination vertices of $\mathcal{M}$ to the vertices of $\hat{G}$. Let R' and R'' be two rows of the grid $\hat{G}$, lying at a distance at least $\lambda/4$ from each other and from the top and the bottom boundaries of the grid, with R' lying above R''. We will map all vertices of $S(\mathcal{M})$ to R', and all vertices of $T(\mathcal{M})$ to R'', as follows.

Locations of the sources. Let $K_1, K_2, \dots, K_{N_1}$ be a collection of N_1 disjoint subpaths of R', where each subpath contains $1024 \cdot \lceil h \cdot \log M \rceil$ vertices; the subpaths are indexed according to their left-to-right ordering on R', and every consecutive pair of the paths is separated by at least $10M$ vertices from each other and from the left and the right boundaries of $\hat{G}$. Observe that the width λ of the grid is large enough to allow this, as $h \leq M$ must hold. For all $1 \leq i \leq N_1$, we call K_i the *block representing the vertex* $v_i \in V_1$. We now fix some $1 \leq i \leq N_1$ and consider the block K_i representing the vertex v_i. We map the source vertices $s_{B_1(v_i)}, s_{B_2(v_i)}, \ldots, s_{B_{\beta(v_i)}(v_i)}$ to vertices of K_i, so that they appear on K_i in this order, and every consecutive pair of sources is separated by exactly $512 \cdot \lceil h \cdot \log M/\beta(v_i) \rceil$ vertices.

Locations of the destinations. Similarly, we let $K'_1, K'_2, \ldots, K'_{N_2}$ be a collection of N_2 disjoint subpaths of R'', each of which contains $1024 \cdot \lceil h \cdot \log M \rceil$ vertices, so that the subpaths are indexed according to their left-to-right ordering on R'', and every consecutive pair of the paths is separated by at least $10M$ vertices from each other and from the left and the right boundaries of $\hat{G}$. We call K'_i the *block representing the vertex* $v'_i \in V_2$. We now fix some $1 \leq i \leq N_2$ and consider the block K'_i representing the vertex v'_i. We map the destination vertices

$$t_{B_1(v'_i)}, t_{B_2(v'_i)}, \ldots, t_{B_{\beta(v'_i)}(v'_i)}$$

to vertices of K'_i, so that they appear on K'_i in this order, and every consecutive pair of destinations is separated by exactly $512 \cdot \lceil h \cdot \log M/\beta(v'_i) \rceil$ vertices.

This concludes the definition of the instance $\hat{\mathcal{I}} = (\hat{G}, \mathcal{M})$ of NDP-Grid, except for the procedure for selecting the orderings ρ and ρ' of the groups in $\mathcal{U}_1$ and $\mathcal{U}_2$, respectively. The following immediate observation will be useful to us.

Observation 6.1. Consider a vertex $v_j \in V_1$, and let $\mathcal{N}_j \subseteq \mathcal{M}$ be any subset of demand pairs, whose sources are all distinct and lie on K_j. Assume that $\mathcal{N}_j = \{(s_1, t_1), \ldots, (s_y, t_y)\}$, where the demand pairs are indexed according to the left-to-right ordering of their source vertices on K_j. Then $t_1, \ldots, t_y$ appear in this left-to-right order on R''.

Choosing the orderings of the groups. We now show an algorithm for selecting the ordering ρ of $\mathcal{U}_1$ and ρ' of $\mathcal{U}_2$ that are used in the reduction. We let ρ be an arbitrary ordering of the groups in $\mathcal{U}_1$. Before we describe an algorithm for computing the ordering ρ' of $\mathcal{U}_2$, we first define the properties that we need from it.

Assume that we are given some ordering ρ' over the groups in $\mathcal{U}_2$. Consider first some vertex $v_i \in V_1$, and recall that we have denoted its bundles by $B_1(v_i), \ldots, B_{\beta(v_i)}$. Recall that every bundle $B_j(v_i)$ contains all edges connecting v_i to a single group of $\mathcal{U}_2$, that we denote, for convenience, by U^i_j. We assume that the bundles are indexed so that $U^i_1, \ldots, U^i_{\beta_i(v_i)}$ appear in ρ' in this order. Recall that, from Equation (6.1), $d/b \leq \beta(v_i) < 2d/b$. Intuitively, we will only focus on the first $\lceil d/(2b) \rceil$ bundles of v_i, and we will never attempt to route demand pairs that correspond to the remaining bundles. We say that vertex v_i is *happy* with the ordering ρ' of $\mathcal{U}_2$, if, for all $1 \leq j < \lceil d/(2b) \rceil$, the total number of groups that appear strictly between U^i_j and U^{i+1}_j in ρ' is at most

$$O\left(\frac{|\mathcal{U}_2| b \log^2 M}{d}\right)$$

(notice that the indexing of the bundles $B_j(v_i)$ and of the corresponding groups U^i_j depends on ρ'). We say that ordering ρ' is *good* if every vertex $v_i \in V_1$ is happy with this ordering. Intuitively, choosing an ordering ρ' that is good will ensure that the distance property mentioned above holds, which will allow us to route a large set of the demand pairs in the case where there is a perfect solution to the (r,h)-GPwB instance. We now show that we can compute such an ordering ρ' efficiently.

Lemma 6.2. *There is an algorithm whose running time is polynomial in $|V(\tilde{G})|$, that computes a good ordering ρ' of $\mathcal{U}_2$.*

Proof. The algorithm will compute a partition $\mathcal{U}^1, \mathcal{U}^2, \ldots, \mathcal{U}^q$ of $\mathcal{U}_2$ into subsets, each of which has cardinality $O\left(\frac{|\mathcal{U}_2| b \log^2 M}{d}\right)$. In the final ordering ρ', groups of $\mathcal{U}_2$ that lie within a single set $\mathcal{U}^i$ are placed consecutively, in an arbitrary order, while groups lying in different sets are ordered according to the ordering $\mathcal{U}^1, \mathcal{U}^2, \ldots, \mathcal{U}^q$ of these sets.

The algorithm consists of a number of iterations, where in the ith iteration we compute the subset $\mathcal{U}^i \subseteq \mathcal{U}_2$ of groups. We now describe a single iteration. Recall that for every vertex $v \in V_1$, we have defined a collection of groups that cover v—these are all groups $U \in \mathcal{U}_2$ such that there is a bundle $B \in \mathcal{B}(v)$, connecting v to vertices of U. The total number of groups that cover v is $\beta(v) \geq d/b$ from Equation 6.1. We say that vertex v is *active* in iteration i if the number of groups that cover u and belong to $\mathcal{U}^1 \cup \cdots \cup \mathcal{U}^{i-1}$ is less than $\lceil d/(2b) \rceil$. Let $\mathcal{U}' = \mathcal{U}_2 \setminus (\mathcal{U}^1 \cup \cdots \cup \mathcal{U}^{i-1})$, and let $V' \subseteq V_1$ be the set of all vertices that are active in iteration i. Recall that by the definition of active vertices, for each $v \in V'$, there are at least $d/(2b)$ groups in $\mathcal{U}'$ that cover v. We next show that we can efficiently compute a subset $\mathcal{U}^i \subseteq \mathcal{U}_2$ of $O\left(\frac{|\mathcal{U}_2| b \log^2 M}{d}\right)$ groups, such that for every active vertex $v \in \mathcal{U}'$, at least one group in $\mathcal{U}^i$ covers v. Note that, if we manage to select the groups $\mathcal{U}^1, \mathcal{U}^2, \ldots, \mathcal{U}^q$ with this property, such that $\bigcup_i \mathcal{U}^i = \mathcal{U}_2$, then it is immediate to verify that the resulting ordering ρ' of $\mathcal{U}_2$ is a good ordering. This is since, for every vertex $v \in V_1$, if v was active for i iterations, then $\mathcal{U}^1 \cup \cdots \cup \mathcal{U}^i$ contain at least $\lceil d/(2b) \rceil$ groups that cover v, and every set $\mathcal{U}^{i'}$ for $1 \leq i' \leq i$ contains at least one such group. The following claim will then complete the proof of the lemma.

Claim 6.3. *There is an efficient algorithm, that, given an integer $i > 0$, collections $\mathcal{U}^1, \ldots, \mathcal{U}^{i-1}$ of subsets of $\mathcal{U}_2$, and the set $V' \subseteq V_1$ of vertices that are active in iteration i, such that $V' \neq \emptyset$, computes a collection $\mathcal{U}^i \subseteq \mathcal{U}_2 \setminus (\mathcal{U}^1 \cup \cdots \cup \mathcal{U}^{i-1})$ of $O\left(\frac{|\mathcal{U}_2| b \log M}{d}\right)$ groups, such that for every active vertex $v \in \mathcal{U}'$, at least one group in $\mathcal{U}^i$ covers v.*

Proof. For convenience, we denote $\mathcal{U}' = \mathcal{U}_2 \setminus (\mathcal{U}^1 \cup \cdots \cup \mathcal{U}^{i-1})$. The algorithm exploits the Set Cover problem. In this problem, we are given as input a collection $\{1, \ldots, n\}$ of elements, and a collection $\mathcal{S} = \{S_1, \ldots, S_m\}$ of subsets of elements, so for all $1 \leq j \leq m$, $S_j \subseteq \{1, \ldots, m\}$. The goal is to compute a minimum cardinality collection $\mathcal{S}' \subseteq \mathcal{S}$ of subsets, such that every element in $\{1, \ldots, n\}$ belongs to at least one subset in $\mathcal{S}'$. We use the following observation, whose proof appears in the Appendix.[4]

Observation 6.4. There is an efficient algorithm, that, given an instance of the Set Cover problem with n elements and m sets, such that every element belongs to at least m/t sets, computes a solution containing at most $O(t \log n)$ sets.

[4]This proof was suggested to us by one of the reviewers of the paper. A previous version of the paper contained a slightly weaker proof.

In order to complete the proof of Claim 6.3, we set up an instance of the Set Cover problem, where every group in $\mathcal{U}'$ represents a set, every vertex in V' is an element, and element $v \in V'$ belongs to a set S_U representing a group $U \in \mathcal{U}'$ if and only if U covers v. Recall that we are guaranteed that every element belongs to at least $d/(2b)$ sets. Setting $t = |\mathcal{U}'| \cdot (2b)/d$, and using the algorithm from Observation 6.4, we obtain a solution to the resulting instance of the Set Cover problem of value

$$O\left(\frac{|\mathcal{U}_2| b \log N_1}{d}\right) \le O\left(\frac{|\mathcal{U}_2| b \log M}{d}\right),$$

which immediately defines the desired set $\mathcal{U}^i \subseteq \mathcal{U}'$ of groups. □

Our algorithm employs Claim 6.3 to construct the sets $\mathcal{U}^1, \mathcal{U}^2, \ldots$ of groups, until the set V' of vertices that are active becomes empty. Each subsequent set $\mathcal{U}^i$ then contains an arbitrary subset of $O\left(\frac{|\mathcal{U}_2| b \log M}{d}\right)$ groups of $\mathcal{U}_2 \setminus (\mathcal{U}^1 \cup \cdots \cup \mathcal{U}^{i-1})$. □

This concludes the description of the reduction from (r,h)-GPwB to NDP-Grid. In order to complete the proof of Theorem 4.3, it is now enough to prove the following two theorems, which is done in the next two subsections.

Theorem 6.5. *Let $\mathcal{I} = (\tilde{G} = (V_1, V_2, E), \mathcal{U}_1, \mathcal{U}_2, h, r)$ be a valid and (d, d', b)-regular instance of* (r,h)-GPwB *such that $\mathcal{I}$ has a perfect solution. Then there is a solution to instance $\hat{\mathcal{I}}$ of* NDP-Grid, *routing $\Omega(\beta^*(\mathcal{I})/\log^2 M)$ demand pairs via a set of spaced-out paths.*

Theorem 6.6. *There is an efficient algorithm, that, given a valid and (d, d', b)-regular instance $\mathcal{I} = (\tilde{G} = (V_1, V_2, E), \mathcal{U}_1, \mathcal{U}_2, h, r)$ of the* (r,h)-GPwB *problem with $|E| = M$, the corresponding instance $\hat{\mathcal{I}}$ of* NDP-Grid, *and a solution $\mathcal{P}^*$ to $\hat{\mathcal{I}}$, computes a solution to the* (r,h)-GPwB *instance $\mathcal{I}$ of value at least $\Omega(|\mathcal{P}^*|/\log^3 M)$.*

6.3 From partitioning to routing: proof of Theorem 6.5

The goal of this subsection is to prove Theorem 6.5.

Let $((W_1, \ldots, W_r), (E_1, \ldots, E_r))$ be a perfect solution to instance $\mathcal{I}$ of (r,h)-GPwB. Recall that by the definition of a perfect solution, for each group $U \in \mathcal{U}_1 \cup \mathcal{U}_2$, every set W_i contains exactly one vertex of U, and moreover, for each $1 \le i \le r$, $|E_i| = h = \beta^*(\mathcal{I})/r$.

