Almost Polynomial Hardness of Node-Disjoint Paths in Grids

Julia Chuzhoy[*]   David H. K. Kim[†]   Rachit Nimavat[‡]

November 6, 2017

Abstract

In the classical Node-Disjoint Paths (NDP) problem, we are given an n-vertex graph $G = (V, E)$, and a collection $\mathcal{M} = \{(s_1, t_1), \ldots, (s_k, t_k)\}$ 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, while, until recently, the best negative result was a factor $\Omega(\log^{1/2-\epsilon} n)$-hardness of approximation, for any constant ϵ, unless $\mathsf{NP} \subseteq \mathsf{ZPTIME}(n^{\mathrm{poly} \log n})$. In a recent work, the authors have shown an improved $2^{\Omega(\sqrt{\log n})}$-hardness of approximation for NDP, unless $\mathsf{NP} \subseteq \mathsf{DTIME}(n^{O(\log n)})$, even if the underlying graph is a **subgraph** of a grid graph, and all source vertices lie on the boundary of the grid. 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. In a recent work, the authors showed a $2^{\tilde{O}(\sqrt{\log n})}$-approximation algorithm for NDP in grid graphs, if all source vertices lie on the boundary of the grid – a result that can be seen as suggesting that a sub-polynomial approximation may be achievable for NDP in grids. In this paper we show that this is unlikely to be the case, and come close to resolving the approximability of NDP in general, and NDP in grids in particular. Our main result is that NDP is $2^{\Omega(\log^{1-\epsilon} n)}$-hard to approximate for any constant ϵ, assuming that $\mathsf{NP} \not\subseteq \mathsf{RTIME}(n^{\mathrm{poly} \log n})$, and that it is $n^{\Omega(1/(\log \log n)^2)}$-hard to approximate, assuming that for some constant $\delta > 0$, $\mathsf{NP} \not\subseteq \mathsf{RTIME}(2^{n^\delta})$. 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. The construction of each new instance of NDP crucially depends on the solutions to the previous instances that were found by the approximation algorithm.

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, where 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 [RS90, RS95] 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 the input, the problem becomes NP-hard [Kar75, EIS76], and it remains NP-hard even for planar graphs [Lyn75], and for grid graphs [KvL84]. The best current upper bound on the approximability of NDP is $O(\sqrt{n})$, obtained by a simple greedy algorithm [KS04]. Until recently, the best known lower bound was an $\Omega(\log^{1/2-\epsilon} n)$-hardness of approximation for any constant ϵ, unless NP $\subseteq$ ZPTIME$(n^{\text{poly} \log n})$ [AZ05, ACG+10], and APX-hardness for the special cases of NDP-Planar and NDP-Grid [CK15]. In a recent paper [CKN17b], the authors have shown an improved $2^{\Omega(\sqrt{\log n})}$-hardness of approximation for NDP, assuming that NP $\not\subseteq$ 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 **sub-graphs** 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 [KS04] was recently slightly improved to an $\tilde{O}(n^{9/19})$-approximation [CKL16].

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 [KS04] was only recently improved to a $\tilde{O}(n^{1/4})$-approximation for NDP-Grid [CK15], while on the negative side only APX-hardness is known. In a very recent paper [CKN17a], 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 sub-graphs of grids with all sources lying on the grid boundary [CKN17b][1]. Furthermore, this result can be seen as suggesting that sub-polynomial 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 $2^{\Omega(\log^{1-\epsilon} n)}$-hard to approximate for any constant ϵ, unless NP $\subseteq$ RTIME$(n^{\text{poly} \log n})$. We further show that it is $n^{\Omega(1/(\log \log n)^2)}$-hard to approximate, assuming that for some constant $\delta > 0$, NP $\not\subseteq$ RTIME(2^{n^δ}). 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

---

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

algorithm for NDP-Grid with all sources lying on the grid boundary [CKN17a].

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 [CKS06], and the recent $2^{\Omega(\sqrt{\log n})}$-hardness of approximation of [CKN17b], under the assumption that NP $\not\subseteq$ 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 [AR95, KT98, KT95]. 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 [CK15]. The results of [CKN17b] extend to a $2^{\Omega(\sqrt{\log n})}$-hardness of approximation for EDP on sub-graphs 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.
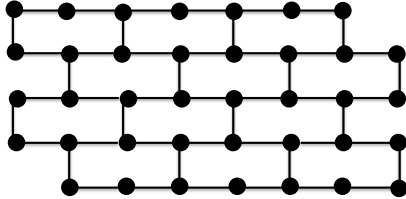

Figure 1: A wall graph.

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 [Kle05] showed an $O(\log^2 n)$-approximation algorithm, while Kawarabayashi and Kobayashi [KK13] 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 [LR99, BFU92, BFSU98, KR96, Fri00], and constant-factor approximation algorithms are known for trees [GVY97, CMS07], and grids and grid-like graphs [AR95, AGLR94, KT98, KT95]. Rao and Zhou [RZ10] showed an efficient randomized $O(\text{poly} \log n)$-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. [FMS16] designed an $O(\sqrt{r} \cdot \log^{1.5}(kr))$-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 [RT87], 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 [CKS05, Räc02, And10, RZ10, Chu16, CL16, CE13, CC] has led to an $O(\operatorname{poly} \log k)$-approximation for NDPwC and EDPwC with congestion bound $c = 2$. This result is 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 $\Omega((\log n)^{\frac{1-\epsilon}{c+1}})$, unless NP $\subseteq$ ZPTIME$(n^{\operatorname{poly} \log n})$ [ACG+10]. When the input graph is planar, Seguin-Charbonneau and Shepherd [SCS11], improving on the result of Chekuri, Khanna and Shepherd [CKS09], 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 $\epsilon > 0$, there is no $2^{O(\log^{1-\epsilon} n)}$-approximation algorithm for NDP, assuming that NP $\nsubseteq$ RTIME$(n^{\operatorname{poly} \log n})$. Moreover, there is no $n^{O(1/(\log \log n)^2)}$-approximation algorithm for NDP, assuming that for some constant $\delta > 0$, NP $\nsubseteq$ RTIME$(2^{n^{\delta}})$. These results hold even when the input graph is a grid graph or a wall graph.*

Theorem 1.2 *For every constant $\epsilon > 0$, there is no $2^{O(\log^{1-\epsilon} n)}$-approximation algorithm for EDP, assuming that NP $\nsubseteq$ RTIME$(n^{\operatorname{poly} \log n})$. Moreover, there is no $n^{O(1/(\log \log n)^2)}$-approximation algorithm for EDP, assuming that for some constant $\delta > 0$, NP $\nsubseteq$ RTIME$(2^{n^{\delta}})$. 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 $(W_1, W_2, \ldots, W_r)$ 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, so that $|E_i| \leq h$ 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 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 $n^{1/4+\epsilon}$-approximation for any constant ϵ [BCC+10]. 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 [Fei02, AAM+11, Kho04, RS10]. In a recent breakthrough, Manurangsi [Man17] 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 [Man17] 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 the 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 [CKN17b]. The proof in [CKN17b] proceeded by performing a reduction from 3SAT(5). Initially, a simple reduction from 3SAT(5) to the NDP problem on a sub-graph 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. The hardness of approximation proof for NDP-Grid appears in Section 4, with the reduction from the graph partitioning problem to NDP-Grid deferred to Section 5. Finally, we extend our hardness results to NDP-Wall and EDP-Wall in Section 6.

2   Preliminaries

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 iff every vertex in $P \cap U$ is an endpoint of P. Similarly, P is internally disjoint from a subgraph G' of G iff 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

$\mathcal{M}'$. 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 $V(G^{h,\ell}) = \{v(i,j) \mid 1 \leq i \leq h, 1 \leq j \leq \ell\}$, and the set of its edges is the union of two subsets: the set $E^H = \{(v_{i,j}, v_{i,j+1}) \mid 1 \leq i \leq h, 1 \leq j < \ell\}$ of horizontal edges and the set $E^V = \{(v_{i,j}, v_{i+1,j}) \mid 1 \leq i < h, 1 \leq j \leq \ell\}$ 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 $1 \leq i \leq h$, 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 $1 \leq j \leq \ell$, 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* iff $h = \ell$. 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 *sub-grid of G spanned by the rows in $\mathcal{R}'$ and the columns in $\mathcal{W}'$* is the sub-graph 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 $e_1^j, \ldots, e_{h-1}^j$ be the edges of W_j indexed in their top-to-bottom order. Let $E^*(G) \subseteq E(G)$ contain all edges e_z^j, where $z \neq j \mod 2$, 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 $R_1, \ldots, R_h$ 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 $W_1, \ldots, W_{\ell/2}$ 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 $\hat{G}$.

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 χ iff $\chi(u) \neq \chi(v)$. The coloring χ is *valid* iff it satisfies every edge. We say that G is a YES-INSTANCE iff 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, iff for every coloring $\chi : V \to \mathcal{C}$, at most a $(1 - \epsilon)$-fraction of the edges are satisfied by χ. We use the following theorem of Feige et al. [FHKS03]:

Theorem 2.1 *[Proposition 15 in [FHKS03]] 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 e to the edge-prover and 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 $(1 - \epsilon/2)$. 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 $\frac{1}{2}$, and so overall the verifier rejects with probability at least $\epsilon/2$. 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 $(1 - \epsilon/2)$.

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 $(e_1, \ldots, e_\ell)$ 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 $(e_1, \ldots, e_\ell)$ to the edge-prover, and $(v_1, \ldots, v_\ell)$ 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 $(v_1, \ldots, v_\ell)$. 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 $i \ne j$, 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 $(e_1, \ldots, e_\ell)$ and $(v_1, \ldots, v_\ell)$ is *matching*, or *consistent*, iff it causes the verifier to accept. We let $\mathcal{G}^\ell$ denote this 2-prover protocol with ℓ repetitions.

Theorem 2.2 (Parallel Repetition) *[Raz98, Hol07, Rao08] 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 $x^{\gamma'\ell}$.*

6

Corollary 2.3 *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. Then $|\mathcal{Q}^V| = n^\ell$. 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 $Q = (e_1, \ldots, e_\ell) \in \mathcal{Q}^E$ of the edge-prover participates in exactly 2^ℓ random strings (one random string for each choice of one endpoint per edge of $\{e_1, \ldots, e_\ell\}$), while each query $Q' = (v_1, \ldots, v_\ell)$ of the vertex-prover participates in exactly 5^ℓ random strings (one random string for each choice of an edge incident to each of $v_1, \ldots, v_\ell$).

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* iff 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* iff 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, there are many perfect global assignments, that neatly partition all answers to the edge-queries.

Theorem 2.4 *Assume that G is a YES-INSTANCE. Then there are 6^ℓ perfect global assignments $f_1, \ldots, f_{6^\ell}$ 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 $1 \leq i \leq 6^\ell$ 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 $1 \leq i \leq 6^\ell$, for which $f_i(Q') = A'$.*

Two Graphs. 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 $|E(H)| = |\mathcal{R}| = (5n)^\ell$.

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

7

convenient for now to think that $H' = H$, but later we will use smaller sub-graphs 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 $v_1(Q', A'), \ldots, v_{2^\ell}(Q', A')$. 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':

$$S(Q') = \left\{ v_i(Q', A') \mid A' \in \mathcal{A}^V, 1 \leq i \leq 2^\ell \right\},$$

so $|S(Q')| = 6^\ell$. We call $S(Q')$ *the group representing* Q'. We denote by $\hat{U}^V$ the resulting set of vertices:

$$\hat{U}^V = \left\{ v_i(Q', A') \mid (Q' \in \mathcal{Q}^V \text{ and } v(Q') \in H'), A' \in \mathcal{A}^V, 1 \leq i \leq 2^\ell \right\}.$$

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

$$E(R) = \left\{ (v(Q^E, A), v_i(Q^V, A')) \mid A \in \mathcal{A}^E, A' \in \mathcal{A}^V, \ A \text{ and } A' \text{ are consistent answers to } R, 1 \leq i \leq 2^\ell \right\}$$

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 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.

