Almost Polynomial Hardness of Node-Disjoint Paths in Grids*

Julia Chuzhoy†
Toyota Technological Institute at Chicago
6045 S. Kenwood Ave
Chicago, IL 60637, U.S.A
cjulia@ttic.edu

David H. K. Kim‡
Computer Science Department
University of Chicago
1100 E. 58th Street
Chicago, IL 60637, U.S.A
hongk@cs.uchicago.edu

Rachit Nimavat§
Toyota Technological Institute at Chicago
6045 S. Kenwood Ave
Chicago, IL 60637, U.S.A
nimavat@ttic.edu

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 NP $\subseteq$ ZPTIME($n^{\text{poly}\log n}$). In a recent work, the authors have shown an improved $2^{\Omega(\sqrt{\log n})}$-hardness of approximation for NDP, unless NP $\subseteq$ DTIME($n^{O(\log n)}$), even if the underlying graph is a **sub-graph** 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 of 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 NP $\not\subseteq$ RTIME($n^{\text{poly}\log n}$), and 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}). 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.

CCS CONCEPTS

•**Theory of computation → Network flows; Routing and network design problems;** *Problems, reductions and completeness;*

ACM Reference format:

Julia Chuzhoy, David H. K. Kim, and Rachit Nimavat. 2018. Almost Polynomial Hardness of Node-Disjoint Paths in Grids. In *Proceedings of 50th Annual ACM SIGACT Symposium on the Theory of Computing, Los Angeles, CA, USA, June 25–29, 2018 (STOC'18),* 14 pages.

DOI: 10.1145/3188745.3188772

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, so that the routing paths are mutually disjoint in their vertices and 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 definition).

*A full version of this paper is available at https://arxiv.org/abs/1711.01980

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

STOC'18, Los Angeles, CA, USA

DOI: 10.1145/3188745.3188772

NDP is a fundamental problem in the area of graph routing, that has been studied extensively. Robertson and Seymour [47, 48] showed, as part of their famous Graph Minors Series, an efficient algorithm for solving the problem if the number k of the demand pairs is bounded by a constant. However, when k is a part of input, the problem becomes NP-hard [22, 29], even in planar graphs and in grids [37, 39]. The best current upper bound on the approximability of NDP is $O(\sqrt{n})$, obtained by a simple greedy algorithm [36]. 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($n^{\text{poly}\log n}$) [3, 4], and APX-hardness for the special cases of NDP-Planar and NDP-Grid [17]. In a recent paper [20], the authors showed an improved $2^{\Omega(\sqrt{\log n})}$-hardness of approximation for NDP, assuming that NP $\notin$ 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 [36] was recently slightly improved to an $\tilde{O}(n^{9/19})$-approximation [18].

The approximability status of NDP-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 [36] was only recently improved to a $\tilde{O}(n^{1/4})$-approximation for NDP-Grid [17], while on the negative side only APX-hardness is known. In a very recent paper [19], 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 on the grid boundary [20][1], and 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 $\notin$ RTIME(2^{n^δ}). These hardness results are stronger than the best currently known hardness for the general NDP problem, and should be contrasted with the $2^{O(\sqrt{\log n}\cdot\log\log n)}$-approximation algorithm for NDP-Grid with all sources on the grid boundary [19].

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 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 vertices, and so it may not preserve approximation factors that depend on $|V(G)|$. 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 $O(\sqrt{n})$-ratio [13], and the recent $2^{\Omega(\sqrt{\log n})}$-hardness of approximation of [20], under the assumption that NP $\notin$ DTIME($n^{O(\log n)}$), extends to EDP. Interestingly, EDP appears to be relatively easy on grid graphs, where it has a constant-factor approximation [5, 34, 35]. The analogue of grids in the setting of EDP seems to be the wall graph: the approximability status of EDP on wall graphs is similar to that of NDP on grid graphs, with the best current upper bound of $\tilde{O}(n^{1/4})$, and the best lower bound of APX-hardness [17]. The results of [20] extend to a $2^{\Omega(\sqrt{\log n})}$-hardness of approximation for EDP on sub-graphs of walls, with all source vertices lying on the wall boundary, under the same complexity assumption. We denote by EDP-Wall the special case of EDP 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.

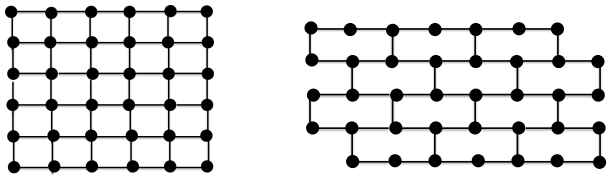

Figure 1: An illustration of a grid and a wall graph.

Other Related Work. Several special cases of EDP are known to have reasonably good approximation algorithms. These include an $O(\log n)$-approximation for both Eulerian and 4-connected planar graphs [30, 32]; polylogarithmic approximation for bounded-degree expanders [8, 9, 26, 33, 38] and for graphs in which the value of the global minimum cut is $\Omega(\log^5 n)$ [45], and a constant-factor approximation for trees [15, 27], and grids and grid-like graphs [5, 6, 34, 35]. Recently, Fleszar et al. [25] designed an $O(\sqrt{r}\cdot\log(kr))$-approximation algorithm for EDP, where r is the feedback vertex set number of the input graph.