We let $E^0 = \bigcup_{i=1}^r E_i$, so $|E^0| = \beta^*(\mathcal{I}) = hr$. Let $\mathcal{M}^0 \subseteq \mathcal{M}$ be the set of all demand pairs corresponding to the edges of E^0. Note that we are guaranteed that no two demand pairs in $\mathcal{M}^0$ share a source or a destination, since no two edges of E^0 belong to the same bundle. We now define a slightly smaller subset $E^1 \subseteq E^0$ of edges, and a corresponding set $\mathcal{M}^1 \subseteq \mathcal{M}^0$ of demand pairs. Since the value of the solution $((W_1, \ldots, W_r), (E_1, \ldots, E_r))$ of the (r,h)-GPwB problem instance is $hr = \beta^*(\mathcal{I}) = \sum_{v \in V_1} \beta(v)$, for every vertex $v \in V_1$, there are exactly $\beta(v)$ edges in E^0—one edge from each bundle $B \in \mathcal{B}(v)$. Recall that the total number of such bundles is $\beta(v) < 2d/b$ from Equation (6.1). Recall that every bundle $B \in \mathcal{B}(v)$

corresponds to a distinct group $U(B) \in \mathcal{U}_2$, that covers v. Moreover, we have used the ordering ρ' of the groups in $\mathcal{U}_2$ in order to define an ordering of the bundles in $\mathcal{B}(v)$. Let $\{e^i_1, \dots, e^i_{\beta(v_i)}\}$ be the set of all edges of E^0 that are incident to vertex v_i. We assume that these edges are indexed according to the ordering of their corresponding bundles. We discard all but the first $\lceil d/(2b) \rceil$ edges from this set. We then denote by $E^1 \subseteq E^0$ the resulting set of edges, obtained after processing every vertex in V_1. It is easy to verify that $|E^1| = \Omega(|E^0|)$. We let $\mathcal{M}^1 \subseteq \mathcal{M}^0$ be the set of all demand pairs corresponding to the edges of E^1.

Assume now that we are given any subset $\mathcal{M}' \subseteq \mathcal{M}^1$ of demand pairs. We define an ordering $\eta(\mathcal{M}')$ of the demand pairs in $\mathcal{M}'$, by exploiting the perfect solution $((W_1, \dots, W_r), (E_1, \dots, E_r))$ to instance $\mathcal{I}$ of (r,h)-GPwB, as follows. Recall that every demand pair $(s,t) \in \mathcal{M}^1$ corresponds to some edge $e \in E_1 \cup \dots \cup E_r$, that we denote by $e(s,t)$. First, we partition the demand pairs in $\mathcal{M}'$ into subsets $\mathcal{M}'_1, \dots, \mathcal{M}'_r$, as follows: demand pair $(s,t) \in \mathcal{M}'$ is added to subset $\mathcal{M}'_i$ iff the corresponding edge $e(s,t)$ lies in E_i. The demand pairs that lie in different sets $\mathcal{M}'_1, \dots, \mathcal{M}'_r$ are then ordered according to the natural order of these sets. Consider now some set $\mathcal{M}'_i$ of demand pairs. We order the demand pairs in $\mathcal{M}'_i$ according to the left-to-right order of their destination vertices on row R''. Let $\eta(\mathcal{M}')$ be the resulting ordering of the demand pairs in $\mathcal{M}'$. Next, we define a property of a subset of the demand pairs, called a *distance property*. We later show that, if we are given a subset $\mathcal{M}' \subseteq \mathcal{M}^1$ of demand pairs that has the distance property, then all demand pairs in $\mathcal{M}'$ can be routed via spaced-out paths. We also show that there is a large subset $\mathcal{M}' \subseteq \mathcal{M}^1$ of the demand pairs that has the the distance property.

We now proceed to define a distance property. We assume that we are given a set $\mathcal{M}' \subseteq \mathcal{M}^1$ of demand pairs, and consider the corresponding ordering $\eta = \eta(\mathcal{M}')$ of these demand pairs. We denote $\mathcal{M}' = \{(s_1,t_1), \dots, (s_{|\mathcal{M}'|}, t_{|\mathcal{M}'|})\}$, where the demand pairs are indexed according to their ordering in η. Notice that this ordering is not necessarily the same as the ordering of the set $S(\mathcal{M}')$ of source vertices on row R', or the ordering of the set $T(\mathcal{M}')$ of the destination vertices on row R''.

Consider some pair $s, s' \in S(\mathcal{M}')$ of source vertices, and assume that $s = s_i$ and $s' = s_j$. We denote $N_{\mathcal{M}'}(s,s') = |i-j| - 1$—the number of the demand pairs in the ordering η, that lie between (s_i,t_i) and (s_j,t_j).

Definition 6.7. Given a set $\mathcal{M}' \subseteq \mathcal{M}^1$ of demand pairs, we say that two source vertices $s, s' \in S(\mathcal{M}')$ are *consecutive* with respect to $\mathcal{M}'$ if $s \neq s'$, and no vertex of $S(\mathcal{M}')$ lies strictly between s and s' on row R'. We say that $\mathcal{M}'$ has the *distance property* if for every pair $s, s' \in S(\mathcal{M}')$ of source vertices that are consecutive with respect to $\mathcal{M}'$, $N_{\mathcal{M}'}(s,s') < d(s,s')/4$ holds, where $d(s,s')$ is the distance between s and s' in graph $\hat{G}$.

In order to complete the proof of Theorem 6.5, we proceed in two steps. First, we show that there is a large enough subset $\mathcal{M}' \subseteq \mathcal{M}^1$ of demand pairs that has the distance property. Next, we show that any such set $\mathcal{M}'$ of demand pairs can be routed via spaced-out paths.

Lemma 6.8. *There is a subset $\mathcal{M}' \subseteq \mathcal{M}^1$ of demand pairs that has the distance property, and $|\mathcal{M}'| = \Omega(|\mathcal{M}^1|/\log^2 M) = \Omega(|\mathcal{M}^0|/\log^2 M)$.*

Lemma 6.9. *Assume that $\mathcal{M}' \subseteq \mathcal{M}^1$ is a subset of demand pairs that has the distance property. Then there is a spaced-out set $\mathcal{P}$ of paths routing all pairs of $\mathcal{M}'$ in graph $\hat{G}$.*

The above two lemmas finish the proof of Theorem 6.5, since $|\mathcal{M}^0| = \beta^*(\mathcal{I})$. We prove these lemmas in the following two subsections.

6.3.1 Proof of Lemma 6.8

Let $\eta^1 = \eta(\mathcal{M}^1)$ be the ordering of the demand pairs in $\mathcal{M}^1$. We show that this ordering *almost* has the distance property: if s, s' are two consecutive sources with respect to $\mathcal{M}^1$, then $N_{\mathcal{M}^1} \le O(d(s,s')\log^2 M)$ holds. We then use a simple procedure in order to select a subset $\mathcal{M}' \subseteq \mathcal{M}^1$ of $\Omega(|\mathcal{M}^1|/\log^2 M)$ demand pairs that has the distance property.

Analyzing ordering η^1. Recall that ordering η^1 of the demand pairs in $\mathcal{M}^1$ was defined using the perfect solution $((W_1,\dots,W_r),(E_1,\dots,E_r))$ to instance $\mathcal{I}$ of (r,h)-GPwB. We have partitioned the demand pairs in $\mathcal{M}^1$ into subsets $\mathcal{M}_1,\dots,\mathcal{M}_r$, by using the partition $\{E_1,\dots,E_r\}$ of their corresponding edges. The demand pairs that lie in different sets $\mathcal{M}_1,\dots,\mathcal{M}_r$ are then ordered according to the natural order of these sets, while the demand pairs in each set $\mathcal{M}_i$ are ordered according to the left-to-right order of their destination vertices on row R''. We need the following claim.

Claim 6.10. *Let $s,s' \in S(\mathcal{M}^1)$ be any pair of source vertices that are consecutive with respect to $\mathcal{M}^1$. Then $N_{\mathcal{M}^1}(s,s') \le O(d(s,s')\log^2 M)$.*

Proof. First, if s and s' belong to distinct blocks K_i on row R', then, since the distance between every pair of blocks is at least M, while $|\mathcal{M}^1| \le M$, the assertion clearly holds. Therefore, we can assume that there is some block K_i, with $s,s' \in K_i$. Recall that block K_i represents vertex $v_i \in V_1$. Assume w.l.o.g. that $v_i \in W_j$, in the solution to the (r,h)-GPwB problem.

Let $(s,t),(s',t') \in \mathcal{M}^1$ be the demand pairs corresponding to these source vertices; the demand pairs are uniquely defined, since the pairs in $\mathcal{M}^1$ do not share their source or their destination vertices. Let e be the edge of $\tilde{G}$ corresponding to the demand pair (s,t), and let e' be the edge of $\tilde{G}$ corresponding to the demand pair (s',t'). Then e and e' are both incident to the vertex v_i, and they lie in distinct bundles of $\mathcal{B}(v_i)$. Recall that we have used the ordering ρ' of the groups in $\mathcal{U}_2$ in order to define an ordering of the bundles in $\mathcal{B}(v_i)$, which we used in turn in order to define an ordering $\{e_i^1,\dots,e_i^{\beta(v_i)}\}$ of the edges incident to v_i that belong to E^0. Only the first $\lceil d/(2b)\rceil$ of these edges were added to E^1. Moreover, since the solution to instance $\mathcal{I}$ of (r,h)-GPwB that we have considered is perfect, every bundle incident to v_i contributed exactly one edge to E^0. Therefore, if we denote $e = e_i^z$, then $e' = e_i^{z+1}$ (or the other way around; we assume it is the former), and $z+1 \le \lceil d/(2b)\rceil$. Let $U \in \mathcal{U}_2$ be the group corresponding to the bundle of e; that is, e connects v_i to a vertex of U. Let U' be defined similarly to edge e'. From our definition of a good ordering, there are at most $O(|\mathcal{U}_2| b \log^2 M/d)$ groups that lie between U and U' in the ordering ρ'. We will use this fact in order to bound $N_{\mathcal{M}^1}(s,s')$, that we denote, for convenience, by $N(s,s')$.

Let $(s'',t'') \in \mathcal{M}^1$ be any demand pair that lies between (s,t) and (s',t') in the ordering η^1. Then the edge $e'' \in E^1$ corresponding to this demand pair must also lie in set E_j that corresponds to the cluster W_j in the solution to instance $\mathcal{I}$ of (r,h)-GPwB, and moreover, t'' must lie between t and t' on row R''. If t'' lies in some block K'_x, that represents a vertex $v'_x \in V_2$, then the group U'' to which v'_x belongs must lie between U and U' in the ordering ρ'. Recall that, since ρ' is a good ordering of the groups in $\mathcal{U}_2$, there are at most $O(|\mathcal{U}_2| b \log^2 M/d)$ groups that lie between U and U' in the ordering ρ'. Let us now consider any such group U'', that consists of r vertices. Exactly one vertex of U'' lies in W_j; we denote it by $v'(U'')$. This vertex has degree at most d' in $\tilde{G}$, and so $\mathcal{B}(v'(U''))$ contains at most d'/b bundles. Therefore, at most d'/b edges that are incident to $v'(U'')$ belong to E^1. Altogether, we conclude that:

$$\begin{aligned} N(s,s') &\le O\left(\frac{|\mathcal{U}_2| b \log^2 M}{d}\right) \cdot \frac{d'}{b} \\ &\le O\left(\frac{|\mathcal{U}_2| d' \log^2 M}{d}\right) \\ &\le O\left(\frac{|E(\tilde{G})| \log^3 M}{dr}\right) \\ &\le O\left(\frac{hb \log^3 M}{d}\right). \end{aligned}$$

(We have used Equation (6.3) for the penultimate inequality and Equation (6.2) for the last inequality.)

Recall that, from our construction, $d(s,s') = 512 \cdot \lceil h \cdot \log M/\beta(v_i) \rceil \ge 256 hb \log M/d$, since, from Equation (6.1), $\beta(v) < 2d/b$. We conclude that $N_{\mathcal{M}^1}(s,s') \le O(d(s,s') \log^2 M)$. □

We obtain the following immediate corollary of Claim 6.10.

Corollary 6.11. *There is a constant c, such that, for any pair $s,s' \in S(\mathcal{M}^1)$ of source vertices (that are not necessarily consecutive with respect to $\mathcal{M}^1$), $N_{\mathcal{M}^1}(s,s') \le c d(s,s') \log^2 M$ holds.*

Defining $\mathcal{M}'$. We denote $\mathcal{M}^1 = \left\{(s_1,t_1), \ldots, (s_{|\mathcal{M}^1|}, t_{|\mathcal{M}^1|})\right\}$, where the demand pairs are indexed according to the ordering η^1. We now let:

$$\mathcal{M}' = \left\{(s_i,t_i) \mid i = 1 \bmod \left\lceil 8c \log^2 M \right\rceil\right\}.$$

We let η be the ordering of the demand pairs in $\mathcal{M}'$, induced by η^1. It is immediate to verify that $|\mathcal{M}'| = \Omega(|\mathcal{M}^1|/\log^2 M) = \Omega(|\mathcal{M}^0|/\log^2 M)$, and that $\eta = \eta(\mathcal{M}')$. Moreover, if $s,s' \in S(\mathcal{M}')$ is any pair of source vertices, then $N_{\mathcal{M}'}(s,s') \le N_{\mathcal{M}^1}/\left\lceil 8c \log^2 M \right\rceil \le d(s,s')/4$ from Corollary 6.11. Therefore, $\mathcal{M}'$ has the distance property.

6.3.2 Proof of Lemma 6.9

Recall that R' and R'' are the rows of $\hat{G}$ containing the vertices of $S(\mathcal{M})$ and $T(\mathcal{M})$ respectively, and that R' and R'' lie at distance at least $\lambda/4$ form each other and from the top and the bottom boundaries of $\hat{G}$, where $\lambda > M^2$ is the dimension of the grid. Let R be any row lying between R' and R'', within distance at least $\lambda/16$ from each of them.