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 $1 \le i \le r$, a subset $E_i \subseteq E(W_i)$ of edges, such that $|E_i| \le h$. 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 graph H, and the graph $L(H)$, as described above. 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 2.4, in order to partition the vertices of $L(H)$ into $r = 6^\ell$ clusters $W_1, \ldots, W_r$, 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 graph $L(H)$ and the properties of the assignments f_i guaranteed by Theorem 2.4, we indeed obtain a partition $W_1, \ldots, W_r$ 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[2]. 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

---

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

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 $U \in \mathcal{U}_1 \cup \mathcal{U}_2$, $|U| = r$. 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 $U \in \mathcal{U}_1$, 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 $\sum_i |E_i|$.

Valid 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 1 *We say that instance $\mathcal{I}$ of* (r,h)-GPwB *is* valid *iff* $h = \beta^*(\mathcal{I})/r$ *and* $h \ge \max_{v \in V_1 \cup V_2} \{\beta(v)\}$.

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 2 *We say that a solution* $((W_1, \dots, W_r), (E_1, \dots, E_r))$ *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 $1 \le i \le r$, $|E_i| = h$.*

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.

3.1   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 subgraph $H' \subseteq H$. We define an instance $\mathcal{I}(H')$ of (r,h)-GPwB, as follows.

10

- 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 3.1 *Let G be an instance of the 3COL(5) problem, $\ell > 0$ an integral parameter, and $H' \subseteq H$ a 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, 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 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.

Assume now that G is a YES-INSTANCE. We define a perfect solution $((W_1, \ldots, W_r), (E_1, \ldots, E_r))$ to this instance. Let $\{f_1, f_2, \ldots, f_{6^\ell}\}$ be the collection of perfect global assignments of answers to the queries, given by Theorem 2.4. Recall that $\mathcal{U}_1 = \{S(Q) \mid Q \in \mathcal{Q}^E, v(Q) \in H'\}$ and $\mathcal{U}_2 = \{S(Q') \mid Q' \in \mathcal{Q}^V, v(Q') \in H'\}$, where each group in $\mathcal{U}_1 \cup \mathcal{U}_2$ has cardinality $r = 6^\ell$. We now fix some $1 \leq i \leq r$, and define the set W_i of vertices. For each query $Q \in \mathcal{Q}^E$ to the edge-prover, 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, if $A' = f_i(Q')$, then we select some index $1 \leq j \leq 2^\ell$, 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 2.4, it is easy to verify that $W_1, \ldots, W_r$ 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 $1 \leq i \leq r$, 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 $1 \leq i \leq r$. 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, $|E_i| = |E(H')| = h$. □

3.2 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 3 *Let $\mathcal{P}$ be a set of paths in a grid $\hat{G}$. We say that $\mathcal{P}$ is a* spaced-out *set iff for each pair $P, P' \in \mathcal{P}$ of paths, $d(V(P), V(P')) \geq 2$, and all paths in $\mathcal{P}$ are internally disjoint from the boundaries of the grid $\hat{G}$.*

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 3.2 *There is a constant $c^* > 0$, and there is an efficient randomized algorithm, that, given a valid instance $\mathcal{I} = (\tilde{G}, \mathcal{U}_1, \mathcal{U}_2, h, r)$ of (r,h)-GPwB with $|E(\tilde{G})| = M$, 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 with probability at least $\frac{1}{2}$ over the construction of $\hat{\mathcal{I}}$, instance $\hat{\mathcal{I}}$ has a solution $\mathcal{P}$ that routes at least $\frac{\beta^*}{c^* \log^3 M}$ demand pairs, such that the paths in $\mathcal{P}$ are spaced-out; and*

- *There is a deterministic efficient algorithm, 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 5.

Assume now that we are given an instance G of 3COL(5), an integral parameter $\ell > 0$, and a subgraph $H' \subseteq H$ of the corresponding constraint graph. Recall that we have constructed a corresponding instance $\mathcal{I}(H')$ of (r,h)-GPwB. We can then use Theorem 3.2 to construct a (random) 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 all $H' \subseteq H$; 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 3.2, and we denote $c_{\text{YI}} = (\hat{c} \cdot c^*)^3$. We obtain the following immediate corollary of Theorem 3.2:

Corollary 3.3 *Suppose we are given 3COL(5) instance G that is a YES-INSTANCE, an integer $\ell > 0$, and a subgraph $H' \subseteq H$ of the corresponding constraint graph. Then with probability at least $\frac{1}{2}$, instance $\hat{\mathcal{I}}(H')$ of NDP-Grid has a solution of value at least $\frac{|E(H')| \cdot 6^\ell}{c_{\text{YI}} \ell^3 \log^3 n}$, where $n = |V(G)|$. (The probability is over the random construction of $\hat{\mathcal{I}}(H')$).*

Proof: From Claim 3.1, instance $\mathcal{I}(H') = (L(H'), \mathcal{U}_1, \mathcal{U}_2, r, h)$ of (r,h)-GPwB is a valid instance, and it has a perfect solution, whose value must be $\beta^* = \beta^*(\mathcal{I}(H')) = h \cdot r = |E(H')| \cdot 6^\ell$. From Theorem 3.2, with probability at least $1/2$, instance $\hat{\mathcal{I}}(H')$ of NDP-Grid has a solution of value at least $\frac{|E(H')| \cdot 6^\ell}{c^* \log^3 M}$, where $M = |E(L(H'))|$. Since $\log M \leq \hat{c}\ell \log n$, and the corollary follows. $\square$

4 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 2.3). We will set the value of the 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 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.

We note that, if G is a YES-INSTANCE, then from Corollary 3.3, with constant probability there is a solution to $\hat{\mathcal{I}}(H)$ of value $\frac{|\mathcal{R}| \cdot 6^\ell}{c_{\mathrm{YI}} \ell^3 \log^3 n}$. 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 $\frac{|\mathcal{R}| \cdot 6^\ell}{c_{\mathrm{YI}}^2 \cdot \alpha^* \ell^6 \log^6 n}$, we would be done. Indeed, in such a case, from Theorem 3.2, every solution to the NDP-Grid instance $\hat{\mathcal{I}}(H)$ routes fewer than $\frac{|\mathcal{R}| \cdot 6^\ell}{c_{\mathrm{YI}} \alpha^* \ell^3 \log^3 n}$ 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 $\frac{|\mathcal{R}| \cdot 6^\ell}{c_{\mathrm{YI}} \alpha^* \ell^3 \log^3 n}$ 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. 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_i. 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 sub-graphs of H decrease very fast. The main tool that allows us to execute this plan is the following theorem.

Theorem 4.1 *Suppose we are given an instance G of the 3COL(5) problem with $|V(G)| = n$, and an integral parameter $\ell > \log^2 n$, together with some subgraph $H' \subseteq H$ of the corresponding constraint graph H, and a parameter $P > 1$. 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 $\alpha = c_{\mathrm{YI}}^2 \cdot \alpha^* \cdot \ell^6 \log^6 n$. Then there is a randomized algorithm whose running time is $O\left(n^{O(\ell)} \cdot \log P\right)$, that returns one of the following:*

- *Either a randomized strategy for the two provers that satisfies, in expectation, 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 sub-graphs of H', such that for each $H'' \in \mathcal{H}$, $|E(H'')| \leq |E(H')|/2^{\gamma\ell/16}$, and with probability at least $(1 - 1/P)$, $\sum_{H'' \in \mathcal{H}} |E(H'')| \geq \frac{c'|E(H')|}{\ell^2\alpha^2}$, 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). Suppose we are given an instance G of 3COL(5).

For an integral parameter $\ell > \log^2 n$, let H be the constraint graph corresponding to G and ℓ. We next show a randomized 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 sub-graphs 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 $\{Q^E \in \mathcal{Q}^E \mid v(Q^E) \in V(H')\}$, and $\{Q^V \in \mathcal{Q}^V \mid v(Q^V) \in V(H')\}$ respectively, and the constraints of the verifier are $\mathcal{R}(H') = \{R \in \mathcal{R} \mid e(R) \in E(H')\}$. For each inactive cluster $H' \in \mathcal{H}_2$, we will store a (possibly randomized) 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, $\mathcal{H}$ contains a single cluster – the graph H, which 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).

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 4.2 *If G is a* YES-INSTANCE*, then with high probability, the algorithm 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) or in Step (5b). We now analyze these two steps.

Consider first Step (3). From Corollary 3.3, with probability at least $1/2$, a random graph $\hat{\mathcal{I}}(H')$ has a solution of value at least $\frac{|E(H')| \cdot 6^\ell}{c_{\mathrm{YI}}\ell^3 \log^3 n}$, and our α^*-approximation algorithm to NDP-Grid must then return a solution of value at least $\frac{|E(H')| \cdot 6^\ell}{c_{\mathrm{YI}}\alpha^*\ell^3 \log^3 n}$. Since we use $n^{4\ell}$ independent random constructions of $\hat{\mathcal{I}}(H')$, with high probability, for at least one of them, we will obtain a solution of value at least $\frac{|E(H')|6^\ell}{c_{\mathrm{YI}}\alpha^*\ell^3 \log^3 n}$. Therefore, with high probability our algorithm will not return "G is a NO-INSTANCE" due to Step (3) in this iteration.

Consider now Step (5b). The algorithm can classify G as a NO-INSTANCE in this step only if $\sum_{H'' \in \mathcal{H}} |E(H'')| < \frac{c'|E(H')|}{\ell^2\alpha^2}$. From Theorem 4.1, this happens with probability at most $1/P$, and from our setting of the parameter P to be $n^{c\ell}$ for a large enough constant c, with high probability our algorithm will not return "G is a NO-INSTANCE" due to Step (5b) in this iteration.

It is not hard to see that our algorithm performs $n^{O(\ell)}$ iterations, and so, using the union bound, with high probability, it will classify G as a YES-INSTANCE. $\qquad\square$

Lemma 4.3 *If G is a* NO-INSTANCE*, then the algorithm always returns "G is a* NO-INSTANCE*".*

Proof: From Corollary 2.3, 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 fraction-$2^{-\gamma\ell}$ of the constraints in $\mathcal{R}$.

14

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

1. Construct an instance $\mathcal{I}(H')$ of (r,h)-GPwB.

2. Use Theorem 3.2 to independently construct $n^{4\ell}$ instances $\hat{\mathcal{I}}(H')$ of the NDP-Grid problem.

3. Run the α^*-approximation algorithm $\mathcal{A}$ on each such instance $\hat{\mathcal{I}}(H')$. If the resulting solution, for each of these instances, routes fewer than $\frac{|E(H')|6^\ell}{c_{\mathrm{YI}}\alpha^*\ell^3 \log^3 n}$ demand pairs, halt and return "G is a NO-INSTANCE".

4. Otherwise, fix any instance $\hat{\mathcal{I}}(H')$ for which the algorithm returned a solution routing at least $\frac{|E(H')|6^\ell}{c_{\mathrm{YI}}\alpha^*\ell^3 \log^3 n}$ demand pairs. Denote $|E(L(H'))| = M$, and recall that $M \le n^{\hat{c}\ell}$. Use Theorem 3.2 to compute a solution $((W_1,\ldots,W_r),(E_1,\ldots,E_r))$ to the instance $\mathcal{I}(H')$ of the (r,h)-GPwB problem, of value at least:

$$\frac{|E(H')| \cdot 6^\ell}{(c_{\mathrm{YI}}\alpha^*\ell^3 \log^3 n)(c^* \log^3 M)} \ge \frac{|E(H')| \cdot 6^\ell}{c_{\mathrm{YI}}c^*\hat{c}^3\alpha^*\ell^6 \log^6 n} \ge \frac{|E(H')| \cdot 6^\ell}{c_{\mathrm{YI}}^2\alpha^*\ell^6 \log^6 n} = \frac{|E(H')| \cdot 6^\ell}{\alpha}.$$

5. Apply the algorithm from Theorem 4.1 to this solution, with the parameter $P = n^{c\ell}$, for a sufficiently large constant c.