A natural variation of NDP and EDP that slightly relaxes the disjointness constraint by allowing a small congestion has been a subject of extensive study. The NDP with Congestion (NDPwC) problem is defined exactly like NDP, except that we are also given an integer $c \geq 1$ as part of input, and the solution is allowed to incur congestion c, that is, each vertex may participate in up to 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 on the vertices. The famous result of

[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.

Raghavan and Thompson [42], 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 [2, 10–12, 16, 21, 41, 45] has led to an $O(\text{poly}\log k)$-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$(n^{\text{poly}\log n})$ [3]. When the input graph is planar, a constant-factor approximation algorithm for EDPwC with congestion 2 is known [14, 49].

Our Results and Techniques. Our main result is the proof of the following theorem.

Theorem 1.1. *For every constant $\epsilon > 0$, there are no $2^{O(\log^{1-\epsilon} n)}$-approximation algorithms for* NDP *and for* EDP*, assuming that* NP $\notin$ RTIME$(n^{\text{poly}\log n})$*. Moreover, there are no $n^{O(1/(\log\log n)^2)}$-approximation algorithms for* NDP *and for* EDP*, assuming that for some constant $\delta > 0$,* NP $\notin$ RTIME(2^{n^δ})*. These results hold even for* NDP-Grid*,* NDP-Wall*, and* EDP-Wall*.*

We now provide a high-level overview of our techniques, focusing on the hardness of NDP-Grid. The starting point of our hardness 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 need 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 and a parameter k, and the goal is to find a subset $U \subseteq V(G)$ 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 [7]. 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 [1, 23, 31, 43]. In a recent breakthrough, Manurangsi [40] 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 his techniques to this problem, or to prove its hardness of approximation via other techniques.

To overcome this difficulty, 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 achieve. 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 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, 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 for NDP of [20]. The proof in [20] performs 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- and No-Instance solutions.

Organization. We start with Preliminaries in Section 2, and introduce the new graph partitioning problem in Section 3. The hardness of approximation proof for NDP-Grid appears in Section 4, and the reduction from the partitioning problem to NDP in Section 5. Some

of the proof details, and the extension to NDP-Wall and EDP-Wall are deferred to the full version of the paper that is available on arxiv.

2 PRELIMINARIES

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 $\left\{(v_{i,j}, v_{i,j+1}) \mid 1 \leq i \leq h, 1 \leq j < \ell\right\}$ of horizontal edges, and the set $\left\{(v_{i,j}, v_{i+1,j}) \mid 1 \leq i < h, 1 \leq j \leq \ell\right\}$ of vertical edges.

Wall Graphs. Let $G = G^{\ell,h}$ be a grid of length ℓ and height h. Assume that $\ell > 0$ is an even integer, and that $h > 0$. For every column W_j of the grid, let $e^j_1, \ldots, e^j_{h-1}$ be the edges of W_j indexed in their top-to-bottom order. Let $E^*(G) \subseteq E(G)$ contain all edges e^j_z, where $z \neq j \mod 2$, 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).

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 $C = \{r, b, g\}$ of 3 colors. A *coloring* $\chi : V \to C$ is an assignment of a color in C to every vertex in V. We say that an edge $e = (u, v)$ is *satisfied* by 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 $\chi : V \to C$. We say that it is a No-Instance with respect to some $0 < \epsilon < 1$, iff for every coloring $\chi : V \to C$, at most a $(1-\epsilon)$-fraction of the edges are satisfied by χ. We use the following theorem of Feige et al. [24]:

Theorem 2.1. *[Proposition 15 in [24]] 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 for 3COL(5). The two provers are called the edge-prover and the 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 C to u and v, such that the two colors are distinct; the vertex-prover must return an assignment of a color from 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 of the provers that guarantees acceptance with probability 1: the provers fix a valid coloring $\chi : V \to C$ of G and respond to the queries according to this coloring. However, if G is a No-Instance, then for any strategy of the two provers, the verifier accepts with probability at most $(1 - \epsilon/2)$. Indeed, using standard techniques, we can assume w.l.o.g. that the strategies of the provers are deterministic. 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 such an edge is at least ϵ. The response of the edge-prover on this edge must differ from the response of the vertex-prover on at least one of its endpoints. The verifier chooses this endpoint with probability at least $1/2$, and so overall the verifier rejects with probability at least $\epsilon/2$. To summarize, if G is a Yes-Instance, then the value of the corresponding game is 1, and if it is a No-Instance, then it 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. The vertex-prover returns a coloring of the vertices in $(v_1, \ldots, v_\ell)$, that also need not be consistent across different indices (so if $v_i = v_j$, we may assign different colors to the two occurrences). The verifier accepts iff for each $1 \leq i \leq \ell$, the coloring of the vertex v_i returned by both provers is consistent. (No verification of consistency is performed across different i's. So, for example, if v_i is an endpoint of e_j for $i \neq j$, then it is possible that the two colorings do not agree and the verifier still accepts). We say that a pair (A, A') of answers to the two queries $(e_1, \ldots, e_\ell)$ 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). *[28, 44, 46] There is a constant $0 < \gamma' < 1$, such that for every 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 $x^{\gamma'\ell}$.*