We denote $M' = |\mathcal{M}'|$, and $\mathcal{M}' = \{(s_1,t_1),\ldots,(s_{M'},t_{M'})\}$, where the pairs are indexed according to their ordering in $\eta = \eta(\mathcal{M}')$. To recap, the ordering η of $\mathcal{M}'$ was defined as follows. We have defined a partition $\mathcal{M}_1,\ldots,\mathcal{M}_r$ of the demand pairs in $\mathcal{M}'$, according to the partition $E_1,\ldots,E_r$ of their corresponding edges. For each $1 \le i \le r$, the demand pairs in $\mathcal{M}_i$ appear consecutively in η, ordered according to the left-to-right ordering of their destination vertices on row R'', while the order of the demand pairs lying in different sets $\mathcal{M}_i$ follows the indexing of these subsets.

Let $X = \{x_i \mid 1 \le i \le M'\}$ be a set of vertices of R, where x_i is the $(2i)$th vertex of R from the left. We will construct a set $\mathcal{P}^*$ of spaced-out paths routing all demand pairs in $\mathcal{M}'$, so that the path P_i routing (s_i,t_i) intersects the row R at exactly one vertex—the vertex x_i. Notice that row R partitions the grid $\hat{G}$ into two subgrids: a top subgrid G^t spanned by all rows that appear above R (including R), and a bottom subgrid G^b spanned by all rows that appear below R (including R).

It is now enough to show that there are two sets of spaced-out paths: set $\mathcal{P}^1$ routing all pairs in $\{(s_i,x_i) \mid 1 \le i \le M'\}$ in the top grid G^t, and set $\mathcal{P}^2$ routing all pairs in $\{(t_i,x_i) \mid 1 \le i \le M'\}$ in the bottom grid G^b, so that both sets of paths are internally disjoint from R.

Routing in the top grid. Consider some vertex $v_j \in V_1$, and the corresponding block K_j. We construct a subgrid $\hat{K}_j$ of $\hat{G}$, containing K_j, that we call a box, as follows. Let $\mathcal{C}_j$ be the set of all columns of $\hat{G}$ intersecting K_j. We augment $\mathcal{C}_j$ by adding $2M$ columns lying immediately to the left and $2M$ columns lying immediately to the right of $\mathcal{C}_j$, obtaining a set $\hat{\mathcal{C}}_j$ of columns. Let $\hat{\mathcal{R}}$ contain three rows: row R'; the row lying immediately above R'; and the row lying immediately below R'. Then box $\hat{K}_j$ is the subgrid of $\hat{G}$ spanned by the rows in $\hat{\mathcal{R}}$ and the columns in $\hat{\mathcal{C}}_j$. Since every block is separated by at least $10M$ columns from every other block, as well as from the left and the right boundaries of $\hat{G}$, the resulting boxes are all disjoint, and every box is separated by at least $2M$ columns of $\hat{G}$ from every other box, and from the left and the right boundaries of $\hat{G}$.

We will initially construct a set $\mathcal{P}^1$ of spaced-out paths in G^t, such that each path $P_i \in \mathcal{P}^1$, for $1 \le i \le M'$, originates from the vertex x_i, and visits the boxes $\hat{K}_1,\hat{K}_2,\ldots,\hat{K}_{N_1}$ in this order. We will ensure that each such path P_i contains the corresponding source vertex s_i. Eventually, by suitably truncating each such path P_i, we will ensure that it connects x_i to s_i.

Claim 6.12. *Consider some vertex $v_j \in V_1$, and the corresponding box $\hat{K}_j$. Denote $Y_j = S(\mathcal{M}') \cap V(K_j)$, $M_j = |Y_j|$, and assume that $Y_j = \{s_{i_1}, s_{i_2}, \ldots, s_{i_{M_j}}\}$, where the indexing of the vertices of Y_j is consistent with the indexing of the vertices of $S(\mathcal{M}')$ induced by the ordering η of the demand pairs in $\mathcal{M}'$, and $i_1 < i_2 < \ldots < i_{M_j}$. Then there is a set $\mathcal{W}_j$ of M' columns of the box $\hat{K}_j$, such that:*

- *set $\mathcal{W}_j$ does not contain a pair of consecutive columns; and*
- *for each $1 \le z \le M_j$, the i_zth column of $\mathcal{W}_j$ from the left contains the source vertex s_{i_z}.*

Proof. Observe that from Observation 6.1, the vertices $s_{i_1}, s_{i_2}, \ldots, s_{i_{M_j}}$ must appear in this left-to-right order on R', while the vertices $x_{i_1}, x_{i_2}, \ldots, x_{i_{M_j}}$ appear in this left-to-right order on R. Moreover, for all $1 \le z < M_j$, $i_{z+1} - i_z - 1 = N_{\mathcal{M}'}(s_{i_z}, s_{i_{z+1}}) \le d(s_{i_z}, s_{i_{z+1}})/4$, from the distance property of $\mathcal{M}_j$. We add to

$\mathcal{W}_j$ all columns of $\hat{K}_j$ where the vertices of Y_j lie. For each $1 \le z < M_j$, we also add to $\mathcal{W}_j$ an arbitrary set of $(i_{z+1} - i_z - 1)$ columns lying between the column of s_{i_z} and the column of $s_{i_{z+1}}$, so that no pair of columns in $\mathcal{W}_j$ is consecutive. Finally, we add to $\mathcal{W}_j$ $(i_1 - 1)$ columns that lie to the left of the column of s_{i_1}, and $(M' - i_{M_j})$ columns that lie to the right of the column of $s_{i_{M_j}}$. We make sure that no pair of columns in $\mathcal{W}_j$ is consecutive—it is easy to see that there are enough columns to ensure that. □

Claim 6.13. *There is a set $\mathcal{P}^1 = \{P_1, \ldots, P_{M'}\}$ of spaced-out paths in G^t, that are internally disjoint from R, such that for each $1 \le i \le M'$, path P_i originates from vertex x_i, and for all $1 \le j \le N_1$, it contains the ith column of $\mathcal{W}_j$; in particular, it contains vertex s_i.*

We defer the proof of the claim to Section C of the appendix; see Figure 4 for an illustration of the routing.

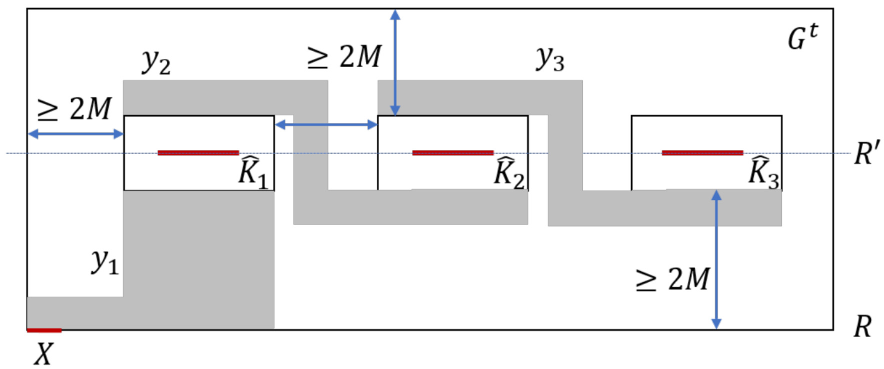

Figure 4: Routing in the top grid

Routing in the bottom grid. Consider some vertex $v'_j \in V_2$, and the corresponding block K'_j. We construct a box $\hat{K}'_j$ containing K'_j exactly as before. As before, the resulting boxes are all disjoint, and every box is separated by at least $2M$ columns of $\hat{G}$ from every other box, and from the left and the right boundaries of $\hat{G}$.

As before, we will initially construct a set $\mathcal{P}^2$ of spaced-out paths in G^b, such that each path $P'_i \in \mathcal{P}^2$, for $1 \le i \le M'$, originates from the vertex x_i, and visits the boxes $\hat{K}'_1, \hat{K}'_2, \ldots, \hat{K}'_{N_2}$ in turn. We will ensure that each such path P'_i contains the corresponding destination vertex t_i. Eventually, by suitably truncating each such path P'_i, we will ensure that it connects x_i to t_i.

Claim 6.14. *Consider some vertex $v'_j \in V_2$, and the corresponding box $\hat{K}'_j$. Denote $Y'_j = T(\mathcal{M}') \cap V(K'_j)$, $M'_j = |Y'_j|$, and assume that $Y'_j = \{t_{i_1}, t_{i_2}, \ldots, t_{i_{M'_j}}\}$, where the indexing of the vertices of Y'_j is consistent with the ordering $\eta = \eta(\mathcal{M}')$ of the demand pairs in $\mathcal{M}'$, and $i_1 < i_2 < \ldots < i_{M'_j}$. Then there is a set $\mathcal{W}'_j$ of M' columns of the box $\hat{K}'_j$, such that:*

- *set $\mathcal{W}'_j$ does not contain a pair of consecutive columns; and*

- *for each* $1 \leq z \leq M'_j$*, the* i_z*th column of* $\mathcal{W}'_j$ *from the left contains the destination vertex* t_{i_z}.

Proof. From the definition of the ordering η of $\mathcal{M}'$, the demand pairs $(s_{i_1}, t_{i_1}), (s_{i_2}, t_{i_2}), \ldots, (s_{i_{M_j}}, t_{i_{M_j}})$ appear consecutively in this order in η. Moreover, from the construction of the NDP-Grid instance $(\hat{G}, \mathcal{M})$, every pair of destination vertices in $T(\mathcal{M}')$ is separated by at least one vertex. We add to $\mathcal{W}'_j$ all columns in which the vertices $t_{i_1}, t_{i_2}, \ldots, t_{i_{M_j}}$ lie. We also add to $\mathcal{W}'_j$ $i_1 - 1$ columns that lie to the left of the column of t_{i_1}, and $M' - i_{M'_j}$ columns that lie to the right of the column of $t_{i_{M_j}}$ in $\hat{K}'_j$. We make sure that no pair of columns in $\mathcal{W}'_j$ is consecutive—it is easy to see that there are enough columns to ensure that. $\square$

The proof of the following claim is identical to the proof of Claim 6.13 and is omitted here.

Claim 6.15. *There is a set* $\mathcal{P}^2 = \left\{P'_1, \ldots, P'_{M'}\right\}$ *of spaced-out paths in* G^b*, that are internally disjoint from* R*, such that for each* $1 \leq i \leq M'$*, path* P'_i *originates from vertex* x_i*, and for all* $1 \leq j \leq N_2$*, it contains the* i*th column of* $\mathcal{W}'_j$*; in particular it contains* t_i.

By combining the paths in sets $\mathcal{P}^1$ and $\mathcal{P}^2$, we obtain a new set $\mathcal{P}^* = \left\{P^*_1, \ldots, P^*_{M'}\right\}$ of spaced-out paths, such that for all $1 \leq i \leq M'$, path P^*_i contains s_i, x_i and t_i. By suitably truncating each such path, we obtain a collection of spaced-out paths routing all demand pairs in $\mathcal{M}'$.

6.4 From routing to partitioning: proof of Theorem 6.6

In this subsection we conclude the proof of Theorem 4.3, by proving Theorem 6.6. Let $\mathcal{M}^* \subseteq \mathcal{M}$ be the set of the demand pairs routed by the solution $\mathcal{P}^*$, and let $E^* \subseteq E$ be the set of all edges e, whose corresponding demand pair belongs to $\mathcal{M}^*$. Let $\tilde{G}' \subseteq \tilde{G}$ be the subgraph of $\tilde{G}$ induced by the edges in E^*. Notice that whenever two edges of $\tilde{G}$ belong to the same bundle, their corresponding demand pairs share a source or a destination. Since all paths in $\mathcal{P}^*$ are node-disjoint, all demand pairs in $\mathcal{M}^*$ have distinct sources and destinations, and so no two edges in E^* belong to the same bundle.

Note that, if $|\mathcal{P}^*| \leq 2^{64} h \log^3 M$, then we can return the solution $((W_1, \ldots, W_r), (E_1, \ldots, E_r))$, where $W_1 = V(\tilde{G})$ and $W_2 = W_3 = \cdots = W_r = \emptyset$; set E_1 contains an arbitrary subset of $\left\lceil \frac{|\mathcal{P}^*|}{2^{64} \log^3 M} \right\rceil \leq h$ edges of E^*, and all other sets E_i are empty. Since no two edges of E^* belong to the same bundle, we obtain a feasible solution to the (r,h)-GPwB problem instance of value $\Omega(|\mathcal{P}^*| / \log^3 M)$. Therefore, from now on, we assume that $|\mathcal{P}^*| > 2^{64} h \log^3 M$.

Our algorithm computes a solution to the (r,h)-GPwB instance $\mathcal{I}$ by repeatedly partitioning $\tilde{G}'$ into smaller and smaller subgraphs, by employing suitably defined balanced cuts.

Recall that, given a graph $\mathbf{H}$, a *cut* in $\mathbf{H}$ is a bi-partition (A, B) of its vertices. We denote by $E_{\mathbf{H}}(A, B)$ the set of all edges with one endpoint in A and another in B, and by $E_{\mathbf{H}}(A)$ and $E_{\mathbf{H}}(B)$ the sets of all edges with both endpoints in A and in B, respectively. Given a cut (A, B) of $\mathbf{H}$, the *value* of the cut is $|E_{\mathbf{H}}(A, B)|$. We will omit the subscript $\mathbf{H}$ when clear from context.