   (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 sub-graphs of H' returned by the algorithm. If $\sum_{H'' \in \mathcal{H}} |E(H'')| < \frac{c'|E(H')|}{\ell^2\alpha^2}$, then return "$G$ is a NO-INSTANCE". Otherwise, remove H' from $\mathcal{H}_1$ and add all graphs of $\tilde{\mathcal{H}}$ to $\mathcal{H}_1$.

Figure 2: An iteration description

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:

$$\sum_{H' \in \mathcal{H}_2} |E(H')| \ge \frac{|\mathcal{R}|}{(\ell^2\alpha^2/c')^{O(\log n)}} = \frac{|\mathcal{R}|}{(\ell^{14} \cdot (\alpha^*)^2 \cdot \log^{12} n)^{O(\log n)}} = \frac{|\mathcal{R}|}{(\alpha^*)^{O(\log n)}}.$$

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
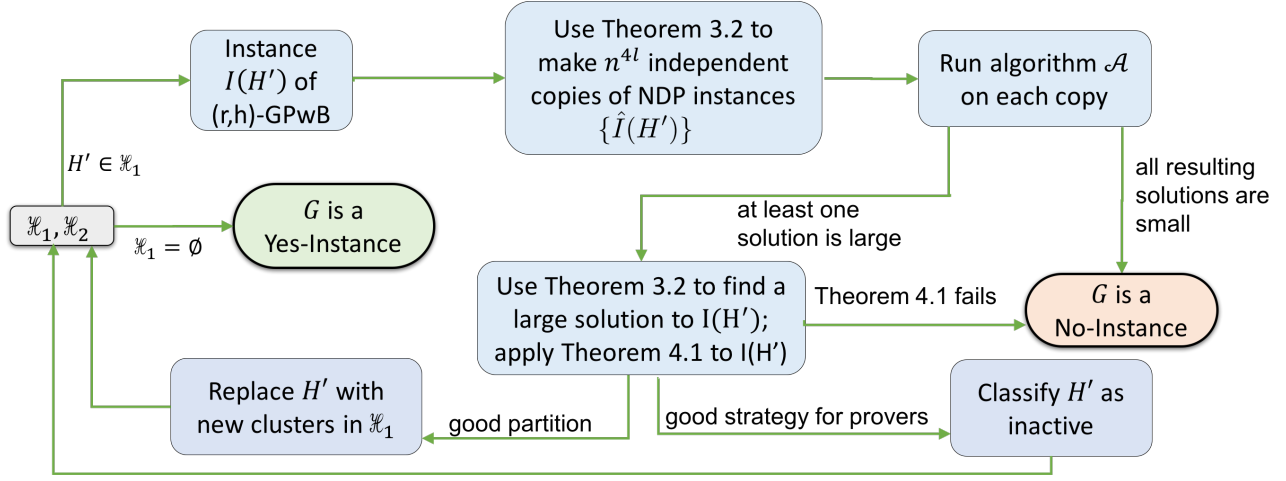
15

Figure 3: Flowchart for an iteration execution

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. As observed above, our 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 each of the $n^{O(\ell)}$ resulting instances 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, if G is a No-Instance, then the algorithm always classifies it as such, and if G is a Yes-Instance, then the algorithm classifies it as a Yes-Instance with high probability. 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. The running time of the algorithm is $n^{O(\ell)}$, and it is a randomized algorithm with a one-sided error.

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 $2^{(\log n)^{(1-\epsilon)}}$-hardness of approximation for NDP-Grid for any constant ϵ, assuming NP $\not\subseteq$ RTIME$(n^{\text{poly} \log n})$.

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 NP $\not\subseteq$ RTIME(2^{n^δ}) for some constant $\delta > 0$.

4.1 Proof of Theorem 4.1

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 $e(R) \in E(H')$. 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* iff E' contains at least $6^\ell/(2\alpha)$ edges of $E(R)$, and we say that it is bad otherwise.

Observation 4.4 *At least $|\mathcal{R}'|/(2\alpha)$ random strings of $\mathcal{R}'$ are good.*

Proof: Let x denote the fraction of good random strings in $\mathcal{R}'$. A good random string contributes at most 6^ℓ edges to E', while a bad random string contributes at most $6^\ell/(2\alpha)$. If $x < 1/(2\alpha)$, 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 4 *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 4.5 *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 iff $|E'(R) \cap E_i| < |W_i \cap S(Q^V)|/(8\alpha)$, 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 $|E^2(R)| > 6^\ell/(16\alpha)$ is identical, and relies on the fact that $|S(Q^E)| = 6^\ell$. □

We will use the following simple observation.

Observation 4.6 *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 $R = (Q^E, Q^V) \in \mathcal{R}'$ iff $|W_i \cap S(Q^E)|, |W_i \cap S(Q^V)| > z$. We say that an edge $e \in E'(R)$ is heavy iff 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* iff 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 randomized strategy for the provers to choose assignment to the queries, so that at least a $2^{-\gamma\ell/2}$-fraction of the constraints in $\mathcal{R}'$ are satisfied in expectation. 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, 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 define 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 iff 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 $|S(Q^E)| \leq z$, or $|S(Q^V)| \leq z$. Assume without loss of generality that it is the former; the other case is symmetric. Then, since e is not terrible, from Observation 4.6, $|S(Q^V)| \leq 8\alpha z$, and so $|\mathcal{A}(Q^V)| \leq 8\alpha z$, while $|\mathcal{A}(Q^E)| \leq z$. 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 $|E^L(R)|/(8 \cdot 6^\ell z^2 \alpha) \geq 1/(64z^2\alpha^2) \geq 4\alpha/2^{\gamma\ell/2}$, since $z = 2^{\gamma\ell/8}$, and $\alpha < 2^{\gamma\ell/32}$.

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

Fix some heavy random string $R \in \mathcal{R}''$ and assume that $R = (Q^E, Q^V)$. For each $1 \leq i \leq r$, let $E_i(R) = E^H(R) \cap E_i$. Recall that, if $E_i(R) \neq \emptyset$, then $|W_i \cap S(Q^E)|, |W_i \cap S(Q^V)| \geq z$ must hold,

18

and, from the definition of terrible clusters, $|E_i(R)| \geq z/(8\alpha)$. It is also immediate that $|E_i(R)| \leq |E'(R)| \leq 6^\ell$.

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

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

Moreover,

$$|\mathcal{C}_{j_R}(R)| \geq \frac{|E^H(R)|}{\log(6^\ell) \cdot 2^j} \geq \frac{6^\ell}{8 \cdot 2^j \cdot \ell\alpha \log 6}. \tag{1}$$

Let j^* be the index that was chosen by at least $|\mathcal{R}''|/\log(6^\ell)$ random strings, and let $\mathcal{R}^* \subseteq \mathcal{R}''$ be the set of all random strings that chose j^*. We are now ready to define a collection $\mathcal{H} = \{H_1, \ldots, H_{6^\ell}\}$ of sub-graphs 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 $W_1, \ldots, W_{6^\ell}$; 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 $\mathcal{H} = \{H_1, \ldots, H_{6^\ell}\}$ of subgraphs of H'. We now show that the family $\mathcal{H}$ of graphs has the desired properties. It is immediate to verify that the graphs in $\mathcal{H}$ are disjoint.

Claim 4.7 *For each $1 \leq i \leq 6^\ell$, $|E(H_i)| \leq |E(H')|/2^{\gamma\ell/16}$.*

Proof: Fix some index $1 \leq i \leq 6^\ell$. 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 $|E_i| \cdot 8\alpha/z \leq 8\alpha h/z = 8\alpha|E(H')|/2^{\gamma\ell/8} \leq |E(H')|/2^{\gamma\ell/16}$, since $\alpha \leq 2^{\gamma\ell/32}$. $\square$

Claim 4.8 $\mathbf{E}\left[\sum_{i=1}^r |E(H_i)|\right] \geq \frac{|E(H')|}{128\ell^2\alpha^2 \log^2 6}$.

Proof: Recall that $|\mathcal{R}^*| \geq |\mathcal{R}''|/\log(6^\ell) \geq |\mathcal{R}'|/(4\alpha \log(6^\ell))$. 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 $1 \leq y \leq 6^\ell$, such that $|W_i \cap S(Q^V)| \geq 2^{j^*-1}$. Clearly, $|J| \leq 6^\ell/2^{j^*-1}$, and $v(Q^V)$ may only belong to graph H_i if $i \in J$. Similarly, let J' be the set of indices $1 \leq y \leq 6^\ell$, such that $|W_i \cap S(Q^E)| \geq 2^{j^*-1}$. As before, $|J'| \leq 6^\ell/2^{j^*-1}$, 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 (1), $|\mathcal{C}_{j^*}(R)| \geq \frac{6^\ell}{8 \cdot 2^{j^*} \cdot \ell\alpha \log 6}$.

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 at least:

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

19

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

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

$\square$

Denote the expectation of $\sum_{i=1}^{r} |E(H_i)|$ by μ, and let $c = 128\log^2 6$, so that $\mu = |E(H')|/(c\ell^2\alpha^2)$. Let $\mathcal{E}$ be the event that $\sum_{i=1}^{r} |E(H_i)| \geq |E(H')|/(2c\ell^2\alpha^2) = \mu/2$. We claim that $\mathcal{E}$ happens with probability at least $1/(2c\ell^2\alpha^2)$. Indeed, assume that it happens with probability $p < 1/(2c\ell^2\alpha^2)$. If $\mathcal{E}$ does not happen, then $\sum_{i=1}^{r} |E(H_i)| \leq \mu/2$, and if it happens, then $\sum_{i=1}^{r} |E(H_i)| \leq |E(H')|$. Overall, this gives us that $\mathbf{E}\left[\sum_{i=1}^{r} |E(H_i)|\right] \leq (1-p)\mu/2 + p|E(H')| < \mu$, a contradiction. We repeat the algorithm for constructing $\mathcal{H}$ $O(\ell^2\alpha^2 \operatorname{poly}\log n \log P)$ times. We are then guaranteed that with probability at least $(1 - 1/P)$, event $\mathcal{E}$ happens in at least one run of the algorithm. It is easy to verify that the running time of the algorithm is bounded by $O(n^{O(\ell)} \cdot \log P)$, since $|V(L(H'))| \leq n^{O(\ell)}$.

5 From (r,h)-GPwB to NDP-Grid

In this section we prove Theorem 3.2, 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 start by describing a randomized construction of the instance $\hat{\mathcal{I}} = (\hat{G}, \mathcal{M})$ of NDP-Grid.

5.1 The Construction

Fix an arbitrary ordering ρ of the groups in $\mathcal{U}_1$. Using 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 $V_1 = \{v_1, v_2, \ldots, v_{N_1}\}$, where the vertices are indexed according to their order in σ. Next, we select a **random** ordering ρ' of the groups in $\mathcal{U}_2$. We then define an ordering σ' of the vertices of V_2 exactly as before, using the ordering ρ' of $\mathcal{U}_2$. We assume that $V_2 = \{v'_1, v'_2, \ldots, v'_{N_2}\}$, where the vertices are indexed according to their ordering in σ'. We note that the choice of the ordering ρ' is the only randomized part of our construction.

Consider some vertex $v \in V_1$. Recall that $\mathcal{B}(v)$ denotes the partition of the edges incident to v into bundles, where every bundle is a non-empty subsets of edges, and that $\beta(v) = |\mathcal{B}(v)|$. Each such 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 $U(B)$. 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 $\mathcal{B}(v) = \{B_1(v), B_2(v), \ldots, B_{\beta(v)}(v)\}$, 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 $\mathcal{B}(v') = \{B_1(v'), B_2(v'), \ldots, B_{\beta(v')}(v')\}$, and we assume that the bundles are indexed according to this ordering.

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 $\ell = 2048 \cdot \lceil M^2 \cdot \log M \rceil$. The graph $\hat{G}$ is simply the $(\ell \times \ell)$-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 $B = \{e_1, \ldots, e_z\}$. 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 $s(e_1), \ldots, s(e_z)$ into a single vertex s_B. We then replace the demand pairs $(s(e_1), t(e_1)), \ldots, (s(e_z), t(e_z))$ with the demand pairs $(s_B, t(e_1)), \ldots, (s_B, t(e_z))$. 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 $\mathcal{M}(E') = \{(s_{B(e)}, t_{B'(e)}) \mid e \in E'\}$ 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 $\ell/4$ from each other and from the top and the bottom boundaries of the grid. We will map all vertices of $S(\mathcal{M})$ to R', and all vertices of $T(\mathcal{M})$ to R''.