Corollary 2.3. *There is a constant $0 < \gamma < 1$ independent of ℓ, 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 of the game 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, and $|\mathcal{Q}^E| = m^\ell = (5n/2)^\ell$. Each query has 6^ℓ possible answers – 6 colorings per edge. The set of feasible answers is the same for each edge-query, and is denoted by $\mathcal{A}^E$. Similarly, let $\mathcal{Q}^V$ denote the set of all possible queries to the vertex-prover, so each query is an ℓ-tuple of vertices, and $|\mathcal{Q}^V| = n^\ell$. Each query has 3^ℓ feasible answers – 3 colorings per vertex. The set of feasible answers is the same for each vertex-query, and is denoted 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.

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. We need the following simple theorem.

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

Proof. 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 $1 \le i \le 6$, permutation π_i defines a valid coloring χ_i of G: for every vertex $v \in V(G)$, if v is assigned a color $c \in \{C\}$ by χ, then χ_i assigns the color $\pi_i(c)$ to v. Notice that for each vertex v and for each color $c \in C$, 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 $c, c' \in C$ 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 $1 \le j \le \ell$, and assume that $e_j = (v_j, u_j)$. Assume that $b_j = z$, for some $1 \le z \le 6$. We assign to v_j the color $\chi_z(v_j)$, and we assign to u_j the color $\chi_z(u_j)$. Since χ_z is a valid coloring of $V(G)$, the two colors are distinct. This defines an answer $A \in \mathcal{A}^E$ to the query Q, that determines $f_b(Q)$.

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

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

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

Two Graphs. Given a 3COL(5) instance G with $|V(G)| = n$, and an integer $\ell > 0$, we associate a bipartite 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 $e(R) = (v(Q), v(Q'))$. We denote $U^E = \{v(Q) \mid Q \in \mathcal{Q}^E\}$, and $U^V = \{v(Q') \mid Q' \in \mathcal{Q}^V\}$.

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, that takes into account the answers to the queries. (It may be convenient for now to think that $H' = H$, but later we will use smaller sub-graphs of H). The vertices of $L(H')$ are partitioned into two subsets. First, 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: $\hat{U}^E = \{v(Q, A) \mid (Q \in \mathcal{Q}^E \text{ and } v(Q) \in H'), A \in \mathcal{A}^E\}$, and we denote by $\mathcal{U}_1$ its resulting partition into groups, $\mathcal{U}_1 = \{S(Q) \mid Q \in \mathcal{Q}^E : v(Q) \in H'\}$.

For the second set of vertices, 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 the resulting set of vertices by $S(Q')$: $S(Q') = \{v_i(Q', A') \mid A' \in \mathcal{A}^V, 1 \le i \le 2^\ell\}$, so $|S(Q')| = 6^\ell$. We call $S(Q')$ *the group representing Q'*. Let $\hat{U}^V$ denote the union of all such sets $S(Q')$:

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

and let $\mathcal{U}_2$ denote its resulting partition into groups:

$$\mathcal{U}_2 = \left\{S(Q') \mid Q' \in \mathcal{Q}^V : v(Q') \in H'\right\}.$$

The final set of vertices of $L(H')$ is $\hat{U}^E \cup \hat{U}^V$. We now define the set of edges of $L(H')$. 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 $E(R)$ denote the set of all such resulting edges, so $|E(R)| = 6^\ell \cdot 2^\ell = 12^\ell$. We denote by $\hat{E}$ the set of all edges of $L(H')$, so $\hat{E} = \bigcup_{R : e(R) \in H'} E(R)$.

Finally, we 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 a vertex 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 $\bigcup_{v \in \hat{U}^V} \mathcal{B}(v)$ is 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 use a new graph partitioning problem as a proxy in order to reduce the 3COL(5) problem to NDP-Grid. Since its definition is somewhat complex, we start by defining its simpler variant, and by providing some intuition and motivation for the final definition.

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

One intuitive way to think about this problem is that we need 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, as it appears to be somewhat similar to the Densest k-Subgraph problem (DkS). Like in DkS, 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 subgraph, we need to partition $V(\tilde{G})$ 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 for (r, h)-GP. In particular, known hardness proofs for DkS do not seem to generalize to this problem. To overcome this difficulty, we use a slightly more general problem in our reduction. Before defining this problem, we provide some 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 clusters $W_1, \ldots, W_r$, as follows. Fix some $1 \leq i \leq r$. 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 $1 \leq i \leq r$, 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 $1 \leq i \leq r$, 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 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 $v \in V_1 \cup V_2$, for each group $U \in \mathcal{U}_1 \cup \mathcal{U}_2$, 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. 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 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, the goal is to compute a partition $(W_1, \ldots, W_r)$ of $V_1 \cup V_2$ into r subsets, and for each $1 \leq i \leq r$, select a subset $E_i \subseteq E(W_i)$ of edges, such that $|E_i| \leq h$. But now there is an additional restriction: we require that for each $1 \leq i \leq r$, 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 $\beta^*(\mathcal{I}) = \sum_{v \in V_1} \beta(v)$.