Definition 6.16. Given a graph $\mathbf{H}$ and a parameter $0 < \rho < 1$, a cut (A, B) of $\mathbf{H}$ is called a ρ-edge-balanced cut if $|E(A)|, |E(B)| \geq \rho \cdot |E(\mathbf{H})|$.

The following theorem is central to the proof of Theorem 6.6.

Theorem 6.17. *There is an efficient algorithm, that, given a vertex-induced subgraph* $\mathbf{H}$ *of* $\tilde{G}'$ *with* $|E(\mathbf{H})| > 2^{64} h \log^3 M$*, computes a* $1/32$*-edge-balanced cut of* $\mathbf{H}$*, of value at most* $\frac{|E(\mathbf{H})|}{64 \log M}$.

We prove Theorem 6.17 below, after we complete the proof of Theorem 6.6 using it. Our algorithm maintains a collection $\mathcal{G}$ of disjoint vertex-induced subgraphs of $\tilde{G}'$, and consists of a number of phases. The input to the first phase is the collection $\mathcal{G}$ containing a single graph—the graph $\tilde{G}'$. The algorithm continues as long as $\mathcal{G}$ contains a graph $\mathbf{H} \in \mathcal{G}$ with $|E(\mathbf{H})| > 2^{64} \cdot h \log^3 M$; if no such graph $\mathbf{H}$ exists, then the algorithm terminates. Each phase is executed as follows. We process every graph $\mathbf{H} \in \mathcal{G}$ with $|E(\mathbf{H})| > 2^{64} \cdot h \log^3 M$ one-by-one. When graph $\mathbf{H}$ is processed, we apply Theorem 6.17 to it, obtaining a $1/32$-edge-balanced cut (A, B) of $\mathbf{H}$, of value at most $\frac{|E(\mathbf{H})|}{64 \log M}$. We then remove $\mathbf{H}$ from $\mathcal{G}$, and add $\mathbf{H}[A]$ and $\mathbf{H}[B]$ to $\mathcal{G}$ instead. This completes the description of the algorithm. We use the following claim to analyze it.

Claim 6.18. *Let* $\mathcal{G}'$ *be the final set of disjoint subgraphs of* $\tilde{G}'$ *obtained at the end of the algorithm. Then* $\sum_{\mathbf{H} \in \mathcal{G}'} |E(\mathbf{H})| \geq \Omega(|E(\tilde{G}')|)$*, and* $|\mathcal{G}'| \leq r$.

Proof. We construct a binary partitioning tree τ of graph $\tilde{G}'$, that simulates the graph partitions computed by the algorithm. For every subgraph $\mathbf{H} \subseteq \tilde{G}'$ that belonged to $\mathcal{G}$ over the course of the algorithm, tree τ contains a node $v(\mathbf{H})$. The root of the tree is the node $v(\tilde{G}')$. If, over the course of our algorithm, we have partitioned the graph $\mathbf{H}$ into two disjoint vertex-induced subgraphs $\mathbf{H}'$ and $\mathbf{H}''$, then we add edges $(v(\mathbf{H}), v(\mathbf{H}'))$ and $(v(\mathbf{H}), v(\mathbf{H}''))$ to the tree, and $v(\mathbf{H}'), v(\mathbf{H}'')$ become the children of $v(\mathbf{H})$ in τ.

The *level* of a node $v(\mathbf{H})$ in the tree is the length of the path connecting $v(\mathbf{H})$ to the root of the tree; so the root of the tree is at level 0. The *depth* of the tree, that we denote by Δ, is the length of the longest leaf-to-root path in the tree. Since the cuts computed over the course of the algorithm are $1/32-$edge-balanced, $\Delta \leq \frac{\log M}{\log(32/31)}$. Consider now some level $0 \leq i \leq \Delta$, and let V_i be the set of all nodes of the tree τ lying at level i. Let V'_i be a set of nodes consisting of all nodes of V_i, and all nodes that lie at levels $0, \ldots, (i-1)$ that are leaves of the tree. Let $\hat{E}_i = \bigcup_{v(\mathbf{H}) \in V'_i} E(\mathbf{H})$ be the set of all edges contained in all subgraphs $\mathbf{H}$ of $\tilde{G}'$, whose corresponding node $v(\mathbf{H})$ lies in V'_i. Finally, let $m_i = |\hat{E}_i|$. Then $m_0 = |E(\tilde{G}')|$, and for all $1 \leq i \leq \Delta$, the number of edges discarded over the course of phase i is $m_{i-1} - m_i \leq m_{i-1}/(64 \log M) \leq m_0/(64 \log M)$—this is since, whenever we partition a graph $\mathbf{H}$ into two subgraphs, we lose at most $|E(\mathbf{H})|/(64 \log M)$ of its edges. Overall, we get that:

$$m_0 - m_\Delta \leq \frac{\Delta m_0}{64 \log M} \leq \frac{\log M}{\log 32/31} \cdot \frac{m_0}{64 \log M} \leq \frac{m_0}{2},$$

and so $\sum_{\mathbf{H} \in \mathcal{G}'} |E(\mathbf{H})| = m_\Delta \geq m_0/2$. This finishes the proof of the first assertion. We now turn to prove the second assertion. Recall that no two edges of E^* may belong to the same bundle. Since $h = \beta^*(\mathcal{I})/r = \left(\sum_{v \in V_1} \beta(v)\right)/r$, for any subset $E' \subseteq E^*$ of edges, $|E'| \leq \sum_{v \in V_1} \beta(v) \leq hr$ must hold, and in particular, $|E^*| \leq hr$. It is now enough to prove that for every leaf vertex $v(\mathbf{H})$ of τ, $|E(\mathbf{H})| \geq h$ holds—since all graphs in $\mathcal{G}'$ are mutually disjoint, and each such graph corresponds to a distinct leaf of τ, this would imply that $|\mathcal{G}'| \leq r$.

Consider now some leaf vertex $v(\mathbf{H})$ of τ, and let $v(\mathbf{H}')$ be its parent. Then $|E(\mathbf{H}')| \geq 2^{64} h \log^3 M$, and, since the partition of $\mathbf{H}'$ that we have computed was $1/32$-balanced, $|E(\mathbf{H})| \geq |E(\mathbf{H}')|/32 \geq h$. We conclude that $|\mathcal{G}'| \leq r$. □

We are now ready to define the solution $((W_1,\ldots,W_r),(E_1,\ldots,E_r))$ to the (r,h)-GPwB problem instance $\mathcal{I}$. Let $\mathcal{G}'$ be the set of the subgraphs of $\tilde{G}'$ obtained at the end of our algorithm, and denote $\mathcal{G}' = \{\mathbf{H}_1,\mathbf{H}_2,\ldots,\mathbf{H}_z\}$. Recall that from Claim 6.18, $z \leq r$. For $1 \leq i \leq z$, we let $W_i = V(\mathbf{H}_i)$. If $|E(\mathbf{H}_i)| \leq h$, then we let $E_i = E(\mathbf{H}_i)$; otherwise, we let E_i contain any subset of h edges of $E(\mathbf{H}_i)$. Since $|E(\mathbf{H}_i)| \leq 2^{64} h \log^3 M$, in either case, $|E_i| \geq \Omega(|E(\mathbf{H}_i)|/\log^3 M)$. For $i > z$, we set $W_i = \emptyset$ and $E_i = \emptyset$. Since, as observed before, no pair of edges of E^* belongs to the same bundle, it is immediate to verify that we obtain a feasible solution to the (r,h)-GPwB problem instance. The value of the solution is:

$$\sum_{i=1}^{r} |E_i| \geq \sum_{i=1}^{r} \Omega(|E(\mathbf{H}_i)|/\log^3 M) = \Omega(|E(\tilde{G}')|/\log^3 M) = \Omega(|\mathcal{P}^*|/\log^3 M),$$

from Claim 6.18. In order to complete the proof of Theorem 6.6, it now remains to prove Theorem 6.17.

Proof of Theorem 6.17. Let $\mathbf{H}$ be a vertex-induced subgraph of $\tilde{G}'$ with $|E(\mathbf{H})| > 2^{64} h \log^3 M$. Our proof consists of two parts. First, we show an efficient algorithm to compute a drawing of $\mathbf{H}$ with relatively few crossings. Next, we show how to exploit this drawing in order to compute a $1/32$-edge-balanced cut of $\mathbf{H}$ of small value. We start by defining a drawing of a given graph in the plane and the crossings in this drawing.

Definition 6.19. A *drawing* of a given graph $\mathbf{H}'$ in the plane is a mapping, in which every vertex of $\mathbf{H}$ is mapped to a point in the plane, and every edge to a continuous curve connecting the images of its endpoints, such that whenever a point x belongs to three such curves, x is an endpoint of all three curves; no curve intersects itself; and no curve contains an image of any vertex other than its endpoints. A *crossing* in such a drawing is a point contained in the images of two edges.

Lemma 6.20. *There is an efficient algorithm that, given a solution $\mathcal{P}^*$ to the* NDP-Grid *problem instance $\hat{\mathcal{I}}$, and a vertex-induced subgraph $\mathbf{H} \subseteq \tilde{G}'$, computes a drawing of $\mathbf{H}$ with at most $2048 \cdot |E(\mathbf{H})| \lceil h \log M \rceil$ crossings.*

Proof. We consider the natural embedding of the grid $\hat{G}$ into the plane. We will use this drawing of $\hat{G}$ in order to guide our drawing of the graph $\mathbf{H}$. Notice that this embedding of $\hat{G}$ naturally defines a drawing of every path $P \in \mathcal{P}^*$, obtained by taking the union of the images of the edges of P. We will map vertices of $\mathbf{H}$ to points that lie inside some faces defined by the cells of the grid. Consider a vertex $v_i \in V(\mathbf{H}) \cap V_1$, and let K_i be the block representing this vertex. Let κ_i be any cell of the grid $\hat{G}$ that has a vertex of K_i on its boundary. We map the vertex v_i to a point p_i lying in the middle of the cell κ_i (it is sufficient that p_i is far enough from the boundaries of the cell). For every vertex $v'_j \in V(\mathbf{H}) \cap V_2$, we select a cell κ'_j whose boundary contains a vertex of the corresponding block K'_j, and map v'_j to a point p'_j lying in the middle of κ'_j similarly.

Next, we define the drawings of the edges of $E(\mathbf{H})$. Consider any such edge $e=(v_i,v'_j)\in E(\mathbf{H})$, with $v_i\in V_1$ and $v'_j\in V_2$, and let $(s,t)\in\mathcal{M}^*$ be its corresponding demand pair. Let K_i and K'_j be the blocks containing s and t respectively, and let $P\in\mathcal{P}^*$ be the path routing the demand pair (s,t) in our solution to the NDP-Grid problem instance $\hat{\mathcal{I}}$. The drawing of the edge e is a concatenation of the following three segments: (i) the image of the path P, that we refer to as a type-1 segment; (ii) a straight line connecting s to the image of v_i, that we refer to as a type-2 segment; and (iii) a straight line connecting t to the image of v'_j, that we refer to as a type-3 segment. If the resulting curve has any self-loops, then we delete them.

We now bound the number of crossings in the resulting drawing of $\mathbf{H}$. Since the paths in $\mathcal{P}^*$ are node-disjoint, whenever the images of two edges e and e' cross, the crossing must be between the type-1 segment of e, and either the type-2 or the type-3 segment of e' (or the other way around).

Consider now some edge $e=(v_i,v'_j)\in E(\mathbf{H})$ with $v_i\in V_1$ and $v'_j\in V_2$, and let K_i and K'_j be the blocks representing v_i and v'_j respectively. Let $(s,t)\in\mathcal{M}^*$ be the demand pair corresponding to e. Assume that a type-1 segment of some edge e' crosses a type-2 segment of e. This can only happen if the path $P'\in\mathcal{P}^*$ routing the demand pair (s',t') corresponding to the edge e' contains a vertex of K_i. Since $|V(K_i)|\le 1024\cdot\lceil h\log M\rceil$, at most $1024\cdot\lceil h\log M\rceil$ type-1 segments of other edges may cross the type-2 segment of e. The same accounting applies to the type-3 segment of e. Overall, the number of crossings in the above drawing is bounded by:

$$\sum_{e\in E(\mathbf{H})} 2\cdot 1024\cdot\lceil h\log M\rceil = 2048\cdot|E(\mathbf{H})|\lceil h\log M\rceil\,. \qquad \square$$

Next, we show an efficient algorithm that computes a small balanced partition of any given graph $\mathbf{H}'$, as long as its maximum vertex degree is suitably bounded, and we are given a drawing of $\mathbf{H}'$ with a small number of crossings.