Locations of the sources. Let $K_1, K_2, \ldots, K_{N_1}$ be a collection of N_1 disjoint sub-paths of R', where each sub-path contains $1024 \cdot \lceil h \cdot \log M \rceil$ vertices; the sub-paths 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 ℓ 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, so that 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 sub-paths of R'', each of which contains $1024 \cdot \lceil h \cdot \log M \rceil$ vertices, so that the sub-paths 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. In the following subsections we analyze its properties. The following immediate observation will be useful to us.

Observation 5.1 *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''.*

We will also use the following two auxiliary lemmas, whose proofs are straightforward and are deferred to Section B of the Appendix.

Auxiliary Lemmas Assume that we are given a set U of n items, such that P of the items are pink, and $Y = n - P$ items are yellow. Consider a random permutation π of these items.

Lemma 5.2 *For any $\log n \leq \mu \leq Y$, the probability that there is a sequence of $\lceil 4n\mu/P \rceil$ consecutive items in π that are all yellow, is at most n/e^{μ}.*

Lemma 5.3 *For any $\log n \leq \mu \leq P$, the probability that there is a set S of $\lfloor \frac{n\mu}{P} \rfloor$ consecutive items in π, such that more than 4μ of the items are pink, is at most $n/4^{\mu}$.*

5.2   From Partitioning to Routing

The goal of this subsection is to prove the following theorem.

Theorem 5.4 *Suppose we are given a valid instance $\mathcal{I} = (\tilde{G} = (V_1, V_2, E), \mathcal{U}_1, \mathcal{U}_2, h, r)$ of (r,h)-GPwB, such that $\mathcal{I}$ has a perfect solution. Then with probability at least $1/2$ over the random choices made in the construction of the corresponding instance $\hat{\mathcal{I}}$ of NDP-Grid, there is a solution to $\hat{\mathcal{I}}$, routing $\Omega(\beta^*(\mathcal{I})/\log^3 M)$ demand pairs via a set of spaced-out paths.*

The remainder of this subsection is devoted to the proof of the theorem. We assume w.l.o.g. that $|E| = M > 2^{50}$, as otherwise, since $\beta^*(\mathcal{I}) \leq |E|$, routing a single demand pair is sufficient.

Let $((W_1, \ldots, W_r), (E_1, \ldots, E_r))$ be a perfect solution to $\mathcal{I}$. 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 $1 \leq i \leq r$, $|E_i| = h = \beta^*(\mathcal{I})/r$.

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.

Next we define a property of subsets of the demand pairs, called a *distance property*. We later show that every subset $\mathcal{M}' \subseteq \mathcal{M}^0$ of demand pairs that has this property can be routed via spaced-out paths, and that there is a large subset $\mathcal{M}' \subseteq \mathcal{M}^0$ of the demand pairs with this property.

Given a subset $\mathcal{M}' \subseteq \mathcal{M}^0$ of the demand pairs, we start by defining an ordering $\sigma_{\mathcal{M}'}$ of the destination vertices in $T(\mathcal{M}')$. This ordering is somewhat different from the ordering of the vertices of $T(\mathcal{M}')$ on row R''. We first provide a motivation and an intuition for this new ordering $\sigma_{\mathcal{M}'}$. Recall that the rows R' and R'' of $\hat{G}$, where all source and all destination vertices lie, respectively, are located at a distance at least $\ell/4$ from each other and from the grid boundaries. Let R be any row of $\hat{G}$, lying between R' and R'', at a distance at least $\ell/16$ from both R' and R''. Let X be some subset of $|\mathcal{M}'|$ vertices of R. If we index the vertices of $T(\mathcal{M}')$ as $\{t_1, t_2, \ldots, t_{|\mathcal{M}'|}\}$ according to their order in the new ordering $\sigma_{\mathcal{M}'}$, then we view the ith vertex of X, that we denote by x_i, as representing the terminal t_i. For each $1 \leq i \leq |\mathcal{M}'|$, we denote the source vertex corresponding to t_i by s_i, that is, $(s_i, t_i) \in \mathcal{M}'$. Note that the ordering of the vertices of $S(\mathcal{M}')$ on R' may be completely different from the one induced by these indices. Similarly, the ordering of the vertices of $T(\mathcal{M}')$ on $\mathcal{R}''$ may be inconsistent with this indexing. Eventually, we will construct a set $\mathcal{P}$ of spaced-out paths routing the demand pairs in $\mathcal{M}'$, so that the path $P_i \in \mathcal{P}$, connecting s_i to t_i, intersects the row R exactly once – at the vertex x_i. In this way, we will use the ordering $\sigma_{\mathcal{M}'}$ of the destination vertices in $T(\mathcal{M}')$ to determine the order in which the path of $\mathcal{P}$ intersect R.

22

Assume now that we are given some subset $\mathcal{M}' \subseteq \mathcal{M}^0$ of demand pairs. Recall that the sources and the destinations of all demand pairs in $\mathcal{M}'$ are distinct. We are now ready to define the ordering $\sigma_{\mathcal{M}'}$ of $T(\mathcal{M}')$. We partition the vertices of $T(\mathcal{M}')$ into subsets $J_1, J_2, \ldots, J_r$, as follows. Consider some vertex $v'_j \in V_2$ of $\tilde{G}$, and assume that it lies in the cluster W_i. Then all destination vertices of $T(\mathcal{M}')$ that belong to the corresponding block K'_j are added to the set J_i. To obtain the final ordering $\sigma_{\mathcal{M}'}$, we place the vertices of $J_1, J_2, \ldots, J_r$ in this order, where within each set J_i, the vertices are ordered according to their ordering along the row R''. Notice that a selection of a subset $\mathcal{M}' \subseteq \mathcal{M}_0$ completely determines the ordering $\sigma_{\mathcal{M}'}$. Given two demand pairs $(s, t), (s', t') \in \mathcal{M}'$, we let $N_{\mathcal{M}'}(s, s')$ denote the number of destination vertices that lie between t and t' in the ordering $\sigma_{\mathcal{M}'}$ (note that this is well-defined as the demand pairs in $\mathcal{M}'$ do not share their sources or destinations). Recall that $d(s, s')$ is the distance between s and s' in graph $\hat{G}$.

Definition 5 *Suppose we are given a subset $\mathcal{M}' \subseteq \mathcal{M}^0$ of the demand pairs. We say that two distinct vertices $s, s' \in S(\mathcal{M}')$ are* consecutive *with respect to $\mathcal{M}'$, iff no other vertex of $S(\mathcal{M}')$ lies between s and s' on R'. We say that $\mathcal{M}'$ has the* distance property *iff for every pair $s, s' \in S(\mathcal{M}')$ of vertices that are consecutive with respect to $\mathcal{M}'$, $N_{\mathcal{M}'}(s, s') < d(s, s')/4$.*

We first show that there is a large subset of the demand pairs in $\mathcal{M}^0$ with the distance property in the following lemma, whose proof appears in the next subsection.

Lemma 5.5 *With probability at least $1/2$ over the construction of $\hat{\mathcal{I}}$, there is a subset $\mathcal{M}' \subseteq \mathcal{M}^0$ of demand pairs that has the distance property, and $|\mathcal{M}'| = \Omega(|\mathcal{M}^0|/\log^3 M)$.*

Finally, we show that every set $\mathcal{M}'$ of demand pairs with the distance property can be routed via spaced-out paths.

Lemma 5.6 *Assume that $\mathcal{M}' \subseteq \mathcal{M}^0$ 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 5.5, since $|\mathcal{M}^0| = \beta^*(\mathcal{I})$. We prove these lemmas in the following two subsections.

5.2.1 Proof of Lemma 5.5

We assume that $|\mathcal{M}^0| > c \log^3 M$ for some large enough constant c, since otherwise we can return a set $\mathcal{M}'$ containing a single demand pair. We gradually modify the set $\mathcal{M}^0$ of the demand pairs, by selecting smaller and smaller subsets $\mathcal{M}^1, \mathcal{M}^2, \mathcal{M}^3$, and $\mathcal{M}'$. For each vertex vertex $v \in V_1 \cup V_2$ of the (r,h)-GPwB instance $\tilde{G}$, let $\delta(v)$ denote the set of all edges of $E(\tilde{G})$ incident to v.

We start by performing two "regularization" steps on the vertices of V_2 and V_1 respectively. Intuitively, we will select two integers p and q, and a large enough subset $\mathcal{M}^2 \subseteq \mathcal{M}^0$ of demand pairs, so that for every vertex $v_i \in V_1$, either no demand pair in $\mathcal{M}^2$ has its source on K_i, or roughly 2^p of them do. Similarly, for every vertex $v'_j \in V_2$, either no demand pairs in $\mathcal{M}^2$ has its destination on K'_j, or roughly 2^q of them do. We will not quite achieve this, but we will come close enough.

Step 1 [Regularizing the degrees in V_2]. In this step we select a large subset $\mathcal{M}^1 \subseteq \mathcal{M}^0$ of the demand pairs, and an integer q, such that, for each vertex $v \in V_2$, the number of edges of $E(\tilde{G})$ incident to v, whose corresponding demand pair lies in $\mathcal{M}^1$, is either 0, or roughly 2^q. In order to do this, we partition the vertices of V_2 into classes $Z_1, \ldots, Z_{\lceil \log M \rceil}$, where a vertex $v' \in V_2$ belongs to class

23

Z_y iff $2^{y-1} \leq |E^0 \cap \delta(v')| < 2^y$. If $v \in Z_y$, then we say that all edges in $\delta(v) \cap E^0$ belong to the class Z_y. Therefore, each edge of E^0 belongs to exactly one class, and there is some index $1 \leq q \leq \lceil \log M \rceil$, such that at least $\Omega(|\mathcal{M}^0|/\log M)$ edges of E^0 belong to class Z_q. We let $E^1 \subseteq E^0$ be the set of all edges that belong to the class Z_q, and we let $\mathcal{M}^1 = \mathcal{M}(E^1) \subseteq \mathcal{M}^0$ be the corresponding subset of the demand pairs.

Step 2 [Regularizing the degrees in V_1]. This step is similar to the previous step, except that it is now performed on the vertices of V_1. We partition the vertices of V_1 into classes $Y_1, \ldots, Y_{\lceil \log M \rceil}$, where a vertex $v \in V_1$ belongs to class Y_z iff $2^{z-1} \leq |E^1 \cap \delta(v)| < 2^z$. If $v \in Y_z$, then we say that all edges in $\delta(v) \cap E^1$ belong to the class Y_z. As before, every edge of E^1 belongs to exactly one class, and there is some index $1 \leq p \leq \lceil \log M \rceil$, such that at least $\Omega(|E^1|/\log M) \geq \Omega(|\mathcal{M}^0|/\log^2 M)$ edges of E^1 belong to the class Y_p. We let $E^2 \subseteq E^1$ denote the set of all edges that belong to class Y_p, and $\mathcal{M}^2 = \mathcal{M}(E^2) \subseteq \mathcal{M}^1$ denote the corresponding subset of the demand pairs, so that $|\mathcal{M}^2| = \Omega(|\mathcal{M}^0|/\log^2 M)$.

Notice that so far, for every vertex $v \in V_1$, if $\delta(v) \cap E^2 \neq \emptyset$, then $2^{p-1} \leq |\delta(v) \cap E^2| < 2^p$. However, for a vertex $v \in V_2$ with $\delta(v) \cap E^2 \neq \emptyset$, we are only guaranteed that $|\delta(v) \cap E^2| < 2^q$, since we may have discarded some edges that were incident to v from E^1. Moreover, the subset $\mathcal{M}^2$ of the demand pairs is completely determined by the solution $((W_1, \ldots, W_r), (E_1, \ldots, E_r))$ to the (r,h)-GPwB problem, and is independent of the random choices made in our construction of the NDP-Grid problem instance. The following observation will be useful for us later.