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

Note that the value of every solution to $\mathcal{I}$ is 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. Next, we define valid instances; they are defined so that the instances that we obtain when reducing from 3COL(5) are always valid, as we show later.

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

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

Definition 3.2. We say that a solution $((W_1, \ldots, W_r), (E_1, \ldots, E_r))$ to a valid (r,h)-GPwB instance $\mathcal{I}$ is *perfect* iff: (i) for each group $U \in \mathcal{U}_1 \cup \mathcal{U}_2$, exactly one vertex of U belongs to each cluster W_i; and (ii) for each $1 \leq i \leq r$, $|E_i| = h$.

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

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. The underlying graph is $L(H') = (\hat{U}^E, \hat{U}^V, \hat{E})$. We set the parameters $r = 6^\ell$ 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 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 $v(Q) \in H'$, $|S(Q)| = 6^\ell$). We use the following simple claim.

Claim 3.3. *Let G be an instance of 3COL(5), $\ell > 0$ 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 $\mathcal{I}(H')$ has a perfect solution.*

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

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:

$$\mathcal{U}_1 = \left\{S(Q) \mid Q \in \mathcal{Q}^E, v(Q) \in H'\right\}, \text{ and}$$
$$\mathcal{U}_2 = \left\{S(Q') \mid Q' \in \mathcal{Q}^V, v(Q') \in H'\right\},$$

where each group in $\mathcal{U}_1 \cup \mathcal{U}_2$ has cardinality $r = 6^\ell$. We now fix some $1 \leq i \leq r$, and define the set W_i of vertices. For each query $Q \in \mathcal{Q}^E$ to the edge-prover with $v(Q) \in V(H')$, if $A = f_i(Q)$, then we add the vertex $v(Q, A)$ to W_i. For each query $Q' \in \mathcal{Q}^V$ to the vertex-prover with $v(Q') \in V(H')$, if $A' = f_i(Q')$, then we select some index $1 \leq j \leq 2^\ell$, 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, $|E_i| = |E(H')| = h$. □

From (r,h)-GPwB *to* NDP. The following theorem is central to our hardness of approximation proof. Its proof appears in Section 5.

Theorem 3.4. *There is a constant $c^* > 1$, 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 $\beta^*(\mathcal{I})$), then with probability at least $\frac{1}{2}$ over the construction of $\hat{\mathcal{I}}$, instance $\hat{\mathcal{I}}$ of* NDP-Grid *has a solution of value at least $\beta^*(\mathcal{I})/(c^* \log^3 M)$; and*
- *There is an efficient deterministic 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 $|\mathcal{P}^*|/(c^* \cdot \log^3 M)$.*

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 $|E(L(H))| = n^{O(\ell)}$. We let $\hat{c}$ be a large enough constant so that $|E(L(H))| \leq n^{\hat{c}\ell}$, and let $c_{\text{YI}} = (\hat{c} \cdot c^*)^3$. We obtain the following corollary of Theorem 3.4:

Corollary 3.5. *Suppose we are given a 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 $\frac{1}{2}$, the resulting instance $\hat{\mathcal{I}}$ of NDP-Grid *has a solution of value at least $|E(H')| \cdot 6^{\ell}/(c_{\text{YI}}\ell^3 \log^3 n)$, where $n = |V(G)|$. (The probability is over the random construction of $\hat{\mathcal{I}}(H')$).*

Proof. From Claim 3.3, 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 3.4, 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 $\log M \leq \hat{c}\ell \log n$, the corollary follows. □

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 $L(H)$. 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 3.5, with constant probability there is a solution to $\hat{\mathcal{I}}(H)$ of value at least $\frac{|\mathcal{R}| \cdot 6^{\ell}}{c_{\text{YI}}\ell^3 \log^3 n}$. 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 $\frac{|\mathcal{R}| \cdot 6^{\ell}}{c_{\text{YI}}^2 \cdot \alpha^* \ell^6 \log^6 n}$, we would be done. Indeed, in such a case, from Theorem 3.4, every solution to the NDP-Grid instance $\hat{\mathcal{I}}(H)$ routes fewer than $\frac{|\mathcal{R}| \cdot 6^{\ell}}{c_{\text{YI}}\alpha^* \ell^3 \log^3 n}$ 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 return a solution to $\hat{\mathcal{I}}(H)$ of value at least $\frac{|\mathcal{R}| \cdot 6^{\ell}}{c_{\text{YI}}\alpha^* \ell^3 \log^3 n}$, 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- 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 indeed has 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 have a high value, but it may not translate into a strategy for the two provers with high acceptance probability. In an extreme case, for each query Q, we may place all vertices of $S(Q)$ into a single cluster W_i. Our main idea in overcoming this difficulty is to note that such a cheating solution can be used to compute a partition of the constraint graph H into 6^{ℓ} subgraphs, each of which is significantly smaller than the original graph H, such that a large fraction of the edges of H 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 we use to execute this plan is the following theorem.