Lemma 6.21. *There is an efficient algorithm that, given any graph $\mathbf{H}$ with $|E(\mathbf{H})|=m$ and maximum vertex degree at most d, and a drawing φ of $\mathbf{H}$ with at most $\mathrm{cr}\le md\alpha$ crossings for some $\alpha>1$, such that $m>2^{20}d\alpha$, computes a $1/32$-edge-balanced cut (A,B) of $\mathbf{H}$ of value $|E(A,B)|\le 64\sqrt{8md\alpha}$.*

Before we complete the proof of the lemma, we show that the proof of Theorem 6.17 follows from it. Let $m=|E(\mathbf{H})|$. Note that the maximum vertex degree in graph $\mathbf{H}$ is bounded by $\max_{v\in V_1\cup V_2}\{\beta(v)\}$, as $E(\mathbf{H})$ cannot contain two edges that belong to the same bundle. From the definition of valid instances, $h\ge\beta(v)$ for all $v\in V_1\cup V_2$, and so the maximum vertex degree in $\mathbf{H}$ is bounded by $d=h$. From Lemma 6.20, the drawing φ of $\mathbf{H}$ has at most $2048|E(\mathbf{H})|\lceil h\log M\rceil\le 2^{12}md\log M$ crossings. Setting $\alpha=2^{12}\log M$, the number of crossings cr in φ is bounded by $md\alpha$. Moreover, since $m>2^{64}h\log^3 M$, we get that $m>2^{20}d\alpha$. We can now apply Lemma 6.21 to graph $\mathbf{H}$ to obtain a $1/32$-edge-balanced cut (A,B) with $|E(A,B)|\le 64\sqrt{8md\alpha}\le 64\sqrt{2^{15}mh\log M}\le 64\sqrt{m^2/(2^{49}\log^2 M)}\le\frac{|E(\mathbf{H})|}{64\log M}$, since we have assumed that $|E(\mathbf{H})|=m>2^{64}h\log^3 M$. It now remains to prove Lemma 6.21.

Proof of Lemma 6.21. We start by providing some intuition for the proof. In general, it is well known that a graph G that can be drawn in the plane with at most cr crossings has a balanced separator (a

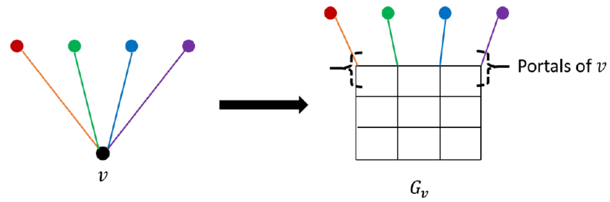

Figure 5: Grid Q_v obtained from v

vertex-cut), containing $O(\sqrt{|V(G)|+\mathrm{cr}})$ vertices. However, in our case, we need to compute an edge-cut, and since some vertex degrees may be quite large, using this bound and then deleting all edges incident to the vertices in the separator is not sufficient to us. In order to overcome this difficulty, we slightly transform our graph into a constant-degree graph, and then work with the transformed graph directly.

For each vertex $v \in V(\mathbf{H})$, we denote the degree of v in $\mathbf{H}$ by d_v. We assume without loss of generality that for all $v \in V(\mathbf{H})$, $d_v \geq 1$: otherwise, we can remove all isolated vertices from $\mathbf{H}$ and then apply our algorithm to compute a $1/32$-edge-balanced cut (A,B) in the remaining graph. At the end we can add the isolated vertices to A or B, while ensuring that the cut remains $1/32$-balanced, and without increasing its value.

We construct a new graph $\hat{\mathbf{H}}$ from graph $\mathbf{H}$ as follows. For every vertex $v \in V(\mathbf{H})$, we add a $(d_v \times d_v)$-grid Q_v to $\hat{\mathbf{H}}$, so that the resulting grids are mutually disjoint. We call the edges of the resulting grids *regular edges*. Let $e_1(v),\ldots,e_{d_v}(v)$ be the edges of $\mathbf{H}$ incident to v, indexed in the clockwise order of their entering the vertex v in the drawing φ of $\mathbf{H}$. We denote by $\Pi(v) = \{p_1(v),\ldots,p_{d_v}(v)\}$ the set of vertices on the top boundary of Q_v, where the vertices are indexed in the clock-wise order of their appearance on the boundary of Q_v. We refer to the vertices of $\Pi(v)$ as the *portals* of Q_v (see Figure 5). Let $\Pi = \bigcup_{v\in V(\mathbf{H})} \Pi(v)$ be the set of all portals. For every edge $e = (u,v) \in E(\mathbf{H})$, we add a new *special edge* to graph $\hat{\mathbf{H}}$, as follows. Assume that $e = e_i(v) = e_j(u)$. Then we add an edge $(p_i(v), p_j(u))$ to $\hat{\mathbf{H}}$. We think of this edge as the special edge representing e. This finishes the definition of the graph $\hat{\mathbf{H}}$. It is immediate to see that the drawing φ of $\mathbf{H}$ can be extended to a drawing φ' of $\hat{\mathbf{H}}$ without introducing any new crossings, that is, the number of crossings in φ' remains at most cr. Note that every portal vertex is incident to exactly one special edge, and the maximum vertex degree in $\hat{\mathbf{H}}$ is 4. We will use the following bound on $|V(\hat{\mathbf{H}})|$:

Observation 6.22. $|V(\hat{\mathbf{H}})| \leq 2md$.

Proof. Clearly, $|V(\hat{\mathbf{H}})| = \sum_{v\in V(\mathbf{H})} d_v^2 \leq \sum_{v\in V(\mathbf{H})} d \cdot d_v \leq 2md$. □

Let $\hat{\mathbf{H}}'$ be the graph obtained from $\hat{\mathbf{H}}$ by replacing every intersection point in the drawing φ' of $\hat{\mathbf{H}}$ with a vertex. Then $\hat{\mathbf{H}}'$ is a planar graph with $|V(\hat{\mathbf{H}}')| \leq |V(\hat{\mathbf{H}})| + \mathrm{cr} \leq 2md + md\alpha \leq 4md\alpha$, as $\alpha \geq 1$. We assign weights to the vertices of $\hat{\mathbf{H}}'$ as follows: every vertex of Π is assigned the weight 1, and every other vertex is assigned the weight 0. Note that the weight of a vertex is exactly the number of special edges incident to it, and the total weight of all vertices is $W = |\Pi| = 2m$. We will use the following version of the planar separator theorem [52, 40, 2].

Theorem 6.23 ([40]). *There is an efficient algorithm, that, given a planar graph $G=(V,E)$ with n vertices, and an assignment $w: V \to R^+$ of non-negative weights to the vertices of G, with $\sum_{v\in V} w(v) = W$, computes a partition (A,X,B) of $V(G)$, such that:*

- *no edge connecting a vertex of A to a vertex of B exists in G;*
- *$\sum_{v\in A} w(v), \sum_{v\in B} w(v) \le 2W/3$; and*
- *$|X| \le 2\sqrt{2n}$.*

We apply Theorem 6.23 to graph $\hat{\mathbf{H}}'$, to obtain a partition (A,X,B) of $V(\hat{\mathbf{H}}')$, with $|X| \le 2\sqrt{2|V(\hat{\mathbf{H}}')|} \le 2\sqrt{8md\alpha}$. Since $W = \sum_{v\in V(\hat{\mathbf{H}}')} w(v) = 2m$, we get that $|A\cap\Pi| = \sum_{v\in A} w(v) \le 2W/3 \le 4m/3$, and similarly $|B\cap\Pi| \le 4m/3$. Assume without loss of generality that $|A\cap\Pi| \le |B\cap\Pi|$. We obtain a bi-partition (A',B') of $V(\hat{\mathbf{H}}')$ by setting $A' = A\cup X$ and $B' = B$. Since $|X| \le 2\sqrt{8md\alpha} \le m/3$ (as $m > 2^{20}d\alpha$), we are guaranteed that $|A'\cap\Pi|, |B'\cap\Pi| \le 4m/3$ holds. Moreover, as all vertex degrees in $\hat{\mathbf{H}}'$ are at most 4, $|E_{\hat{\mathbf{H}}'}(A',B')| \le 4|X| \le 8\sqrt{8md\alpha}$.

Note that cut (A',B') in graph $\hat{\mathbf{H}}'$ naturally defines a cut (A'',B'') in graph $\hat{\mathbf{H}}$, where a vertex $v \in V(\hat{\mathbf{H}})$ is added to A'' if it lies in the set A', and it is added to B'' otherwise. It is easy to verify that $|A''\cap\Pi|, |B''\cap\Pi| \le 4m/3$ still holds. Moreover, the number of edges in the cut may only decrease, that is, $|E_{\hat{\mathbf{H}}}(A'',B'')| \le |E_{\hat{\mathbf{H}}'}(A',B')| \le 8\sqrt{8md\alpha}$. For convenience, we will denote the cut (A'',B'') by (A',B') from now on, and we will only continue working with this cut.

Unfortunately, the cut (A',B') of $\hat{H}$ does not directly translate into a balanced cut in $\mathbf{H}$, since for some vertices $v \in V(\mathbf{H})$, the corresponding grid Q_v may be split between A' and B'. We now show how to overcome this difficulty, by moving each such grid entirely to one of the two sides. Before we proceed, we state a simple fact about grid graphs.

Observation 6.24. Let $z > 1$ be an integer, and let Q be the $(z\times z)$-grid. Let U be the set of vertices lying on the top row of Q, and let (X,Y) be a bi-partition of $V(Q)$. Then $|E_Q(X,Y)| \ge \min\{|U\cap X|, |U\cap Y|\}$.

Proof. It is easy to verify that for any bi-partition (X',Y') of U into two disjoint subsets, there is a set $\mathcal{P}$ of $\min\{|X'|,|Y'|\}$ node-disjoint paths in Q connecting vertices of X' to vertices of Y'. The observation follows from the maximum flow–minimum cut theorem. $\square$

We say that a vertex $v\in V(\mathbf{H})$ is *split* by the cut (A',B') if $V(Q_v)\cap A'$ and $V(Q_v)\cap B' \ne \emptyset$. We say that it is *split evenly* iff if $|\Pi(v)\cap A'|, |\Pi(v)\cap B'| \ge d_v/8$; otherwise we say that it is *split unevenly*. We modify the cut (A',B') in the following two steps, to ensure that no vertex of $V(\mathbf{H})$ remains split.

Step 1 [Unevenly split vertices]. We process each vertex $v \in V(\mathbf{H})$ that is unevenly split one-by-one. Consider any such vertex v. If $|\Pi(v)\cap A'| > |\Pi(v)\cap B'|$, then we move all vertices of Q_v to A'; otherwise we move all vertices of Q_v to B'. Assume without loss of generality that the former happens. Notice that from Observation 6.24, $|E(A',B')|$ does not increase, since $E(Q_v)$ contributed at least $|\Pi(v)\cap B'|$

regular edges to the cut before the current iteration. Moreover, $|\Pi(v) \cap A'|$ increases by the factor of at most $8/7$. Therefore, at the end of this procedure, once all unevenly split vertices of $\mathbf{H}$ are processed, $|A' \cap \Pi|, |B' \cap \Pi| \leq \frac{8}{7} \cdot \frac{4}{3} m = \frac{32}{21} m$ and $|E(A', B')| \leq 8\sqrt{8md\alpha}$.

Step 2 [Evenly split vertices]. In this step, we process each vertex $v \in V(\mathbf{H})$ that is evenly split one-by-one. Consider an iteration where some such vertex $v \in V(\mathbf{H})$ is processed. If $|A' \cap \Pi| \leq |B' \cap \Pi|$, then we move all vertices of Q_v to A'; otherwise we move them to B'. Assume without loss of generality that the former happened. Then before the current iteration $|A' \cap \Pi| \leq |\Pi|/2 \leq m$, and, since $|\Pi(v)| \leq d < m/21$, $|A' \cap \Pi| \leq \frac{32}{21} m$, while $|B' \cap \Pi| \leq \frac{32}{21} m$ as before. Moreover, from Observation 6.24, before the current iteration, the regular edges of Q_v contributed at least $d(v)/8$ edges to $E(A', B')$, and after the current iteration, no regular edges of Q_v contribute to the cut, but we may have added up to $d(v)$ new special edges to it. Therefore, after all vertices of $\mathbf{H}$ that are evenly split are processed, $|E(A', B')|$ grows by the factor of at most 8, and remains at most $64\sqrt{8md\alpha}$.

We are now ready to define the final cut (A^*, B^*) in graph $\mathbf{H}$. We let A^* contain all vertices $v \in V(\mathbf{H})$ with $V(Q_v) \subseteq A'$, and we let B^* contain all remaining vertices of $V(\mathbf{H})$. Clearly, $|E_{\mathbf{H}}(A^*, B^*)| \leq |E_{\hat{\mathbf{H}}'}(A', B')| \leq 64\sqrt{8md\alpha}$. It remains to show that $|E_{\mathbf{H}}(A^*)|, |E_{\mathbf{H}}(B^*)| \geq |E(\mathbf{H})|/32$. We show that $|E_{\mathbf{H}}(A^*)| \geq |E(\mathbf{H})|/32$; the proof that $|E_{\mathbf{H}}(B^*)| \geq |E(\mathbf{H})|/32$ is symmetric. Observe that $\sum_{v \in B^*} d_v = |B' \cap \Pi| \leq \frac{32m}{21}$, while $\sum_{v \in V(\mathbf{H})} d_v = 2m$. Therefore, $\sum_{v \in A^*} d_v \geq 2m - \frac{32m}{21} = \frac{10m}{21}$. But $|E_{\mathbf{H}}(A^*, B^*)| \leq 64\sqrt{8md\alpha} \leq 64\sqrt{m^2/2^{17}} < m/4$ (since $m > 2^{20} d\alpha$). Therefore,

$$|E_{\mathbf{H}}(A^*)| = \frac{\sum_{v \in A^*} d_v - |E_{\mathbf{H}}(A^*, B^*)|}{2} \geq \frac{5m}{21} - \frac{m}{8} \geq \frac{m}{32}.$$

□

□

7 Hardness of NDP and EDP on wall graphs

In this section we extend our results to NDP and EDP on wall graphs, completing the proofs of Theorem 1.1 and Theorem 1.2. We first prove hardness of NDP-Wall, and show later how to extend it to EDP-Wall. Let $\hat{G} = G^{\ell,h}$ be a grid of length ℓ and height h, where $\ell > 0$ is an even integer, and $h > 0$. We denote by $\hat{G}'$ the wall corresponding to $\hat{G}$, as defined in Section 2. We prove the following analogue of Theorem 4.3.