Observation 5.7 $h \geq 2^p \cdot 2^q/4$.

Proof: From our assumption that $|\mathcal{M}^0| > c \log^3 M$, $E^2 \neq \emptyset$. Therefore, there must be an index $1 \leq i \leq r$ with $E_i \cap E^2 \neq \emptyset$. Fix any such index i. Then there is a vertex $v'_j \in W_i \cap V_2$, such that at least one edge of $\delta(v'_j)$ belongs to E^2. But then, from the definition of E^2, at least 2^{q-1} edges of $\delta(v'_j)$ belong to E^2. Assume without loss of generality that these edges connect v'_j to vertices $v_1, \ldots, v_{2^{q-1}} \in V_1$. All these vertices must also belong to W_i, and for each $1 \leq x \leq 2^{q-1}$, vertex v_x has at least one edge in $\delta(v_x) \cap E^2$. From our definition of E^2 and E^1, at least 2^{p-1} edges of $\delta(v_x)$ belonged to E^1. Therefore, $|E^1 \cap E_i| \geq 2^{q-1} \cdot 2^{p-1}$. But $|E_i| \leq h$, and so $h \geq 2^p \cdot 2^q/4$. $\qquad \square$

For each vertex $v_i \in V_1$, let $X_i \subseteq V(K_i)$ be the set of all vertices that serve as the sources of the demand pairs in $\mathcal{M}$, so $X_i = S(\mathcal{M}) \cap V(K_i)$. Recall that $|X_i| = \beta(v_i)$, and every pair of vertices in X_i is separated by at least $512 \left\lceil \frac{h \log M}{\beta(v_i)} \right\rceil$ vertices of K_i. We let $X'_i \subseteq X_i$ denote the subset of vertices that serve as sources of the demand pairs in $\mathcal{M}^2$. We say that a sub-path $Q \subseteq K_i$ is *heavy* iff $|V(Q)| = \left\lfloor \frac{512 h \log^2 M}{2^p} \right\rfloor$, and $|V(Q) \cap X'_i| > 16 \log M$.

Observation 5.8 *With probability at least* 0.99 *over the choice of the random permutation* ρ', *for all* $v_i \in V_1$, *no heavy sub-path* $Q \subseteq K_i$ *exists.*

Proof: Since $|V_1| \leq M$, from the union bound, it is enough to prove that for a fixed vertex $v_i \in V_1$, the probability that a heavy sub-path $Q \subseteq K_i$ exists is at most $1/(100M)$. We now fix some vertex $v_i \in V_1$. Observe that K_i may only contain a heavy sub-path if $\beta(v_i) = |X_i| \geq 16 \log M$. We call the vertices of X'_i pink, and the remaining vertices of X_i yellow. Let P denote the number of the pink vertices. Then $2^{p-1} \leq P < 2^p$. Let $\mu = 4 \log M$, and let $\mathcal{E}_i$ be the bad event that there is a set of $\lfloor |X_i| \mu/P \rfloor$ consecutive vertices of X_i, such that at least 4μ of them are pink.

Observe that the selection of the pink vertices only depends on the solution to the (r,h)-GPwB problem, and is independent of our construction of the NDP-Grid instance. The ordering of the vertices in X_i is

determined by the permutation ρ' of $\mathcal{U}_2$, and is completely random. Therefore, from Lemma 5.3, the probability of $\mathcal{E}_i$ is at most $|X_i|/4^\mu \leq M/4^{4\log M} \leq 1/M^7$.

Let $\mathcal{E}_i'$ be the event that some sub-path Q of K_i is heavy. We claim that $\mathcal{E}_i'$ may only happen if event $\mathcal{E}_i$ happens. Indeed, consider some sub-path Q of K_i, and assume that it is heavy. Recall that Q contains $\left\lfloor \frac{512h\log^2 M}{2^p} \right\rfloor$ vertices. Since every pair of vertices in X_i is separated by at least $512\left\lceil \frac{h\log M}{\beta(v_i)} \right\rceil$ vertices, we get that:

$$|V(Q) \cap X_i| \leq \frac{\left\lfloor 512h\log^2 M/2^p \right\rfloor}{512\left\lceil h\log M/\beta(v_i) \right\rceil} + 1 \leq \frac{\beta(v_i)\log M}{2^p} + 1 \leq \left\lfloor \frac{|X_i|\mu}{P} \right\rfloor,$$

as $2^{p-1} \leq P < 2^p$. Since Q is heavy, at least $4\mu = 16\log M$ of the vertices of $V(Q) \cap X_i$ belong to X_i', that is, they are pink. Therefore, there is a set of $\lfloor |X_i|\mu/P \rfloor$ consecutive vertices of X_i, out of which 4μ are pink, and $\mathcal{E}_i$ happens. We conclude that $\mathbf{Pr}\left[\mathcal{E}_i'\right] \leq \mathbf{Pr}\left[\mathcal{E}_i\right] \leq 1/M^7$, and overall, since we have assumed that $M > 2^{50}$, the probability that a heavy path exists in any block K_i is bounded by 0.99 as required. $\square$

Let $\mathcal{E}_{\mathsf{bad}}$ be the bad event that for some $v_i \in V_1$, block K_i contains a heavy path. From Observation 5.8, the probability of $\mathcal{E}_{\mathsf{bad}}$ is at most 0.01.

The following claim will be used to bound the values $N_{\mathcal{M}^2}(s, s')$.

Claim 5.9 *Consider some vertex $v_j \in V_1$ in graph $\tilde{G}$, and the block K_j representing it. Then with probability at least $(1 - 1/M^3)$, for every pair $s, s' \in S(\mathcal{M}^2) \cap V(K_j)$ of source vertices that are consecutive with respect to $\mathcal{M}^2$, $N_{\mathcal{M}^2}(s, s') \leq 128h\log M/2^p$.*

Proof: Fix some vertex $v_j \in V_1$ and consider the block K_j representing it. Assume that v_j belongs to the cluster W_i in our solution to the (r, h)-GPwB problem. Let $s, s' \in S(\mathcal{M}^2) \cap V(K_j)$ be a pair of source vertices that are consecutive with respect to $\mathcal{M}^2$. Recall that we have defined a subset $J_i \subseteq T(\mathcal{M}^2)$ of destination vertices, that appear consecutively in the ordering $\sigma_{\mathcal{M}^2}$, and contain all vertices of $T(\mathcal{M}^2)$, that lie in blocks $K_{j'}'$, whose corresponding vertices $v_{j'} \in V_2 \cap W_i$.

Let $A = W_i \cap V_2$; let $A' \subseteq A$ contain all vertices that have an edge of $E^2 \cap E_i$ incident to them; and let $A'' \subseteq A'$ contain all vertices that have an edge of $E^2 \cap E_i$ connecting them to v_j. Since the solution to the (r, h)-GPwB problem instance is perfect, every vertex of A (and hence A' and A'') belongs to a distinct group of $U \in \mathcal{U}_2$. We denote by $\mathcal{U}' \subseteq \mathcal{U}_2$ the set of all groups to which the vertices of A' belong, and we define $\mathcal{U}'' \subseteq \mathcal{U}'$ similarly for A''. Consider now some group $U \in \mathcal{U}'$, and let v_a' be the unique vertex of U that belongs to A'. We denote by $C(U)$ the set of all vertices of the corresponding block K_a' that belong to $T(\mathcal{M}^2)$. Therefore, we now obtain a partition $\{C(U)\}_{U \in \mathcal{U}'}$ of all vertices of J_i into subsets, where each subset contains at most 2^q vertices. Moreover, in the ordering $\sigma_{\mathcal{M}^2}$, the vertices of each such set $C(U)$ appear consecutively, in the order of their appearance on R'', while the ordering between the different sets $C(U)$ is determined by the ordering of the corresponding groups U in ρ'. Let ρ'' be the ordering of the groups in $\mathcal{U}'$ induced by ρ', so that ρ'' is a random ordering of $\mathcal{U}'$. Observe that, since the choice of the set $\mathcal{M}^2$ is independent of the ordering ρ' (and only depends on the solution to the (r, h)-GPwB problem instance), so is the choice of the sets $\mathcal{U}'$ and $\mathcal{U}''$.

Let t and t' be the destination vertices that correspond to s and s', respectively, that is, $(s, t), (s', t') \in \mathcal{M}^2$. Assume that $t \in K_z'$ and $t' \in K_{z'}'$, where v_z' and $v_{z'}'$ are vertices of V_2. From our definition, both v_z' and $v_{z'}'$ must belong to the set A''. Assume that v_z' belongs to the group U' in $\mathcal{U}_2$, while $v_{z'}'$ belongs to group U''. Again, from our definitions, both $U', U'' \in \mathcal{U}''$. From the above discussion, if the number of groups $U \in \mathcal{U}'$ that fall between U' and U'' is γ, then the number of destination vertices lying between t and t' in $\sigma_{\mathcal{M}^2}$ is at most $2^q \cdot (\gamma + 2)$. Therefore, it is now enough to bound the value of

γ. In order to do so, we think of the groups of $\mathcal{U}''$ as pink, and the remaining groups of $\mathcal{U}'$ as yellow. Let P denote the total number of all pink groups, and let $n^* = |\mathcal{U}'|$. From the construction of $\mathcal{M}^2$, $P = |S(\mathcal{M}^2) \cap V(K_j)| \geq 2^{p-1}$. We use the following observation to upper-bound n^*.

Observation 5.10 $n^* \leq h/2^{q-1}$.

Proof: Since we have started with a perfect solution to $\mathcal{I}$, for each group $U \in \mathcal{U}'$, there is exactly one vertex of U in W_i. Due to Step 1 of regularization, each such vertex contributed at least 2^{q-1} edges to $E^1 \cap E_i$, while $|E^1 \cap E_i| \leq h$. Therefore, $n^* \leq h/2^{q-1}$. $\qquad\square$

Let $\mu = 4 \log M$, and let $\mathcal{E}_Y$ be the event that there are at least $\lceil 4n^*\mu/P \rceil$ consecutive yellow groups in the ordering ρ'' of $\mathcal{U}'$. From Lemma 5.2, the probability of $\mathcal{E}_Y$ is at most $n^*/e^\mu \leq M/e^\mu \leq 1/M^3$. If event $\mathcal{E}_Y$ does not happen, then the length of the longest consecutive sub-sequence of ρ'' containing only yellow groups is bounded by:

$$\left\lceil \frac{4n^*\mu}{P} \right\rceil \leq \frac{64h \log M}{2^q 2^p}.$$

Assume now that $\mathcal{E}_Y$ does not happen, and consider any two vertices $s, s' \in S(\mathcal{M}^2) \cap V(K_j)$ that are consecutive with respect to $\mathcal{M}^2$. Assume that their corresponding destination vertices are t and t' respectively, and that t and t' belong to the groups U and U', respectively. Then U and U' are pink groups. Moreover, since the ordering of the vertices of $S(\mathcal{M}) \cap V(K_j)$ on R' is identical to the ordering of the groups of $\mathcal{U}''$ to which their destinations belong in ρ', no other pink group appears between U and U' in ρ''. Therefore, at most $\frac{64h \log M}{2^q 2^p}$ groups of $\mathcal{U}'$ lie between U and U' in ρ''. Recall that for each group $U'' \in \mathcal{U}'$, at most one vertex $v'_z \in U''$ belongs to W_i, and that the corresponding block K'_z may contribute at most 2^q destination vertices to $T(\mathcal{M}^2)$. We conclude that the number of vertices separating t from t' in $\sigma_{\mathcal{M}_3}$ is bounded by:

$$\left(\frac{64h \log M}{2^q 2^p} + 2 \right) \cdot 2^q \leq \frac{128h \log M}{2^p}.$$

(We have used Observation 5.7). Therefore, if event $\mathcal{E}_Y$ does not happen, then for every pair $s, s' \in S(\mathcal{M}^2) \cap V(K_j)$ of vertices that are consecutive with respect to $\mathcal{M}^2$, $N_{\mathcal{M}^2}(s, s') \leq \frac{128h \log M}{2^p}$. $\qquad\square$

Let $\mathcal{E}'_{\mathsf{bad}}$ be the bad event that for some vertex $v_j \in V_1$, for some pair $s, s' \in S(\mathcal{M}^2) \cap V(K_j)$ of source vertices that are consecutive with respect to $\mathcal{M}^2$, $N_{\mathcal{M}^2}(s, s') > 128h \log M/2^p$. By applying the Union Bound to the result of Claim 5.9, we get that the probability of $\mathcal{E}'_{\mathsf{bad}}$ is at most $1/M^2$. Notice that the probability that neither $\mathcal{E}_{\mathsf{bad}}$ nor $\mathcal{E}'_{\mathsf{bad}}$ happen is at least $1/2$. We assume from now on that this is indeed the case, and show how to compute a large subset $\mathcal{M}' \subseteq \mathcal{M}^2$ of the demand pairs that has the distance property. This is done in the following two steps.