Theorem 4.1. *Suppose we are given an instance G of 3COL(5) with $|V(G)| = n$, an integral parameter $\ell > \log^2 n$, 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 $\alpha = c_{\text{YI}}^2 \cdot \alpha^* \cdot \ell^6 \log^6 n$. Then there is a randomized algorithm, that, in time $O\left(n^{O(\ell)} \cdot \log P\right)$, 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 fixed 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- 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 with running time $n^{O(\ell)}$, that uses $\mathcal{A}$ as a subroutine, in order to determine whether G is a Yes- or a No-Instance.

Throughout the algorithm, we maintain a collection $\mathcal{H}$ of disjoint 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:

$$\left\{Q^E \in \mathcal{Q}^E \mid v(Q^E) \in V(H'')\right\}, \text{ and}$$

$$\left\{Q^V \in \mathcal{Q}^V \mid v(Q^V) \in V(H'')\right\},$$

respectively, and the constraints of the verifier are:

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

For each inactive cluster $H'' \in \mathcal{H}_2$, we will store a 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 now describe an iteration when an active cluster $H' \in \mathcal{H}_1$ is processed.

(1) Construct an instance $\mathcal{I}(H')$ of (r,h)-GPwB.
(2) Use Theorem 3.4 to independently construct $n^{4\ell}$ instances $\hat{\mathcal{I}}(H')$ of NDP-Grid.
(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 $\frac{|E(H')|6^\ell}{c_{\text{YI}}\alpha^*\ell^3\log^3 n}$ demand pairs, halt and return "G is a No-Instance".
(4) Otherwise, fix any instance $\hat{\mathcal{I}}(H')$ for which $\mathcal{A}$ returned a solution of value at least $\frac{|E(H')|6^\ell}{c_{\text{YI}}\alpha^*\ell^3\log^3 n}$. Denote $|E(L(H'))| = M$, and recall that $M \le n^{\hat{c}\ell}$. Use Theorem 3.4 to compute a solution $((W_1,\ldots,W_r),(E_1,\ldots,E_r))$ to the instance $\mathcal{I}(H')$ of (r,h)-GPwB, of value at least:

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

(5) Apply the algorithm from Theorem 4.1 to this solution, with the parameter $P = n^{c\ell}$, for a sufficiently large constant c. 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 to $\mathcal{H}_2$; store the resulting strategy of the provers. Otherwise, let $\tilde{\mathcal{H}}$ be the collection of sub-graphs of H' returned by the algorithm. If $\sum_{H''\in\tilde{\mathcal{H}}}|E(H'')| < c'|E(H')|/(\ell^2\alpha^2)$, then return "$G$ is a No-Instance". Otherwise, remove H' from $\mathcal{H}_1$ and add all graphs of $\tilde{\mathcal{H}}$ to $\mathcal{H}_1$.

If the algorithm terminates with $\mathcal{H}$ containing only inactive clusters, then we return "G is a Yes-Instance". 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 (5). We now analyze these two steps.

Consider first Step (3). From Corollary 3.5, with probability at least $1/2$, a random instance $\hat{\mathcal{I}}(H')$ of NDP-Grid has a solution of value at least $\frac{|E(H')|\cdot 6^\ell}{c_{\text{YI}}\ell^3\log^3 n}$, and our α^*-approximation algorithm to NDP-Grid must then return a solution of value at least $\frac{|E(H')|\cdot 6^\ell}{c_{\text{YI}}\alpha^*\ell^3\log^3 n}$. 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 $\frac{|E(H')|6^\ell}{c_{\text{YI}}\alpha^*\ell^3\log^3 n}$. Therefore, with high probability our algorithm will not return "G is a No-Instance" due to Step (3) in this iteration.

Consider now Step (5). The algorithm can classify G as a No-Instance in this step only if $\sum_{H''\in\tilde{\mathcal{H}}}|E(H'')| < \frac{c'|E(H')|}{\ell^2\alpha^2}$. 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 (5) 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. □

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 $\mathcal{R}$.

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:

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

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

Running Time and the Hardness Factor. It is easy to see that 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 Lemma 4.2, 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($2^{n^{\delta}}$) 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. □

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 $|W_i \cap S(Q^E)|$ or $|W_i \cap S(Q^V)|$. We now give a formal definition of a terrible cluster.