Theorem 7.1. *There is a constant $c^* > 0$, and there is an algorithm, that, given a valid and a (d, d', b)-regular instance $\mathcal{I} = (\tilde{G}, \mathcal{U}_1, \mathcal{U}_2, h, r)$ of* (r,h)-GPwB*, for some $d, d', b > 0$, with $|V(\tilde{G})| = N$ and $|E(\tilde{G})| = M$, in time* poly(NM) *constructs an instance $\hat{\mathcal{I}}' = (\hat{G}', \mathcal{M})$ of* NDP-Wall *with $|V(\hat{G}')| = O(M^4 \log^2 M)$, such that the following hold:*

- *if $\mathcal{I}$ has a perfect solution (of value $\beta^* = \beta^*(\mathcal{I})$), then instance $\hat{\mathcal{I}}'$ has a solution $\mathcal{P}'$ that routes at least $\frac{\beta^*}{c^* \log^2 M}$ demand pairs via node-disjoint paths; and*
- *there is an algorithm with running time* $\text{poly}(NM)$*, that, given a solution $\mathcal{P}^*$ to the* NDP-Wall *problem instance $\hat{\mathcal{I}}'$, constructs a solution to the* (r,h)-GPwB *instance $\mathcal{I}$, of value at least $\frac{|\mathcal{P}^*|}{c^* \cdot \log^3 M}$.*

Notice that plugging Theorem 7.1 into the hardness-of-approximation proof instead of Theorem 4.3, we extend the hardness result to the NDP problem on wall graphs and complete the proof of Theorem 1.1.

Proof of Theorem 7.1. Let $\hat{\mathcal{I}} = (\hat{G}, \mathcal{M})$ be the instance of NDP-Grid constructed in Theorem 4.3. In order to obtain an instance $\hat{\mathcal{I}}'$ of NDP-Wall, we replace the grid $\hat{G}$ with the corresponding wall $\hat{G}'$ as described above; the set of the demand pairs remains unchanged. We now prove the two assertions about the resulting instance $\hat{\mathcal{I}}'$, starting from the second one.

Suppose we are given a solution $\mathcal{P}^*$ to the NDP-Wall problem instance $\hat{\mathcal{I}}'$. Since $\hat{G}' \subseteq \hat{G}$, and the set of demand pairs in instances $\hat{\mathcal{I}}$ and $\hat{\mathcal{I}}'$ is the same, $\mathcal{P}^*$ is also a feasible solution to the NDP-Grid problem instance $\hat{\mathcal{I}}$, and so we can use the efficient algorithm from Theorem 4.3 to construct a solution to the (r,h)-GPwB instance $\mathcal{I}$, of value at least $\frac{|\mathcal{P}^*|}{c^* \cdot \log^3 M}$.

It now remains to prove the first assertion. Assume that $\mathcal{I}$ has a perfect solution. From Theorem 4.3, instance $\hat{\mathcal{I}}$ of NDP-Grid has a solution $\mathcal{P}$ that routes at least $\frac{\beta^*}{c^* \log^2 M}$ demand pairs via paths that are spaced-out. It is now enough to show that there is a solution of value $\frac{\beta^*}{c^* \log^2 M}$ to the corresponding instance $\hat{\mathcal{I}}'$ of NDP-Wall.

Consider the spaced-out set $\mathcal{P}$ of paths in $\hat{G}$. Recall that for every pair P, P' of paths, $d(V(P), V(P')) \geq 2$, and all paths in $\mathcal{P}$ are internally disjoint from the boundaries of the grid $\hat{G}$. For each path $P \in \mathcal{P}$, we will slightly modify P to obtain a new path P' contained in the wall $\hat{G}'$, so that the resulting set $\mathcal{P}' = \{P' \mid P \in \mathcal{P}\}$ of paths is node-disjoint.

For all $1 \leq i < \ell$, $1 \leq j \leq \ell$, let e_i^j denote the ith edge from the top lying in column W_j of the grid $\hat{G}$, so that $e_j^i = (v(i,j), v(i+1,j))$. Let $E^* = E(\hat{G}) \setminus E(\hat{G}')$ be the set of edges that were deleted from the grid $\hat{G}$ when constructing the wall $\hat{G}'$. We call the edges of E^* *bad edges*. Notice that only vertical edges may be bad, and, if $e_i^j \in E(W_j)$ is a bad edge, for $1 < j < \ell$, then e_i^{j+1} is a good edge. Consider some bad edge $e_i^j = (v(i,j), v(i+1,j))$, such that $1 < j < \ell$, so e_i^j does not lie on the boundary of $\hat{G}$. Let Q_i^j be the path $(v(i,j), v(i,j+1), v(i+1,j+1), v(i+1,j))$. Clearly, path Q_i^j is contained in the wall $\hat{G}'$. For every path $P \in \mathcal{P}$, we obtain the new path P' by replacing every bad edge $e_i^j \in P$ with the corresponding path Q_i^j. It is easy to verify that P' is a path with the same endpoints as P, and that it is contained in the wall $\hat{G}'$. Moreover, since the paths in $\mathcal{P}$ are spaced-out, the paths in the resulting set $\mathcal{P}' = \{P' \mid P \in \mathcal{P}\}$ are node-disjoint. $\square$

This completes the proof of Theorem 1.1. In order to prove Theorem 1.2, we show an approximation-preserving reduction from NDP-Wall to EDP-Wall.

Claim 7.2. *Let $\mathcal{I} = (G, \mathcal{M})$ be an instance of* NDP-Wall*, and let $\mathcal{I}'$ be the instance of* EDP-Wall *consisting* *of the same graph G and the same set $\mathcal{M}$ of demand pairs. Let* OPT *and* OPT' *be the optimal solution values for* $\mathcal{I}$ *and* $\mathcal{I}'$*, respectively. Then* $\mathsf{OPT}' \geq \mathsf{OPT}$*, and there is an efficient algorithm, that, given any solution* $\mathcal{P}'$ *to instance* $\mathcal{I}'$ *of* EDP-Wall*, computes a solution* $\mathcal{P}$ *to instance* $\mathcal{I}$ *of* NDP-Wall *of value* $\Omega(|\mathcal{P}'|)$*.*

Notice that, from Claim 7.2, if there is an α-approximation algorithm for EDP-Wall with running time $f(n)$, for $\alpha > 1$ that may be a function of the graph size n, then there is an $O(\alpha)$-approximation algorithm for NDP-Wall with running time $f(n) + \mathrm{poly}(n)$. Therefore, the proof of Claim 7.2 will complete the proof of Theorem 1.2

Proof. The assertion that $\mathsf{OPT}' \geq \mathsf{OPT}$ is immediate, as any set $\mathcal{P}$ of node-disjoint paths in the wall G is also a set of edge-disjoint paths.

Assume now that we are given a set $\mathcal{P}'$ of edge-disjoint paths in G. We show an efficient algorithm to compute a subset $\mathcal{P} \subseteq \mathcal{P}'$ of $\Omega(|\mathcal{P}'|)$ paths that are node-disjoint. Since the maximum vertex degree in G is 3, the only way for two paths $P, P' \in \mathcal{P}'$ to share a vertex x is when x is an endpoint of at least one of these two paths. If x is an endpoint of P, and $x \in V(P')$, then we say that P has a conflict with P'.

We construct a directed graph H, whose vertex set is $\{v_P \mid P \in \mathcal{P}'\}$, and there is an edge $(v_P, v_{P'})$ iff P has a conflict with P'. It is immediate to verify that the maximum out-degree of any vertex in H is at most 4, as each of the two endpoints of a path P may be shared by at most two additional paths. Therefore, every subgraph $H' \subseteq H$ of H contains a vertex of total degree at most 8. We construct a set U of vertices of H, such that no two vertices of U are connected by an edge, using a standard greedy algorithm: while $H \neq \emptyset$, select a vertex $v \in V(H)$ with total degree at most 8 and add it to U; remove v and all its neighbors from H. It is easy to verify that at the end of the algorithm, $|U| = \Omega(|V(H)|) = \Omega(|\mathcal{P}'|)$, and no pair of vertices in U is connected by an edge. Let $\mathcal{P} = \{P \mid v_P \in U\}$. Then the paths in $\mathcal{P}$ are node-disjoint, and $|\mathcal{P}| = \Omega(|\mathcal{P}'|)$. □

Appendix

A Proof of Theorem 2.6

Suppose G is a YES-INSTANCE, and let χ be a valid coloring of $V(G)$. Let $\pi_1, \ldots, \pi_6$ be 6 different permutations of $\{r, g, b\}$. For each $1 \leq i \leq 6$, permutation π_i defines a valid coloring χ_i of G: for every vertex $v \in V(G)$, if v is assigned a color $c \in \{\mathcal{C}\}$ by χ, then χ_i assigns the color $\pi_i(c)$ to v. Notice that for each vertex v and for each color $c \in \mathcal{C}$, there are exactly two indices $i \in \{1, \ldots, 6\}$, such that χ_i assigns the color c to v. Notice also that for each edge (u, v), if $c, c' \in \mathcal{C}$ is any pair of distinct colors, then there is exactly one index $i \in \{1, \ldots, 6\}$, such that u is assigned the color c and v is assigned the color c' by χ_i.

Let B be the set of all vectors of length ℓ, whose entries belong to $\{1, \ldots, 6\}$, so that $|B| = 6^\ell$. For each such vector $b \in B$, we define a perfect global assignment f_b of answers to the queries, as follows. Let $Q \in \mathcal{Q}^E$ be a query to the edge-player, and assume that $Q = (e_1, \ldots, e_\ell)$. Fix some index $1 \leq j \leq \ell$, and assume that $e_j = (v_j, u_j)$. Assume that $b_j = z$, for some $1 \leq z \leq 6$. We assign to v_j the color $\chi_z(v_j)$, and we assign to u_j the color $\chi_z(u_j)$. Since χ_z is a valid coloring of $V(G)$, the two colors are distinct. This defines an answer $A \in \mathcal{A}^E$ to the query Q, that determines $f_b(Q)$.

Consider now some query $Q' \in \mathcal{Q}^V$ to the vertex-player, and assume that $Q' = (v_1, \ldots, v_\ell)$. Fix some index $1 \le j \le \ell$, and assume that $b_j = z$, for some $1 \le z \le 6$. We assign to v_j the color $\chi_z(v_j)$. This defines an answer $A' \in \mathcal{A}^V$ to the query Q', that determines $f_b(Q')$. Notice that for each $1 \le j \le \ell$, the answers that we choose for the jth coordinate of each query are consistent with the valid coloring χ_{b_j} of G. Therefore, it is immediate to verify that for each $b \in B$, f_b is a perfect global assignment.

We now fix some query $Q \in \mathcal{Q}^E$ of the edge-prover, and some answer $A \in \mathcal{A}^E$ to it. Assume that $Q = (e_1, \ldots, e_\ell)$, where for $1 \le j \le \ell$, $e_j = (v_j, u_j)$. Let c_j, c'_j are the assignments to v_j and u_j given by the jth coordinate of A, so that $c_j \ne c'_j$. Recall that there is exactly one index $z_j \in \{1, \ldots, 6\}$, such that χ_{z_j} assigns the color c_j to v_j and the color c'_j to u_j. Let $b^* \in B$ be the vector, where for $1 \le j \le \ell$, $b^*_j = z_j$. Then $f_{b^*}(Q) = A$, and for all $b \ne b^*$, $f_b(Q) \ne A$.

Finally, fix some query $Q' \in \mathcal{Q}^V$ of the vertex-prover, and some answer $A' \in \mathcal{A}^V$ to it. Let $Q' = (v_1, \ldots, v_\ell)$. Assume that for each $1 \le j \le \ell$, the jth coordinate of A' contains the color c_j. Recall that there are exactly two indices $z \in \{1, \ldots, 6\}$, such that χ_z assigns the color c_j to v_j. Denote this set of two indices by $Z_j \subseteq \{1, \ldots, 6\}$. Consider now some vector $b \in B$. If, for all $1 \le j \le \ell$, $b_j \in Z_j$, then $f_b(Q') = A'$; otherwise, $f_b(Q') \ne A'$. Therefore, the total number of vectors $b \in B$, for which $f_b(Q') = A'$ is exactly 2^ℓ.

B Proof of Observation 6.4

We use the standard greedy algorithm. Start with $\mathcal{S}' = \emptyset$, and then iterate. In each iteration i, we consider the current partial solution $\mathcal{S}'$, and let J_i be the set of all elements j, such that $j \notin \bigcup_{S \in \mathcal{S}'} S$. We let $S_i \in \mathcal{S}$ be the set containing largest number of elements in J_i, add S_i to $\mathcal{S}'$, and continue to the next iteration. The algorithm terminates once $J_i = \emptyset$.