Step 3 [Sparsifying the Sources.] Assume that $\mathcal{M}^2 = \{(s_1, t_1), \ldots, (s_{|\mathcal{M}^2|}, t_{|\mathcal{M}^2|})\}$, where the demand pairs are indexed according to the left-to-right ordering of their sources on R', that is $s_1, s_2, \ldots, s_{|\mathcal{M}^2|}$ appear on R' in this order. We now define:

$$\mathcal{M}^3 = \{(s_i, t_i) \mid i = 1 \mod \lceil 32 \log M \rceil\}.$$

Let $E^3 \subseteq E^2$ be the set of edges of $\tilde{G}$ whose corresponding demand pairs belong to $\mathcal{M}^3$. It is easy to verify that $|\mathcal{M}^3| \geq \Omega(|\mathcal{M}^2|/\log M) = \Omega(|\mathcal{M}^0|/\log^3 M)$.

We also obtain the following claim.

Claim 5.11 *Assume that events $\mathcal{E}_{\mathsf{bad}}$ and $\mathcal{E}'_{\mathsf{bad}}$ did not happen. Then for each $1 \leq j \leq N_1$, for every pair $s, s' \in S(\mathcal{M}^3) \cap V(K_j)$ of source vertices, $N_{\mathcal{M}^3}(s, s') \leq 128 d(s, s')$.*

Proof: Fix some $1 \leq j \leq N_1$, and some pair $s, s' \in S(\mathcal{M}^3) \cap V(K_j)$ of vertices. Let $S' = \{s_1, s_2, \ldots, s_z\}$ be the set of all vertices of $S(\mathcal{M}^2)$ that appear between s and s' on R'. Assume w.l.o.g. that s lies to the left of s' on R', and denote $s_0 = s$ and $s_{z+1} = s'$. Assume further that the vertices of S' are indexed according to their left-to-right ordering on R'. Note that, from the definition of $\mathcal{M}^3$, $z \geq \lceil 32 \log M \rceil$ must hold.

Let $I \subseteq K_j$ be the sub-path of K_j between s and s'. We partition I into paths containing $\lfloor 512 h \log^2 M / 2^p \rfloor$ vertices each, except for the last path that may contain fewer vertices. Since no such path may be heavy, we obtain at least $n' = \left\lfloor \frac{z}{16 \log M} \right\rfloor$ disjoint sub-paths of I, each of which contains $\lfloor 512 h \log^2 M / 2^p \rfloor$ vertices. We conclude that:

$$d(s, s') \geq n' \cdot \lfloor 512 h \log^2 M / 2^p \rfloor - 1 \geq \left\lfloor \frac{z}{16 \log M} \right\rfloor \cdot \lfloor 512 h \log^2 M / 2^p \rfloor - 1 \geq \frac{2zh \log M}{2^p}.$$

On the other hand, since $\sigma_{\mathcal{M}^3}$ is the same as the ordering of $T(\mathcal{M}^3)$ induced by $\sigma_{\mathcal{M}^2}$, we get that:

$$N_{\mathcal{M}^3}(s, s') \leq N_{\mathcal{M}^2}(s, s') \leq \sum_{i=0}^{z} N_{\mathcal{M}^2}(s_i, s_{i+1}) \leq (z+1) \cdot \frac{128 h \log M}{2^p} \leq 128 d(s, s').$$

$\square$

Step 4 [Decreasing the Values $N_{\mathcal{M}^3}(s, s')$]. We are now ready to define the final set $\mathcal{M}' \subseteq \mathcal{M}^3$ of demand pairs.

Assume that $\mathcal{M}^3 = \left\{ (s_1, t_1), \ldots, (s_{|\mathcal{M}^3|}, t_{|\mathcal{M}^3|}) \right\}$, where the demand pairs are indexed according to the ordering of their destinations in $\sigma_{\mathcal{M}^3}$ (notice that this is the same as the ordering of $T(\mathcal{M}^3)$ induced by $\sigma_{\mathcal{M}^2}$). We now define:

$$\mathcal{M}' = \{ (s_i, t_i) \mid i = 1 \mod 512 \}.$$

We claim that $\mathcal{M}'$ has the distance property. Indeed, consider any two pairs $(s, t), (s', t') \in \mathcal{M}'$, such that s and s' are consecutive with respect to $\mathcal{M}'$. If s and s' lie in different blocks K_j, then, since the distance between any such pair of blocks is at least $10M$, while $|\mathcal{M}'| \leq M$, we get that $N_{\mathcal{M}'}(s, s') \leq d(s, s')/4$. Otherwise, both s and s' belong to the same block K_j. But then it is easy to verify that $N_{\mathcal{M}'}(s, s') \leq N_{\mathcal{M}^3}(s, s')/512 \leq d(s, s')/4$ from Claim 5.11. We conclude that with probability at least $1/2$, neither of the events $\mathcal{E}_{\mathsf{bad}}, \mathcal{E}'_{\mathsf{bad}}$ happens, and in this case, $\mathcal{M}'$ has the distance property.

5.2.2 Proof of Lemma 5.6

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 $\ell/4$ form each other and from the top and the bottom boundaries of $\hat{G}$, where $\ell > M^2$ is the dimension of the grid. Let R be any row lying between R' and R'', within distance at least $\ell/16$ 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 the ordering of their destination vertices in $\sigma_{\mathcal{M}'}$. To recap, the ordering $\sigma_{\mathcal{M}'}$ of $T(\mathcal{M}')$ was defined

as follows. We have defined a partition $(J_1, \ldots, J_r)$ of the vertices of $T(\mathcal{M}')$ into subsets, where each set J_i represents a cluster W_i in our solution to the (r,h)-GPwB problem instance, and contains all destination vertices $t \in T(\mathcal{M}')$ that lie in blocks K_j', for which the corresponding vertex $v_j \in V_2$ belongs to W_i. The ordering of the destination vertices inside each set J_i is the same as their ordering on R'', and the different sets $J_1, \ldots, J_r$ are ordered in the order of their indices.

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 sub-grids: a top sub-grid G^t spanned by all rows that appear above R (including R), and a bottom sub-grid 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 sub-grid $\hat{K}_j$ 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 sub-grid 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 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 turn. 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 5.12 *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 $Y_j = \left\{ s_{i_1}, s_{i_2}, \ldots, s_{i_{M_j}} \right\}$, where the indexing of the vertices of Y_j is consistent with the indexing of the vertices of $S(\mathcal{M}')$ that we have defined above, 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 5.1, 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 $1 \le z < M_j$, $i_{z+1} - i_z - 1 = N_{\mathcal{M}'}(s_{i_z}, s_{i_{z+1}}) \le d(s_{i_z}, s_{i_{z+1}})/4$. 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. $\qquad\square$

Claim 5.13 *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 $1 \leq i \leq M'$, path P_i originates from vertex x_i, and for all $1 \leq j \leq N_1$, it contains the ith column of $\mathcal{W}_j$; in particular, it contains s_i.*

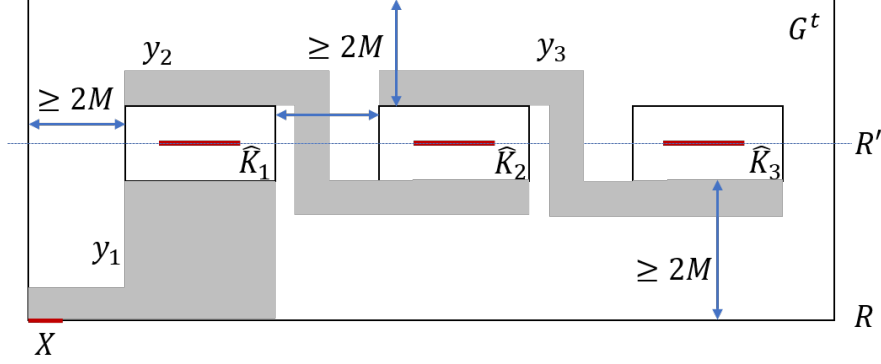We defer the proof of the claim to Section C of the appendix; see Figure 4 for an illustration of the routing.

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 $1 \leq i \leq M'$, 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 5.14 *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 $Y'_j = \left\{ t_{i_1}, t_{i_2}, \ldots, t_{i_{M_j}} \right\}$, where the indexing of the vertices of Y'_j is consistent with the indexing of the vertices of $T(\mathcal{M}')$ that we have defined above, 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_zth column of $\mathcal{W}'_j$ from the left contains the destination vertex t_{i_z}.*

Proof: From our construction of $\sigma_{\mathcal{M}'}$, the vertices $t_{i_1}, t_{i_2}, \ldots, t_{i_{M_j}}$ appear consecutively in this order in $\sigma_{\mathcal{M}'}$. Moreover, from the construction of $\mathcal{M}$, every pair of these destination vertices 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. □

The proof of the following claim is identical to the proof of Claim 5.13 and is omitted here.

Claim 5.15 *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 $1 \le i \le M'$, path P'_i originates from vertex x_i, and for all $1 \le j \le N_2$, it contains the ith 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 $1 \le i \le M'$, 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 $\mathcal{M}'$.

5.3   From Routing to Partitioning

The goal of this subsection is to prove the following theorem, that will complete the proof of Theorem 3.2.

Theorem 5.16 *There is a deterministic efficient algorithm, that, given a valid 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 (random) 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 $\Omega(|\mathcal{P}^*|/\log^3 M)$.*

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 sub-graph 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 $|\mathcal{P}^*| \le 2^{64} h \log^3 M$, 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 $\left\lceil \frac{|\mathcal{P}^*|}{2^{64} \log^3 M} \right\rceil \le h$ 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 sub-graphs, 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 *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 iff $|E(A)|, |E(B)| \ge \rho \cdot |E(\mathbf{H})|$.*

The following theorem is central to the proof of Theorem 5.16.

Theorem 5.17 *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 5.17 below, after we complete the proof of Theorem 5.16 using it. Our algorithm maintains a collection $\mathcal{G}$ of disjoint vertex-induced sub-graphs 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, 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 5.17 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 5.18 *Let $\mathcal{G}'$ be the final set of disjoint sub-graphs 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 sub-graph $\mathbf{H} \subseteq \tilde{G}'$ that belonged to $\mathcal{G}$ over the course of the algorithm, tree τ contains a vertex $v(\mathbf{H})$. The root of the tree is the vertex $v(\tilde{G}')$. If, over the course of our algorithm, we have partitioned the graph $\mathbf{H}$ into two disjoint vertex-induced sub-graphs $\mathbf{H}'$ and $\mathbf{H}''$, then we add an edge from $v(\mathbf{H})$ to $v(\mathbf{H}')$ and to $v(\mathbf{H}'')$, that become the children of $v(\mathbf{H})$ in τ.

The *level* of a vertex $v(\mathbf{H})$ 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 vertices of the tree τ lying at level i. Let $\hat{E}_i = \bigcup_{v(\mathbf{H}) \in V_i} E(\mathbf{H})$ be the set of all edges contained in all sub-graphs of $\tilde{G}'$, whose corresponding vertex belongs to level i. Finally, let $m_i = |E_i|$. 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, $|E(\mathbf{H})| \geq h$ — 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. The $|E(\mathbf{H}')| \geq 2^{64} h \log^3 M$, 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 $|\mathcal{G}'| \leq r$. $\square$

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 sub-graphs 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 5.18, $z \leq r$. 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:

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

31

from Claim 5.18. In order to complete the proof of Theorem 5.16, it now remains to prove Theorem 5.17.