Definition 4.5. Given a random string $R = (Q^E, Q^V) \in \mathcal{R}$ 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 4.6. *For each good random string $R \in \mathcal{R}'$, at most $6^{\ell}/(4\alpha)$ edges of $E'(R)$ are terrible.*

Proof. Assume for contradiction that more than $6^{\ell}/(4\alpha)$ edges of $E'(R)$ are terrible. Denote $R = (Q^E, Q^V)$. Consider some such terrible edge $e \in E'(R)$, and assume that $e \in E_i$ for some cluster W_i, that is terrible for R. We say that e is a type-1 terrible edge 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 4.7. *Let $R \in \mathcal{R}'$ be a good random string, with $R = (Q^E, Q^V)$, and let $1 \le i \le 6^{\ell}$ be an index, such that W_i is not terrible for R. Then $|W_i \cap S(Q^V)| \ge |W_i \cap S(Q^E)|/(8\alpha)$ and $|W_i \cap S(Q^E)| \ge |W_i \cap S(Q^V)|/(8\alpha)$.*

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

The proof for the second case, where $|W_i \cap S(Q^E)| < |W_i \cap S(Q^V)|/(8\alpha)$ is identical. As before, each edge of $E(R) \cap E_i$ must be incident to a distinct vertex of $S(Q^E)$, as otherwise, a pair $e, e' \in E(R)$ of edges that are incident on the same vertex $v(Q^E, A) \in S(Q^E)$ belong the the same bundle. Therefore, $|E_i \cap E(R)| \le |W_i \cap S(Q^E)| < |W_i \cap S(Q^V)|/(8\alpha)$, 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 $|E'(R)| \ge 6^{\ell}/(4\alpha)$ 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 4.7, $|S(Q^V)| \leq 8\alpha z$, 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, 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 $C_j(R)$ iff $2^{j-1} < |E^H(R) \cap E_y| \leq 2^j$. Then there is some index j_R, so that $\sum_{y \in C_{j_R}(R)} |E^H(R) \cap E_y| \geq |E^H(R)|/\log(6^\ell)$. We say that R *chooses* the index j_R. Notice that:

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

Moreover,

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

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 $i \in C_{j^*}(R)$, 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. The proof of the following claim is deferred to the full version of the paper.

Claim 4.8. *For each* $1 \leq i \leq 6^\ell$, $|E(H_i)| \leq |E(H')|/2^{\gamma\ell/16}$.

Claim 4.9. $\mathbf{E}\left[\sum_{i=1}^r |E(H_i)|\right] \geq \frac{|E(H')|}{128\ell^2\alpha^2 \log^2 6}$.

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 $y \in C_{j^*}(R)$ must belong to $J \cap J'$, and, since $j^* = j_R$, from Equation (1), $|C_{j^*}(R)| \geq \frac{6^\ell}{8 \cdot 2^{j^*} \cdot \ell\alpha \log 6}$.

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

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

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

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

□

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 3.4, 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.

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 $\lambda = 2048 \cdot \lceil M^2 \cdot \log M \rceil$. The graph $\hat{G}$ is simply the $(\lambda \times \lambda)$-grid, so $V(\hat{G}) = O(M^4 \log^2 M)$ as required. We now turn to define the set $\mathcal{M}$ of the demand pairs. We first define the set $\mathcal{M}$ itself, without specifying the locations of the corresponding vertices in $\hat{G}$, and later specify a mapping of all vertices participating in the demand pairs to $V(\hat{G})$.

Consider the underlying graph $\tilde{G} = (V_1, V_2, E)$ of the (r,h)-GPwB problem instance. Initially, for every edge $e = (u, v) \in E$, with $u \in V_1, v \in V_2$, we define a demand pair $(s(e), t(e))$ representing e, and add it to $\mathcal{M}$, so that the vertices participating in the demand pairs are all distinct. Next, we process the vertices $v \in V_1 \cup V_2$ one-by-one. Consider first some vertex $v \in V_1$, and some bundle $B \in \mathcal{B}(v)$. Assume that $B = \{e_1, \ldots, e_z\}$. 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 $\lambda/4$ 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 λ of the grid is large enough to allow this, as $h \leq M$ must hold. For all $1 \leq i \leq N_1$, we call K_i the *block representing the vertex* $v_i \in V_1$. We now fix some $1 \leq i \leq N_1$ and consider the block K_i representing the vertex v_i. We map the source vertices $s_{B_1(v_i)}, s_{B_2(v_i)}, \ldots, s_{B_{\beta(v_i)}(v_i)}$ to vertices of K_i, so that they appear on K_i in this order, 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. We now turn to analyze its properties.

Theorem 5.1. *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 $\Omega(|\mathcal{P}^*|/\log^3 M)$.*

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

Our algorithm computes a solution to the (r,h)-GPwB instance $\mathcal{I}$ by repeatedly partitioning $\tilde{G}'$ into smaller and smaller 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 5.2. 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 $|E(A)|, |E(B)| \geq \rho \cdot |E(\mathbf{H})|$.

The following theorem is central to the proof of Theorem 5.1. Due to lack of space, its proof is deferred to the full version of the paper.

Theorem 5.3. *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 $\frac{|E(\mathbf{H})|}{64 \log M}$.*

The proof of the theorem exploits the connection between routing in grids and graph drawings, and the fact that graphs with low crossing number have small balanced cuts. We now complete the proof of Theorem 5.1. 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 5.3 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. Due to lack of space, the proof is deferred to the full version of the paper.

Claim 5.4. *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 $|\mathcal{G}'| \leq r$.*

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 5.4, $z \leq r$. 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. From Claim 5.4, the value of the solution is:

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

□

The following theorem concludes the proof of Theorem 3.4. Its proof is omitted due to lack of space.