In order to analyze the algorithm, consider the ith iteration. Notice that, since every element belongs to at least m/t sets of $\mathcal{S}$, there must be at least one set $S \in \mathcal{S}$ that contains at least $|J_i|/t$ elements of J_i. Therefore, set S_i that the algorithm chooses covers at least $|J_i|/t$ elements of J_i, and $|J_{i+1}| \le (1 - 1/t)|J_i|$. Overall, we get that for all i, $|J_i| \le (1 - 1/t)^{i-1} n$, and so both $|\mathcal{S}'|$, and the number of iterations are bounded by $O(t \log n)$.

C Proof of Claim 6.13

The goal of this section is to prove Claim 6.13. We show an efficient algorithm to construct a set $\mathcal{P}^1 = \{P_1, \ldots, P_{M'}\}$ of spaced-out paths, that originate at the vertices of X on R, and traverse the boxes $\hat{K}_j$ in a snake-like fashion (see Figure 4). We will ensure that for each $1 \le i \le M'$ and $1 \le j \le N_1$, the intersection of the path P_i with the box $\hat{K}_j$ is the ith column of $\mathcal{W}_j$, and that P_i contains the vertex x_i.

Fix some index $1 \le j \le N_1$, and consider the box $\hat{K}_j$. Let I'_j and I''_j denote the top and the bottom boundaries of $\hat{K}_j$, respectively. For each $1 \le i \le M'$, let W^i_j denote the ith column of $\mathcal{W}_j$, and let $x'(j,i)$

and $x''(j,i)$ denote the topmost and the bottommost vertices of W_j^i, respectively.

For convenience, for each $1 \le i \le M'$ we denote $x''(0,i) = x_i$, and we let $I_0'' \subseteq R$ be a subpath of R containing exactly $2M$ vertices, such that $x_1, \dots, x_{M'} \in I_0''$. The following claim is central to the proof.

Claim C.1. *There is an efficient algorithm to construct, for each $1 \le j \le N_1$, a set $\mathcal{P}_j = \left\{P_j^1, \dots, P_j^{M'}\right\}$ of paths, such that:*

- *For each $1 \le j \le N_1$ and $1 \le i \le M'$, path P_j^i connects $x''(j-1,i)$ to $x'(j,i)$;*
- *the set $\bigcup_{j=1}^{N_1} \mathcal{P}_j$ of paths is spaced-out; and*
- *all paths in $\bigcup_{j=1}^{N_1} \mathcal{P}_j$ are contained in G^t, and are internally disjoint from R.*

Notice that by combining the paths in $\bigcup_{j=1}^{N_1} \mathcal{P}_j$ with the set $\bigcup_{j=1}^{N_1} \mathcal{W}_j$ of columns of the boxes, we obtain the desired set $\mathcal{P}^1$ of paths, completing the proof of Claim 6.13. In the remainder of this section we focus on proving Claim C.1.

Recall that each box $\hat{K}_j$ is separated by at least $2M$ columns from every other such box, and from the left and right boundaries of $\hat{G}$. It is also separated by at least $4M$ rows from the top boundary of $\hat{G}$ and from the row R. We exploit this spacing in order to construct the paths, by utilizing a special structure called a *snake*, that we define next.

Given a set $\mathcal{L}$ of consecutive rows of $\hat{G}$ and a set $\mathcal{W}$ of consecutive columns of $\hat{G}$, we denote by $\Upsilon(\mathcal{L}, \mathcal{W})$ the subgraph of $\hat{G}$ spanned by the rows in $\mathcal{L}$ and the columns in $\mathcal{W}$; we refer to such a graph as a *corridor*. Let $\Upsilon = \Upsilon(\mathcal{L}, \mathcal{W})$ be any such corridor. Let L' and L'' be the top and the bottom row of $\mathcal{L}$ respectively, and let W' and W'' be the first and the last column of $\mathcal{W}$ respectively. The four paths $\Upsilon \cap L', \Upsilon \cap L'', \Upsilon \cap W'$ and $\Upsilon \cap W''$ are called the top, bottom, left and right boundaries of Υ respectively, and their union is called the *boundary* of Υ. The width of the corridor Υ is $w(\Upsilon) = \min\{|\mathcal{L}|, |\mathcal{W}|\}$. We say that two corridors Υ, Υ' are *internally disjoint* if every vertex $v \in \Upsilon \cap \Upsilon'$ belongs to the boundaries of both corridors. We say that two internally disjoint corridors Υ, Υ' are *neighbors* if $\Upsilon \cap \Upsilon' \neq \emptyset$. We are now ready to define snakes.

A *snake* $\mathcal{Y}$ of length z is a sequence $(\Upsilon_1, \Upsilon_2, \dots, \Upsilon_z)$ of z corridors that are pairwise internally disjoint, such that for all $1 \le z', z'' \le z$, the corridor $\Upsilon_{z'}$ is a neighbor of $\Upsilon_{z''}$ iff $|z' - z''| = 1$. The width of the snake is defined to be the minimum of two quantities: (i) $\min_{1 \le z' < z}\{|\Upsilon_{z'} \cap \Upsilon_{z'+1}|\}$; and (ii) $\min_{1 \le z' \le z}\{w(\Upsilon_{z'})\}$. Notice that, given a snake $\mathcal{Y}$, there is a unique simple cycle $\sigma(\mathcal{Y})$ contained in $\bigcup_{z'=1}^{z} \Upsilon_{z'}$, such that, if D denotes the disc on the plane whose boundary is $\sigma(\mathcal{Y})$, then every vertex of $\bigcup_{z'=1}^{z} \Upsilon_{z'}$ lies in D, while every other vertex of $\hat{G}$ lies outside D. We call $\sigma(\mathcal{Y})$ the *boundary* of $\mathcal{Y}$. We say that a vertex u belongs to a snake $\mathcal{Y}$, and denote $u \in \mathcal{Y}$, if $u \in \bigcup_{z'=1}^{z} \Upsilon_{z'}$. We use the following simple claim, whose proof can be found, e. g., in [19].

Claim C.2. *Let $\mathcal{Y} = (\Upsilon_1, \dots, \Upsilon_z)$ be a snake of width w, and let A and A' be two sets of vertices with $|A| = |A'| \le w - 2$, such that the vertices of A lie on a single boundary edge of Υ_1, and the vertices of A' lie on a single boundary edge of Υ_z. There is an efficient algorithm, that, given the snake $\mathcal{Y}$, and the sets A and A' of vertices as above, computes a set $\mathcal{Q}$ of node-disjoint paths contained in $\mathcal{Y}$, that connect every vertex of A to a distinct vertex of A'.*

Let L, L' be any pair of rows of G. Let $\mathcal{Q}$ be a set of node-disjoint paths connecting some set of vertices $B \subseteq L$ to $B' \subseteq L'$. We say that the paths in $\mathcal{Q}$ are *order-preserving* if the left-to-right ordering of their endpoints on L is same as that of their endpoints on L'.

Corollary C.3. *Let $\mathcal{Y} = (\Upsilon_1, \ldots, \Upsilon_z)$ be a snake of width w, and let B and B' be two sets of $r \leq \lfloor w/2 \rfloor - 1$ vertices each, such that the vertices of B lie on the bottom boundary edge of Υ_1, the vertices of B' lie on the top boundary edge of Υ_z and for every pair $v, v' \in B \cup B'$ of vertices, $d_{\hat{G}}(v, v') \geq 2$. There is an efficient algorithm, that, given the snake $\mathcal{Y}$, and the sets B and B' of vertices as above, computes a set $\hat{\mathcal{Q}}$ of spaced-out order-preserving paths contained in $\mathcal{Y}$.*

Proof. Let $B = \{b_1, b_2, \ldots, b_r\}$ and $B' = \{b'_1, b'_2, \ldots, b'_r\}$. Assume that the vertices in both sets are indexed according to their left-to-right ordering on their corresponding rows of the grid. Since set B does not contain a pair of neighboring vertices, we can augment it to a larger set A, by adding a vertex between every consecutive pair of vertices of B. In other words, we obtain a vertex set $A = \{a_1, \ldots, a_{2r-1}\}$, such that for all $1 \leq i \leq r$, $a_{2i-1} = b_i$, and the vertices of A are indexed according to their left-to-right ordering on the bottom boundary of Υ_1. Similarly, we can augment the set B' to a set $A' = \left\{a'_1, \ldots, a'_{2r-1}\right\}$ of vertices, such that for all $1 \leq i \leq r$, $a'_{2i-1} = b'_i$, and the vertices of A' are indexed according to their left-to-right ordering on the top boundary of Υ_z.

We apply Claim C.2 to the sets A, A' of vertices, obtaining a set $\mathcal{Q}$ of node-disjoint paths, that are contained in $\mathcal{Y}$, and connect every vertex of A to a distinct vertex of A'. For all $1 \leq i \leq r$, let Q_i be the path originating from a_i. We claim that $\mathcal{Q}$ is an order-preserving set of paths. Indeed, assume for contradiction that some path $Q_i \in \mathcal{Q}$ connects a_i to $a'_{i'}$, for $i \neq i'$. Notice that the path Q_i partitions the snake $\mathcal{Y}$ into two subgraphs: one containing $(i-1)$ vertices of A and $(i'-1)$ vertices of A'; and the other containing the remaining vertices of A and A' (excluding the endpoints of Q_i). Since $i \neq i'$, there must be a path $Q_{i''} \in \mathcal{Q}$ intersecting the path Q_i, a contradiction to the fact that $\mathcal{Q}$ is a set of node-disjoint paths.

Similarly, it is easy to see that for all $1 \leq i < r$, $d(Q_{2i-1}, Q_{2i+1}) \geq 2$. This is since the removal of the path Q_{2i} partitions the snake $\mathcal{Y}$ into two disjoint subgraphs, with path Q_{2i-1} contained in one and path Q_{2i+1} contained in the other.

Our final set of path is $\hat{\mathcal{Q}} = \{Q_{2i-1} : 1 \leq i \leq r\}$. From the above discussion, it is a spaced-out set of paths contained in $\mathcal{Y}$, and for each $1 \leq i \leq r$, path $Q_{2i-1} \in \hat{\mathcal{Q}}$ connects b_i to b'_i. $\square$

In order to complete the proof, we need the following easy observation.

Observation C.4. There is an efficient algorithm that constructs, for each $1 \leq j \leq N_1$, a snake $\mathcal{Y}_j$ of width at least $2M$ in G^t, such that all resulting snakes are mutually disjoint, and for each $1 \leq j \leq N_1$:

- the bottom boundary of the first corridor of $\mathcal{Y}_j$ contains I''_{j-1};
- the top boundary of the last corridor of $\mathcal{Y}_j$ contains I'_j; and
- all snakes are disjoint from R, except for $\mathcal{Y}_1$, that contains $I''_0 \subseteq R$ as part of is boundary, and does not contain any other vertices of R.

The construction of the snakes is immediate and exploits the ample space between the boxes $\hat{K}_j$; (see Figure 4 for an illustration). From Corollary C.3, for each $1 \leq j \leq N_1$, we obtain a set $\mathcal{P}_j$ of spaced-out paths contained in $\mathcal{Y}_j$, such that for each $1 \leq i \leq M'$, there is a path $P^i_j \in \mathcal{P}_j$ connecting vertex $x''(j-1,i)$ to vertex $x'(j,i)$. For each $1 \leq i \leq M'$, let P_i be the path obtained by concatenating the paths $\left\{P^i_1, \mathcal{W}^i_1, P^i_2, \ldots, P^i_{N_1}, \mathcal{W}^i_{N_1}\right\}$. The final set of paths is $\mathcal{P}^1 = \{P_1, \ldots, P_{M'}\}$.

D Acknowledgment

We thank the anonymous reviewers for carefully reading this manuscript and for their many helpful suggestions on simplifying and improving the presentation, especially for the suggestion on how to derandomize the original randomized reduction.