Proof of Theorem 5.17. 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 7 *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 no three curves meet at the same point; 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 5.19 *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 think of the grid $\hat{G}$ underlying the NDP-Grid instance $\hat{\mathcal{I}}$ as the drawing board, and map the vertices of $\mathbf{H}$ to points inside some of its carefully selected cells. 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'_i, 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 $|V(K_i)| \leq 1024 \cdot \lceil h\log M\rceil$, 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:

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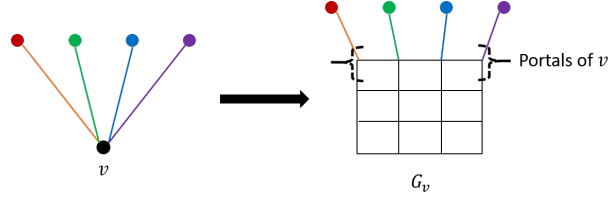
$\square$

Figure 5: Grid Q_v obtained from v

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 5.20 *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 $\mathrm{cr} \leq md\alpha$ 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 $|E(A, B)| \leq 64\sqrt{8md\alpha}$.*

Before we complete the proof of the lemma, we show that the proof of Theorem 5.17 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, $h \geq \beta(v)$, and so the maximum vertex degree in $\mathbf{H}$ is bounded by $d = h$. From Lemma 5.19, the drawing φ of $\mathbf{H}$ has at most $2048|E(\mathbf{H})|\lceil h \log M \rceil \leq 2^{12}md \log M$ 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 5.20 to graph $\mathbf{H}$ to obtain a $1/32$-edge-balanced cut (A, B) with $|E(A, B)| \leq 64\sqrt{8md\alpha} \leq 64\sqrt{2^{15}mh \log M} \leq 64\sqrt{m^2/(2^{49} \log^2 M)} \leq \frac{|E(\mathbf{H})|}{64 \log M}$, since we have assumed that $|E(\mathbf{H})| = m > 2^{64}h \log^3 M$. It now remains to prove Lemma 5.20.

Proof of Lemma 5.20. 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 5.21 $|V(\hat{\mathbf{H}})| \leq (2m + d)d$.

Proof: Clearly, $|V(\hat{\mathbf{H}})| = \sum_{v \in V(\mathbf{H})} d_v^2$. Observe that for every pair $a \geq b \geq 0$ of integers, $(a + 1)^2 + (b - 1)^2 \geq a^2 + b^2$. Since maximum vertex degree in $\mathbf{H}$ is bounded by d, the sum is maximized when

33

all but possibly one summand are equal to d, and, since $\sum_{v \in \mathbf{H}} d_v = 2m$, there are at most $\lceil 2m/d \rceil$ summands. Therefore, $|V(\hat{\mathbf{H}})| \leq \lfloor 2m/d \rfloor \cdot d^2 + d^2 \leq (2m + d)d$. $\qquad\qquad\qquad\square$

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 $|V(\hat{\mathbf{H}}')| \leq |V(\hat{\mathbf{H}})| + \mathrm{cr} \leq (2m + d)d + md\alpha \leq 4md\alpha$, 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 [Ung51, LT79, AST94].

Theorem 5.22 ([LT79]) *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;*

- $\sum_{v \in A} w(v), \sum_{v \in B} w(v) \leq 2W/3$*; and*

- $|X| \leq 2\sqrt{2n}$*.*

We apply Theorem 5.22 to graph $\hat{\mathbf{H}}'$, to obtain a partition (A, X, B) of $V(\hat{\mathbf{H}}')$, with $|X| \leq 2\sqrt{2|V(\hat{\mathbf{H}}')|} \leq 2\sqrt{8md\alpha}$. Since $W = \sum_{v \in V(\hat{\mathbf{H}}')} w(v) = 2m$, we get that $|A \cap \Pi| = \sum_{v \in A} w(v) \leq 2W/3 \leq 4m/3$, and similarly $|B \cap \Pi| \leq 4m/3$. Assume without loss of generality that $|A \cap \Pi| \leq |B \cap \Pi|$. We obtain a bi-partition (A', B') of $V(\hat{\mathbf{H}}')$ by setting $A' = A \cup X$ and $B' = B$. Since $|X| \leq 2\sqrt{8md\alpha} \leq m/3$ (as $m > 2^{20} d\alpha$), we are guaranteed that $|A' \cap \Pi|, |B' \cap \Pi| \leq 4m/3$ holds. Moreover, as all vertex degrees in $\hat{\mathbf{H}}'$ are at most 4, $|E(A', B')| \leq 4|X| \leq 8\sqrt{8md\alpha}$.

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 5.23 *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 $|E_Q(X, Y)| \geq \min\{|U \cap X|, |U \cap Y|\}$.*

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. $\qquad\square$

We say that a vertex $v \in V(\mathbf{H})$ is *split* by the cut (A', B') iff $V(Q_v) \cap A'$ and $V(Q_v) \cap B' \neq \emptyset$. We say that it is *split evenly* iff $|\Pi(v) \cap A'|, |\Pi(v) \cap B'| \geq d_v/8$; 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 5.23, $|E(A', B')|$ does not increase, since $E(Q_v)$ contributed at least $|\Pi(v) \cap B'|$ regular edges to the cut before the current iterations. 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 $|\Gamma(v)| \leq d < m/21$, $|A' \cap \Pi| \leq \frac{32}{21}m$, while $|B' \cap \Pi| \leq \frac{32}{21}m$ as before. Moreover, from Observation 5.23, 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,

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

$\square$

$\square$

6 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 3.2.

Theorem 6.1 *There is a constant $c^* > 0$, and there is an efficient randomized algorithm, that, given a valid instance $\mathcal{I} = (\tilde{G}, \mathcal{U}_1, \mathcal{U}_2, h, r)$ of (r,h)-GPwB with $|E(\tilde{G})| = M$, 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 with probability at least $\frac{1}{2}$ over the construction of $\hat{\mathcal{I}}'$, instance $\hat{\mathcal{I}}'$ has a solution $\mathcal{P}'$ that routes at least $\frac{\beta^*}{c^* \log^3 M}$ demand pairs via node-disjoint paths; and*

- *There is a deterministic efficient algorithm, 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 6.1 into the hardness of approximation proof instead of Theorem 3.2, we extend the hardness result to the NDP problem on wall graphs and complete the proof of Theorem 1.1.

Proof of Theorem 6.1. Let $\hat{\mathcal{I}} = (\hat{G}, \mathcal{M})$ be the instance of NDP-Grid constructed in Theorem 3.2. 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 deterministic efficient algorithm from Theorem 3.2 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. Let $\mathcal{E}$ be the good event that the instance $\hat{\mathcal{I}}$ of NDP-Grid has a solution $\mathcal{P}$ that routes at least $\frac{\beta^*}{c^* \log^3 M}$ demand pairs via paths that are spaced-out. From Theorem 3.2, event $\mathcal{E}$ happens with probability at least $\frac{1}{2}$. It is now enough to show that whenever event $\mathcal{E}$ happens, there is a solution of value $\frac{\beta^*}{c^* \log^3 M}$ 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 6.2 *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 OPT$' \geq$ 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 $\Omega(|\mathcal{P}'|)$.*

The following corollary immediately follows from Claim 6.2 and completes the proof of Theorem 1.2.

Corollary 6.3 *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 $f(n) + \text{poly}(n)$.*

It now remains to prove Claim 6.2.

Proof: The assertion that OPT$' \geq$ 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 sub-graph $H' \subseteq H$ of H contains a vertex of total degree at most 8. We construct a set U of vertices, such that no two vertices of U are connected by an edge, using a standard greedy algorithm: while $H \neq \emptyset$, select a vertex $v \in H$ 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 2.4

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 $1 \leq j \leq \ell$, and assume that $b_j = z$, for some $1 \leq z \leq 6$. 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 $1 \leq j \leq \ell$, 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 $1 \leq j \leq \ell$, $e_j = (v_j, u_j)$. Let c_j, c'_j are the assignments to v_j and u_j given by the jth coordinate of A, so that $c_j \neq c'_j$. 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 $1 \leq j \leq \ell$, $b^*_j = z_j$. Then $f_{b^*}(Q) = A$, and for all $b \neq b^*$, $f_b(Q) \neq A$.

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 $1 \leq j \leq \ell$, 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 $1 \leq j \leq \ell$, $b_j \in Z_j$, then $f_b(Q') = A'$; otherwise, $f_b(Q') \neq A'$. Therefore, the total number of vectors $b \in B$, for which

$f_b(Q') = A'$ is exactly 2^ℓ.

B Auxiliary Lemmas

The goal of this section is to prove Lemma 5.2 and Lemma 5.3. We will use the following simple observation.

Observation B.1 *For any two positive integers a and b, $\frac{b-1}{a+b-1} < \frac{b}{a+b}$.*

Recall that we are given a set U of n items, such that P of the items are pink, and the remaining $Y = n - P$ items are yellow. We consider a random permutation π of these items. Given a set $S \subseteq \{1, \ldots, n\}$ of δ indices, we let $\mathcal{E}(S)$ be the event that for all $i \in S$, the item of π located at the ith position is yellow.

Claim B.2 $\mathbf{Pr}\left[\mathcal{E}(S)\right] \leq \left(\frac{Y}{n}\right)^{|S|}$.

Proof:

As every subset of δ items of U is equally likely to appear at the indices of S, we get that:

$$\mathbf{Pr}\left[\mathcal{E}(S)\right] = \frac{\binom{Y}{\delta}}{\binom{P+Y}{\delta}} = \frac{Y \cdot (Y-1) \cdots (Y-\delta+1)}{(P+Y) \cdot (P+Y-1) \cdots (P+Y-\delta+1)} \leq \left(\frac{Y}{P+Y}\right)^\delta = \left(\frac{Y}{n}\right)^{|S|}.$$

(the last inequality follows from Observation B.1). □

We now turn to prove Lemma 5.2.

Lemma B.3 (Restatement of Lemma 5.2.) *For any $\log n \leq \mu \leq Y$, the probability that there is a sequence of $\lceil 4n\mu/P \rceil$ consecutive items in π that are all yellow, is at most n/e^μ.*

Proof: Let $\delta = \left\lceil \frac{4n\mu}{P} \right\rceil$. Consider a set S of δ consecutive indices of $\{1, \ldots, n\}$. From Claim B.2, the probability that all items located at the indices of S are yellow is at most:

$$\left(\frac{Y}{n}\right)^\delta = \left(1 - \frac{P}{n}\right)^{\lceil 4n\mu/P \rceil} \leq \left(1 - \frac{P}{n}\right)^{4n\mu/P} \leq e^{-\mu}.$$

Since there are at most n possible choices of a set S of δ consecutive indices, from the Union Bound, the probability that any such set only contains yellow items is bounded by n/e^μ. □

Lemma B.4 (Restatement of Lemma 5.3.) *For any $\log n \leq \mu \leq P$, the probability that there is a set S of $\left\lfloor \frac{n\mu}{P} \right\rfloor$ consecutive items in π, such that more than 4μ of the items are pink, is at most $n/4^\mu$.*

Proof:

Let $x = \left\lfloor \frac{n\mu}{P} \right\rfloor$, and let S be any set of x consecutive indices of $\{1, \ldots, n\}$. Denote $\delta = \lceil 4\mu \rceil$, and let $S' \subseteq S$ be any subset of δ indices from S. From Claim B.2 (by reversing the roles of the pink and the yellow items), the probability that all items located at the indices of S' are pink is at most $\left(\frac{P}{n}\right)^\delta$. Since there are $\binom{x}{\delta}$ ways to choose the subset S' of S, from the Union Bound, the probability that at least δ items of S are pink is at most:

$$\left(\frac{P}{n}\right)^{\delta} \cdot \binom{x}{\delta} \leq \left(\frac{P}{n}\right)^{\delta} \cdot \left(\frac{ex}{\delta}\right)^{\delta}$$
$$\leq \left(\frac{P}{n}\right)^{\delta} \cdot \left(\frac{e \cdot n\mu}{P\delta}\right)^{\delta}$$
$$= \left(\frac{e \cdot \mu}{\lceil 4\mu \rceil}\right)^{\lceil 4\mu \rceil}$$
$$\leq \left(\frac{e}{4}\right)^{4\mu} < \frac{1}{4\mu}.$$

Since there are at most n sets S of x consecutive items, taking the Union Bound over all such sets completes the proof. $\qquad\square$

C Proof of Claim 5.13

The goal of this section is to prove Claim 5.13. 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 $1 \leq i \leq M'$ and $1 \leq j \leq N_1$, 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 $1 \leq j \leq N_1$, 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 $1 \leq i \leq M'$, 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^i_j, respectively.