Theorem 5.5. *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 Node-Disjoint paths.*

REFERENCES

[1] Noga Alon, Sanjeev Arora, Rajsekar Manokaran, Dana Moshkovitz, and Omri Weinstein. 2011. Inapproximabilty of Densest k-Subgraph from Average Case Hardness.

[2] Matthew Andrews. 2010. Approximation Algorithms for the Edge-Disjoint Paths Problem via Raecke Decompositions. In *Proceedings of IEEE FOCS*. 277–286. https://doi.org/10.1109/FOCS.2010.33

[3] Matthew Andrews, Julia Chuzhoy, Venkatesan Guruswami, Sanjeev Khanna, Kunal Talwar, and Lisa Zhang. 2010. Inapproximability of Edge-Disjoint Paths and low congestion routing on undirected graphs. *Combinatorica* 30, 5 (2010), 485–520.

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

[5] Yonatan Aumann and Yuval Rabani. 1995. Improved bounds for all optical routing. In *Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms*

(SODA '95). Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 567–576. http://dl.acm.org/citation.cfm?id=313651.313820

[6] B. Awerbuch, R. Gawlick, T. Leighton, and Y. Rabani. 1994. On-line admission control and circuit routing for high performance computing and communication. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*. 412–423. https://doi.org/10.1109/SFCS.1994.365675

[7] Aditya Bhaskara, Moses Charikar, Eden Chlamtac, Uriel Feige, and Aravindan Vijayaraghavan. 2010. 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*. 201–210. https://doi.org/10.1145/1806689.1806718

[8] Andrei Z. Broder, Alan M. Frieze, Stephen Suen, and Eli Upfal. 1998. Optimal Construction of Edge-Disjoint Paths in Random Graphs. *SIAM J. Comput.* 28, 2 (1998), 541–573. https://doi.org/10.1137/S0097539795290805 arXiv:https://doi.org/10.1137/S0097539795290805

[9] Andrei Z. Broder, Alan M. Frieze, and Eli Upfal. 1992. 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)*. ACM, New York, NY, USA, 140–149. https://doi.org/10.1145/129712.129727

[10] Chandra Chekuri and Julia Chuzhoy. [n. d.]. Half-Integral All-or-Nothing Flow. Unpublished Manuscript.

[11] Chandra Chekuri and Alina Ene. 2013. Poly-logarithmic Approximation for Maximum Node Disjoint Paths with Constant Congestion. In *Proc. of ACM-SIAM SODA*.

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

[13] Chandra Chekuri, Sanjeev Khanna, and F. Bruce Shepherd. 2006. An $O(\sqrt{n})$ Approximation and Integrality Gap for Disjoint Paths and Unsplittable Flow. *Theory of Computing* 2, 1 (2006), 137–146.

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

[15] Chandra Chekuri, Marcelo Mydlarz, and F. Bruce Shepherd. 2007. Multicommodity Demand Flow in a Tree and Packing Integer Programs. *ACM Trans. Algorithms* 3, 3, Article 27 (Aug. 2007). https://doi.org/10.1145/1273340.1273343

[16] Julia Chuzhoy. 2016. Routing in Undirected Graphs with Constant Congestion. *SIAM J. Comput.* 45, 4 (2016), 1490–1532. https://doi.org/10.1137/130910464

[17] Julia Chuzhoy and David H. K. Kim. 2015. On Approximating Node-Disjoint Paths in Grids. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2015, August 24-26, 2015, Princeton, NJ, USA (LIPIcs)*, Naveen Garg, Klaus Jansen, Anup Rao, and José D. P. Rolim (Eds.), Vol. 40. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 187–211. https://doi.org/10.4230/LIPIcs.APPROX-RANDOM.2015.187

[18] Julia Chuzhoy, David H. K. Kim, and Shi Li. 2016. Improved Approximation for Node-disjoint Paths in Planar Graphs. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2016)*. ACM, New York, NY, USA, 556–569. https://doi.org/10.1145/2897518.2897538

[19] Julia Chuzhoy, David H. K. Kim, and Rachit Nimavat. 2017. Improved Approximation Algorithm for Node-Disjoint Paths in Grid Graphs with Sources on Grid Boundary. (2017). Unpublished Manuscript.

[20] Julia Chuzhoy, David H. K. Kim, and Rachit Nimavat. 2017. 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*. 86–99. https://doi.org/10.1145/3055399.3055411

[21] Julia Chuzhoy and Shi Li. 2016. A Polylogarithmic Approximation Algorithm for Edge-Disjoint Paths with Congestion 2. *J. ACM* 63, 5 (2016), 45:1–45:51. http://dl.acm.org/citation.cfm?id=2893472

[22] Shimon Even, Alon Itai, and Adi Shamir. 1976. On the Complexity of Timetable and Multicommodity Flow Problems. *SIAM J. Comput.* 5, 4 (1976), 691–703. https://doi.org/10.1137/0205048

[23] Uriel Feige. 2002. Relations Between Average Case Complexity and Approximation Complexity. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing (STOC '02)*. ACM, New York, NY, USA, 534–543. https://doi.org/10.1145/509907.509985