References

[1] NOGA ALON, SANJEEV ARORA, RAJSEKAR MANOKARAN, DANA MOSHKOVITZ, AND OMRI WEINSTEIN: Inapproximabilty of densest k-subgraph from average case hardness, 2011. Manuscript. 6

[2] NOGA ALON, PAUL SEYMOUR, AND ROBIN THOMAS: Planar separators. *SIAM J. Discr. Math.*, 7(2):184–193, 1994. [doi:10.1137/S0895480191198768] 44

[3] MATTHEW ANDREWS: Approximation algorithms for the edge-disjoint paths problem via Raecke decompositions. In *Proc. 51st FOCS*, pp. 277–286. IEEE Comp. Soc., 2010. [doi:10.1109/FOCS.2010.33] 5

[4] MATTHEW ANDREWS, JULIA CHUZHOY, VENKATESAN GURUSWAMI, SANJEEV KHANNA, KUNAL TALWAR, AND LISA ZHANG: Inapproximability of edge-disjoint paths and low congestion routing on undirected graphs. *Combinatorica*, 30(5):485–520, 2010. [doi:10.1007/s00493-010-2455-9] 3, 5

[5] MATTHEW ANDREWS AND LISA ZHANG: Logarithmic hardness of the undirected edge-disjoint paths problem. *J. ACM*, 53(5):745–761, 2006. [doi:10.1145/1183907.1183910] 3

[6] YONATAN AUMANN AND YUVAL RABANI: Improved bounds for all optical routing. In *Proc. 6th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA'95)*, pp. 567–576. SIAM, 1995. ACM DL. 4, 5

[7] BARUCH AWERBUCH, RAINER GAWLICK, TOM LEIGHTON, AND YUVAL RABANI: On-line admission control and circuit routing for high performance computing and communication. In *Proc. 35th FOCS*, pp. 412–423. IEEE Comp. Soc., 1994. [doi:10.1109/SFCS.1994.365675] 5

[8] ADITYA BHASKARA, MOSES CHARIKAR, EDEN CHLAMTAC, URIEL FEIGE, AND ARAVINDAN VIJAYARAGHAVAN: Detecting high log-densities: an $O(n^{1/4})$ approximation for densest k-subgraph. In *Proc. 42nd STOC*, pp. 201–210. ACM Press, 2010. [doi:10.1145/1806689.1806719] 6

[9] ANDREI Z. BRODER, ALAN M. FRIEZE, STEPHEN SUEN, AND ELI UPFAL: Optimal construction of edge-disjoint paths in random graphs. *SIAM J. Comput.*, 28(2):541–573, 1998. [doi:10.1137/S0097539795290805] 5

[10] ANDREI Z. BRODER, ALAN M. FRIEZE, AND ELI UPFAL: Existence and construction of edge disjoint paths on expander graphs. *SIAM J. Comput.*, 23(5):976–989, 1994. Preliminary version in STOC'92. [doi:10.1137/S0097539792232021] 5

[11] CHANDRA CHEKURI AND ALINA ENE: Poly-logarithmic approximation for maximum node disjoint paths with constant congestion. In *Proc. 24th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA'13)*, pp. 326–341. SIAM, 2013. [doi:10.1137/1.9781611973105.24] 5

[12] CHANDRA CHEKURI, SANJEEV KHANNA, AND F. BRUCE SHEPHERD: Multicommodity flow, well-linked terminals, and routing problems. In *Proc. 37th STOC*, pp. 183–192. ACM Press, 2005. [doi:10.1145/1060590.1060618] 5

[13] CHANDRA CHEKURI, SANJEEV KHANNA, AND F. BRUCE SHEPHERD: An $O(\sqrt{n})$ approximation and integrality gap for disjoint paths and unsplittable flow. *Theory of Computing*, 2(7):137–146, 2006. [doi:10.4086/toc.2006.v002a007] 4

[14] CHANDRA CHEKURI, SANJEEV KHANNA, AND F. BRUCE SHEPHERD: Edge-disjoint paths in planar graphs with constant congestion. *SIAM J. Comput.*, 39(1):281–301, 2009. Preliminary version in STOC'06. [doi:10.1137/060674442] 5

[15] CHANDRA CHEKURI, MARCELO MYDLARZ, AND F. BRUCE SHEPHERD: Multicommodity demand flow in a tree and packing integer programs. *ACM Trans. Algorithms*, 3(3):27:1–27:23, 2007. [doi:10.1145/1273340.1273343] 5

[16] JULIA CHUZHOY: Routing in undirected graphs with constant congestion. *SIAM J. Comput.*, 45(4):1490–1532, 2016. [doi:10.1137/130910464] 5

[17] JULIA CHUZHOY AND DAVID HONG KYUN KIM: On approximating node-disjoint paths in grids. In *Proc. 18th Internat. Workshop on Approximation Algorithms for Combinat. Opt. Probl. (APPROX'15)*, pp. 187–211. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015. [doi:10.4230/LIPIcs.APPROX-RANDOM.2015.187] 3, 4

[18] JULIA CHUZHOY, DAVID HONG KYUN KIM, AND SHI LI: Improved approximation for node-disjoint paths in planar graphs. In *Proc. 48th STOC*, pp. 556–569. ACM Press, 2016. [doi:10.1145/2897518.2897538] 3

[19] JULIA CHUZHOY, DAVID HONG KYUN KIM, AND RACHIT NIMAVAT: New hardness results for routing on disjoint paths. In *Proc. 49th STOC*, pp. 86–99. ACM Press, 2017. [doi:10.1145/3055399.3055411] 3, 4, 7, 50

[20] JULIA CHUZHOY, DAVID HONG KYUN KIM, AND RACHIT NIMAVAT: Almost polynomial hardness of node-disjoint paths in grids. In *Proc. 50th STOC*, pp. 1220–1233. ACM Press, 2018. [doi:10.1145/3188745.3188772] 1

[21] JULIA CHUZHOY, DAVID HONG KYUN KIM, AND RACHIT NIMAVAT: Improved approximation for node-disjoint paths in grids with sources on the boundary. In *Proc. 45th Internat. Colloq. on Automata, Languages, and Programming (ICALP'18)*, pp. 38:1–38:14. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018. [doi:10.4230/LIPIcs.ICALP.2018.38] 4

[22] JULIA CHUZHOY AND SHI LI: A polylogarithmic approximation algorithm for edge-disjoint paths with congestion 2. *J. ACM*, 63(5):45:1–45:51, 2016. Preliminary version in FOCS'12. [doi:10.1145/2893472] 5

[23] SHIMON EVEN, ALON ITAI, AND ADI SHAMIR: On the complexity of timetable and multicommodity flow problems. *SIAM J. Comput.*, 5(4):691–703, 1976. [doi:10.1137/0205048] 3

[24] URIEL FEIGE: Relations between average case complexity and approximation complexity. In *Proc. 34th STOC*, pp. 534–543. ACM Press, 2002. [doi:10.1145/509907.509985] 6

[25] URIEL FEIGE, MAGNÚS M. HALLDÓRSSON, GUY KORTSARZ, AND ARAVIND SRINIVASAN: Approximating the domatic number. *SIAM J. Comput.*, 32(1):172–195, 2002. [doi:10.1137/S0097539700380754] 9

[26] KRZYSZTOF FLESZAR, MATTHIAS MNICH, AND JOACHIM SPOERHASE: New algorithms for maximum disjoint paths based on tree-likeness. *Math. Programming*, 171:433–461, 2018. Preliminary version in ESA'16. [doi:10.1007/s10107-017-1199-3] 5

[27] ALAN M. FRIEZE: Edge-disjoint paths in expander graphs. *SIAM J. Comput.*, 30(6):1790–1801, 2001. Preliminary version in SODA'00. [doi:10.1137/S0097539700366103] 5

[28] NAVEEN GARG, VIJAY V. VAZIRANI, AND MIHALIS YANNAKAKIS: Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18(1):3–20, 1997. [doi:10.1007/BF02523685] 5

[29] THOMAS HOLENSTEIN: Parallel repetition: Simplification and the no-signaling case. *Theory of Computing*, 5(8):141–172, 2009. Preliminary version in STOC'07. [doi:10.4086/toc.2009.v005a008] 11

[30] RICHARD KARP: On the complexity of combinatorial problems. *Networks*, 5(1):45–68, 1975. [doi:10.1002/net.1975.5.1.45] 3

[31] KEN-ICHI KAWARABAYASHI AND YUSUKE KOBAYASHI: An $O(\log n)$-approximation algorithm for the edge-disjoint paths problem in Eulerian planar graphs. *ACM Trans. Algorithms*, 9(2):16:1–16:13, 2013. [doi:10.1145/2438645.2438648] 5

[32] SUBHASH KHOT: Ruling out PTAS for graph min-bisection, dense k-subgraph, and bipartite clique. *SIAM J. Comput.*, 36(4):1025–1071, 2006. [doi:10.1137/S0097539705447037] 6

[33] JON KLEINBERG: An approximation algorithm for the disjoint paths problem in even-degree planar graphs. In *Proc. 46th FOCS*, pp. 627–636. IEEE Comp. Soc., 2005. [doi:10.1109/SFCS.2005.18] 5

[34] JON KLEINBERG AND RONITT RUBINFELD: Short paths in expander graphs. In *Proc. 37th FOCS*, pp. 86–95. IEEE Comp. Soc., 1996. [doi:10.1109/SFCS.1996.548467] 5

[35] JON M. KLEINBERG AND ÉVA TARDOS: Disjoint paths in densely embedded graphs. In *Proc. 36th FOCS*, pp. 52–61. IEEE Comp. Soc., 1995. [doi:10.1109/SFCS.1995.492462] 4, 5

[36] JON M. KLEINBERG AND ÉVA TARDOS: Approximations for the disjoint paths problem in high-diameter planar networks. *J. Comput. System Sci.*, 57(1):61–73, 1998. [doi:10.1006/jcss.1998.1579] 4, 5

[37] STAVROS G. KOLLIOPOULOS AND CLIFFORD STEIN: Approximating disjoint-path problems using packing integer programs. *Math. Programming*, 99(1):63–87, 2004. [doi:10.1007/s10107-002-0370-6] 3, 4

[38] MARK R. KRAMER AND JAN VAN LEEUWEN: The complexity of wire-routing and finding minimum area layouts for arbitrary VLSI circuits. *Adv. Comput. Res.*, 2:129–146, 1984. 3

[39] TOM LEIGHTON AND SATISH RAO: Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM*, 46(6):787–832, 1999. [doi:10.1145/331524.331526] 5

[40] RICHARD J. LIPTON AND ROBERT ENDRE TARJAN: A separator theorem for planar graphs. *SIAM J. Appl. Math.*, 36(2):177–189, 1979. [doi:10.1137/0136016] 44, 45

[41] JAMES F. LYNCH: The equivalence of theorem proving and the interconnection problem. *SIGDA Newsl.*, 5(3):31–36, 1975. [doi:10.1145/1061425.1061430] 3

[42] PASIN MANURANGSI: Almost-polynomial ratio ETH-hardness of approximating densest k-subgraph. In *Proc. 49th STOC*, pp. 954–961. ACM Press, 2017. [doi:10.1145/3055399.3055412] 6

[43] HARALD RÄCKE: Minimizing congestion in general networks. In *Proc. 43rd FOCS*, pp. 43–52. IEEE Comp. Soc., 2002. [doi:10.1109/SFCS.2002.1181881] 5

[44] PRABHAKAR RAGHAVAN AND CLARK D. THOMPSON: Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374, 1987. [doi:10.1007/BF02579324] 5

[45] PRASAD RAGHAVENDRA AND DAVID STEURER: Graph expansion and the unique games conjecture. In *Proc. 42nd STOC*, pp. 755–764. ACM Press, 2010. [doi:10.1145/1806689.1806792] 6

[46] ANUP RAO: Parallel repetition in projection games and a concentration bound. *SIAM J. Comput.*, 40(6):1871–1891, 2011. Preliminary version in STOC'08. [doi:10.1137/080734042] 11

[47] SATISH RAO AND SHUHENG ZHOU: Edge disjoint paths in moderately connected graphs. *SIAM J. Comput.*, 39(5):1856–1887, 2010. [doi:10.1137/080715093] 5

[48] RAN RAZ: A parallel repetition theorem. *SIAM J. Comput.*, 27(3):763–803, 1998. [doi:10.1137/S0097539795280895] 11

[49] NEIL ROBERTSON AND PAUL D. SEYMOUR: Outline of a disjoint paths algorithm. In *Paths, Flows and VLSI-Layout*. Springer, 1990. WorldCat.org. 3

[50] NEIL ROBERTSON AND PAUL D. SEYMOUR: Graph minors. XIII. The disjoint paths problem. *J. Combin. Theory–B*, 63(1):65–110, 1995. [doi:10.1006/jctb.1995.1006] 3

[51] LOÏC SÉGUIN-CHARBONNEAU AND F. BRUCE SHEPHERD: Maximum edge-disjoint paths in planar graphs with congestion 2. *Math. Programming*, 188:295–317, 2021. Preliminary version in FOCS'11. [doi:10.1007/s10107-020-01513-1] 5

[52] PETER UNGAR: A theorem on planar graphs. *J. London Math. Soc.*, 26(4):256–262, 1951. [doi:10.1112/jlms/s1-26.4.256] 44

AUTHORS

Julia Chuzhoy
Professor
Toyota Technological Institute at Chicago
Chicago, IL, USA
cjulia@ttic.edu
https://ttic.uchicago.edu/~cjulia

David Hong Kyun Kim
Software Engineer
Google
Sunnyvale, CA, USA
dave.hongk@gmail.com
http://people.cs.uchicago.edu/~hongk

Rachit Nimavat
Ph. D. Student
Toyota Technological Institute at Chicago
Chicago, IL, USA
nimavat@ttic.edu
https://home.ttic.edu/~nimavat

ABOUT THE AUTHORS

JULIA CHUZHOY is a Professor at the Toyota Technological Institute at Chicago. She completed her Ph. D. at the Technion, Israel, and spent three years as a postdoctoral scholar at MIT, the University of Pennsylvania and the Institute for Advanced Study in Princeton. She mainly works on algorithms for graph problems. In her spare time she likes to read books, learns to play piano and studies French.

DAVID HONG KYUN KIM received his Ph. D. in Computer Science in May 2018 from the University of Chicago, where he studied approximation algorithms for network routing problems under the supervision of Julia Chuzhoy. He also worked on scheduling problems and their applications to real systems with Henry Hoffmann. He now works as a software engineer at Google and enjoys spending his free time hiking and listening to music.

RACHIT NIMAVAT is a Ph. D. student at the Toyota Technological Institute, Chicago (TTIC). His advisor is Julia Chuzhoy. In 2015, Rachit received his BTech in Computer Science and Engineering from the Indian Institute of Technology, Kanpur, under the supervision of Prof. Surender Baswana. Rachit's work is mainly in approximation algorithms and hardness of approximation. In his free time he likes to cook, while trying to optimize the ratio of taste over effort. His favorite cuisine is of his home state of Gujarat, India.