For convenience, we also denote $I''_0 = R$, and, for each $1 \leq i \leq M'$, $x''(0, i) = x_i$. The following claim is central to our proof.

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

- *For each $1 \leq j \leq N_1$ and $1 \leq i \leq M'$, path P^i_j 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 5.13. 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*, iff 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* iff $\Upsilon \cap \Upsilon' \neq \emptyset$. We are now ready to define snakes.

A snake $\mathcal{Y}$ of length z is a sequence $(\Upsilon_1, \Upsilon_2, \ldots, \Upsilon_z)$ of z corridors that are pairwise internally disjoint, such that for all $1 \leq z', z'' \leq z$, $\Upsilon_{z'}$ 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) $\min_{1 \leq z' < z}\{|\Upsilon_{z'} \cap \Upsilon_{z'+1}|\}$; and (ii) $\min_{1 \leq z' \leq z}\{w(\Upsilon_{z'})\}$. 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 $u \in \mathcal{Y}$, iff $u \in \bigcup_{z'=1}^{z} \Upsilon_{z'}$. We use the following simple claim, whose proof can be found, e.g. in [CKN17b].

Claim C.2 *Let $\mathcal{Y} = (\Upsilon_1, \ldots, \Upsilon_z)$ be a snake of width w, and let A and A' be two sets of vertices with $|A| = |A'| \leq w - 2$, 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* iff 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 $A = \{a_1, \ldots, a_{2r-1}\}$, 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 $A' = \{a'_1, \ldots, a'_{2r-1}\}$ 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 sub-graphs: 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 sub-graphs, 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'. $\qquad\qquad\square$

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 $P_j^i \in \mathcal{P}_j$ connecting $x''(j-1, i)$ to $x'(j, i)$. For each $1 \leq i \leq M'$, let P_i be the path obtained by concatenating the paths $\{P_1^i, \mathcal{W}_1^i, P_2^i, \ldots, P_{N_1}^i, \mathcal{W}_{N_1}^i\}$. The final set of paths is $\mathcal{P}^1 = \{P_1, \ldots, P_{M'}\}$.

References

[AAM+11] Noga Alon, Sanjeev Arora, Rajsekar Manokaran, Dana Moshkovitz, and Omri Weinstein. Inapproximabilty of densest k-subgraph from average case hardness, 2011.

[ACG+10] 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.

[AGLR94] B. Awerbuch, R. Gawlick, T. Leighton, and Y. Rabani. On-line admission control and circuit routing for high performance computing and communication. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 412–423, Nov 1994.

[And10] Matthew Andrews. Approximation algorithms for the edge-disjoint paths problem via Raecke decompositions. In *Proceedings of IEEE FOCS*, pages 277–286, 2010.

[AR95] Yonatan Aumann and Yuval Rabani. Improved bounds for all optical routing. In *Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms*, SODA '95, pages 567–576, Philadelphia, PA, USA, 1995. Society for Industrial and Applied Mathematics.

[AST94] Noga Alon, Paul Seymour, and Robin Thomas. Planar separators. *SIAM Journal on Discrete Mathematics*, 7(2):184–193, 1994.

[AZ05] Matthew Andrews and Lisa Zhang. Hardness of the undirected edge-disjoint paths problem. In *STOC*, pages 276–283. ACM, 2005.

[BCC+10]   Aditya Bhaskara, Moses Charikar, Eden Chlamtac, Uriel Feige, and Aravindan Vija-yaraghavan. Detecting high log-densities: an $O(n^{1/4})$ approximation for densest k-subgraph. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 201–210, 2010.

[BFSU98]   Andrei Z. Broder, Alan M. Frieze, Stephen Suen, and Eli Upfal. Optimal construction of edge-disjoint paths in random graphs. *SIAM Journal on Computing*, 28(2):541–573, 1998.

[BFU92]   Andrei Z. Broder, Alan M. Frieze, and Eli Upfal. Existence and construction of edge disjoint paths on expander graphs. In *Proceedings of the Twenty-fourth Annual ACM Symposium on Theory of Computing*, STOC '92, pages 140–149, New York, NY, USA, 1992. ACM.

[CC]   Chandra Chekuri and Julia Chuzhoy. Half-integral all-or-nothing flow. Unpublished Manuscript.

[CE13]   Chandra Chekuri and Alina Ene. Poly-logarithmic approximation for maximum node disjoint paths with constant congestion. In *Proc. of ACM-SIAM SODA*, 2013.

[Chu16]   Julia Chuzhoy. Routing in undirected graphs with constant congestion. *SIAM J. Comput.*, 45(4):1490–1532, 2016.

[CK15]   Julia Chuzhoy and David H. K. Kim. On approximating node-disjoint paths in grids. In Naveen Garg, Klaus Jansen, Anup Rao, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, AP-PROX/RANDOM 2015, August 24-26, 2015, Princeton, NJ, USA*, volume 40 of *LIPIcs*, pages 187–211. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

[CKL16]   Julia Chuzhoy, David H. K. Kim, and Shi Li. Improved approximation for node-disjoint paths in planar graphs. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2016, pages 556–569, New York, NY, USA, 2016. ACM.

[CKN17a]   Julia Chuzhoy, David H. K. Kim, and Rachit Nimavat. Improved approximation algorithm for node-disjoint paths in grid graphs with sources on grid boundary. Unpublished Manuscript, 2017.

[CKN17b]   Julia Chuzhoy, David H. K. Kim, and Rachit Nimavat. New hardness results for routing on disjoint paths. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 86–99, 2017.

[CKS05]   Chandra Chekuri, Sanjeev Khanna, and F. Bruce Shepherd. Multicommodity flow, well-linked terminals, and routing problems. In *Proc. of ACM STOC*, pages 183–192, 2005.

[CKS06]   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(1):137–146, 2006.

[CKS09]   Chandra Chekuri, Sanjeev Khanna, and F Bruce Shepherd. Edge-disjoint paths in planar graphs with constant congestion. *SIAM Journal on Computing*, 39(1):281–301, 2009.

[CL16]   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.

[CMS07]   Chandra Chekuri, Marcelo Mydlarz, and F. Bruce Shepherd. Multicommodity demand flow in a tree and packing integer programs. *ACM Trans. Algorithms*, 3(3), August 2007.

[EIS76]   Shimon Even, Alon Itai, and Adi Shamir. On the complexity of timetable and multicommodity flow problems. *SIAM J. Comput.*, 5(4):691–703, 1976.

[Fei02]   Uriel Feige. Relations between average case complexity and approximation complexity. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pages 534–543, New York, NY, USA, 2002. ACM.

[FHKS03]   Uriel Feige, Magnús M. Halldórsson, Guy Kortsarz, and Aravind Srinivasan. Approximating the domatic number. *SIAM J. Comput.*, 32(1):172–195, January 2003.

[FMS16]   Krzysztof Fleszar, Matthias Mnich, and Joachim Spoerhase. New Algorithms for Maximum Disjoint Paths Based on Tree-Likeness. In Piotr Sankowski and Christos Zaroliagis, editors, *24th Annual European Symposium on Algorithms (ESA 2016)*, volume 57 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 42:1–42:17, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[Fri00]   Alan M. Frieze. Edge-disjoint paths in expander graphs. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '00, pages 717–725, Philadelphia, PA, USA, 2000. Society for Industrial and Applied Mathematics.

[GVY97]   N. Garg, V.V. Vazirani, and M. Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18(1):3–20, 1997.

[Hol07]   Thomas Holenstein. Parallel repetition: Simplifications and the no-signaling case. In *Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing*, STOC '07, pages 411–419, New York, NY, USA, 2007. ACM.

[Kar75]   R. Karp. On the complexity of combinatorial problems. *Networks*, 5:45–68, 1975.

[Kho04]   Subhash Khot. Ruling out ptas for graph min-bisection, densest subgraph and bipartite clique. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '04, pages 136–145, Washington, DC, USA, 2004. IEEE Computer Society.

[KK13]   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, March 2013.

[Kle05]   Jon Kleinberg. An approximation algorithm for the disjoint paths problem in even-degree planar graphs. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '05, pages 627–636, Washington, DC, USA, 2005. IEEE Computer Society.

[KR96]   J. Kleinberg and R. Rubinfeld. Short paths in expander graphs. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, FOCS '96, pages 86–95, Washington, DC, USA, 1996. IEEE Computer Society.

[KS04]   Stavros G. Kolliopoulos and Clifford Stein. Approximating disjoint-path problems using packing integer programs. *Mathematical Programming*, 99:63–87, 2004.

[KT95]   Jon M. Kleinberg and Éva Tardos. Disjoint paths in densely embedded graphs. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 52–61, 1995.

[KT98]   Jon M. Kleinberg and Éva Tardos. Approximations for the disjoint paths problem in high-diameter planar networks. *J. Comput. Syst. Sci.*, 57(1):61–73, 1998.

[KvL84]   MR Kramer and Jan van Leeuwen. The complexity of wire-routing and finding minimum area layouts for arbitrary vlsi circuits. *Advances in computing research*, 2:129–146, 1984.

[LR99]   Tom Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM*, 46(6):787–832, November 1999.

[LT79]   Richard J Lipton and Robert Endre Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979.

[Lyn75]   James F. Lynch. The equivalence of theorem proving and the interconnection problem. *SIGDA Newsl.*, 5(3):31–36, September 1975.

[Man17]   Pasin Manurangsi. Almost-polynomial ratio eth-hardness of approximating densest k-subgraph. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 954–961, 2017.

[Räc02]   Harald Räcke. Minimizing congestion in general networks. In *Proc. of IEEE FOCS*, pages 43–52, 2002.

[Rao08]   Anup Rao. Parallel repetition in projection games and a concentration bound. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 1–10, New York, NY, USA, 2008. ACM.

[Raz98]   Ran Raz. A parallel repetition theorem. *SIAM J. Comput.*, 27(3):763–803, June 1998.

[RS90]   N. Robertson and P. D. Seymour. Outline of a disjoint paths algorithm. In *Paths, Flows and VLSI-Layout*. Springer-Verlag, 1990.

[RS95]   Neil Robertson and Paul D Seymour. Graph minors. XIII. the disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 63(1):65–110, 1995.

[RS10]   Prasad Raghavendra and David Steurer. Graph expansion and the unique games conjecture. In *Proceedings of the Forty-second ACM Symposium on Theory of Computing*, STOC '10, pages 755–764, New York, NY, USA, 2010. ACM.

[RT87]   Prabhakar Raghavan and Clark D. Thompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7:365–374, December 1987.

[RZ10]   Satish Rao and Shuheng Zhou. Edge disjoint paths in moderately connected graphs. *SIAM J. Comput.*, 39(5):1856–1887, 2010.

[SCS11]   Loïc Seguin-Charbonneau and F. Bruce Shepherd. Maximum edge-disjoint paths in planar graphs with congestion 2. In *Proceedings of the 2011 IEEE 52Nd Annual Symposium on Foundations of Computer Science*, FOCS '11, pages 200–209, Washington, DC, USA, 2011. IEEE Computer Society.

[Ung51]   Peter Ungar. A theorem on planar graphs. *Journal of the London Mathematical Society*, 1(4):256–262, 1951.