[24] Uriel Feige, Magnús M. Halldórsson, Guy Kortsarz, and Aravind Srinivasan. 2003. Approximating the Domatic Number. *SIAM J. Comput.* 32, 1 (Jan. 2003), 172–195. https://doi.org/10.1137/S0097539700380754

[25] Krzysztof Fleszar, Matthias Mnich, and Joachim Spoerhase. 2016. New Algorithms for Maximum Disjoint Paths Based on Tree-Likeness. In *24th Annual European Symposium on Algorithms (ESA 2016) (Leibniz International Proceedings in Informatics (LIPIcs))*, Piotr Sankowski and Christos Zaroliagis (Eds.), Vol. 57. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 42:1–42:17. https://doi.org/10.4230/LIPIcs.ESA.2016.42

[26] Alan M. Frieze. 2000. Edge-disjoint Paths in Expander Graphs. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '00)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 717–725. http://dl.acm.org/citation.cfm?id=338219.338631

[27] N. Garg, V.V. Vazirani, and M. Yannakakis. 1997. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica* 18, 1 (1997), 3–20. https://doi.org/10.1007/BF02523685

[28] Thomas Holenstein. 2007. Parallel Repetition: Simplifications and the No-signaling Case. In *Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing (STOC '07)*. ACM, New York, NY, USA, 411–419. https://doi.org/10.1145/1250790.1250852

[29] R. Karp. 1975. On the Complexity of Combinatorial Problems. *Networks* 5 (1975), 45–68. Issue 1.

[30] Ken-Ichi Kawarabayashi and Yusuke Kobayashi. 2013. An O(log n)-Approximation Algorithm for the Edge-Disjoint Paths Problem in Eulerian Planar Graphs. *ACM Trans. Algorithms* 9, 2, Article 16 (March 2013), 13 pages. https://doi.org/10.1145/2438645.2438648

[31] Subhash Khot. 2004. 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)*. IEEE Computer Society, Washington, DC, USA, 136–145. https://doi.org/10.1109/FOCS.2004.59

[32] Jon Kleinberg. 2005. 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)*. IEEE Computer Society, Washington, DC, USA, 627–636. https://doi.org/10.1109/SFCS.2005.18

[33] J. Kleinberg and R. Rubinfeld. 1996. Short Paths in Expander Graphs. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science (FOCS '96)*. IEEE Computer Society, Washington, DC, USA, 86–95. http://dl.acm.org/citation.cfm?id=874062.875507

[34] Jon M. Kleinberg and Éva Tardos. 1995. Disjoint Paths in Densely Embedded Graphs. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*. 52–61.

[35] Jon M. Kleinberg and Éva Tardos. 1998. Approximations for the Disjoint Paths Problem in High-Diameter Planar Networks. *J. Comput. Syst. Sci.* 57, 1 (1998), 61–73.

[36] Stavros G. Kolliopoulos and Clifford Stein. 2004. Approximating disjoint-path problems using packing integer programs. *Mathematical Programming* 99 (2004), 63–87. https://doi.org/10.1007/s10107-002-0370-6

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

[38] Tom Leighton and Satish Rao. 1999. Multicommodity Max-flow Min-cut Theorems and Their Use in Designing Approximation Algorithms. *J. ACM* 46, 6 (Nov. 1999), 787–832. https://doi.org/10.1145/331524.331526

[39] James F. Lynch. 1975. The Equivalence of Theorem Proving and the Interconnection Problem. *SIGDA Newsl.* 5, 3 (Sept. 1975), 31–36. https://doi.org/10.1145/1061425.1061430

[40] Pasin Manurangsi. 2017. 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*. 954–961. https://doi.org/10.1145/3055399.3055412

[41] Harald Räcke. 2002. Minimizing Congestion in General Networks. In *Proc. of IEEE FOCS*. 43–52.

[42] Prabhakar Raghavan and Clark D. Thompson. 1987. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica* 7 (December 1987), 365–374. Issue 4. https://doi.org/10.1007/BF02579324

[43] Prasad Raghavendra and David Steurer. 2010. Graph Expansion and the Unique Games Conjecture. In *Proceedings of the Forty-second ACM Symposium on Theory of Computing (STOC '10)*. ACM, New York, NY, USA, 755–764. https://doi.org/10.1145/1806689.1806792

[44] Anup Rao. 2008. Parallel Repetition in Projection Games and a Concentration Bound. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing (STOC '08)*. ACM, New York, NY, USA, 1–10. https://doi.org/10.1145/1374376.1374378

[45] Satish Rao and Shuheng Zhou. 2010. Edge Disjoint Paths in Moderately Connected Graphs. *SIAM J. Comput.* 39, 5 (2010), 1856–1887.

[46] Ran Raz. 1998. A Parallel Repetition Theorem. *SIAM J. Comput.* 27, 3 (June 1998), 763–803. https://doi.org/10.1137/S0097539795280895

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

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

[49] Loïc Seguin-Charbonneau and F. Bruce Shepherd. 2011. 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)*. IEEE Computer Society, Washington, DC, USA, 200–209. https://doi.org/10.1109/FOCS.2011.30