Hardness of the Undirected Edge-Disjoint Paths Problem with Congestion

Matthew Andrews* Julia Chuzhoy† Sanjeev Khanna‡ Lisa Zhang*

Abstract

In the Edge-Disjoint Paths problem with Congestion (EDPwC), we are given a graph with n nodes, a set of terminal pairs and an integer c. The objective is to route as many terminal pairs as possible, subject to the constraint that at most c demands can be routed through any edge in the graph. When $c = 1$, the problem is simply referred to as the Edge-Disjoint Paths (EDP) problem. In this paper, we study the hardness of EDPwC in undirected graphs.

We obtain an improved hardness result for EDP, and also show the first polylogarithmic integrality gaps and hardness of approximation results for EDPwC. Specifically, we prove that EDP is $(\log^{\frac{1}{2}-\varepsilon} n)$-hard to approximate for any constant $\varepsilon > 0$, unless $NP \subseteq ZPTIME(n^{\textit{polylog } n})$. We also show that for any congestion $c = o(\log \log n / \log \log \log n)$, there is no $(\log^{\frac{1-\varepsilon}{c+1}} n)$-approximation algorithm for EDPwC, unless $NP \subseteq ZPTIME(n^{\textit{polylog } n})$. For larger congestion, where $c \leq \eta \log \log n / \log \log \log n$ for some constant η, we obtain superconstant inapproximability ratios. All of our hardness results can be converted into integrality gaps for the multicommodity flow relaxation. We also present a separate elementary direct proof of this integrality gap result.

Finally, we note that similar results can be obtained for the All-or-Nothing Flow (ANF) problem, a relaxation of EDP, in which the flow unit routed between the source-sink pairs does not have follow a single path, so the resulting flow is not necessarily integral. Using standard transformations, our results also extend to the node-disjoint versions of these problems as well as to the directed setting.

1 Introduction

In the *edge-disjoint paths* (EDP) problem we are given a graph $G = (V, E)$ and a set $\{(s_1, t_1), (s_2, t_2), \ldots, (s_k, t_k)\}$ of pairs of vertices called terminals. The objective is to connect as many pairs as possible via edge-disjoint paths. Even highly restricted cases of EDP correspond to well-studied important optimization problems. For instance, EDP on trees of height one is equivalent to the graph matching problem. EDP and its variants also have a host of applications to network routing, resource allocation, and VLSI design. It is then not surprising that EDP is one of the most well-studied problems in combinatorial optimization. In directed graphs, the problem becomes NP-hard even when we are given only two source-sink pairs [16]. In undirected graphs, the seminal work of Robertson and Seymour [28] gives a polynomial time algorithm for any constant number of pairs. These results are suggestive of the inherent differences between the undirected and directed versions of EDP. However, the tractability of undirected EDP with constant number of pairs does not hold once the number of pairs is allowed to grow as a function of the input size. In particular, the problem is NP-hard even on planar graphs [17].

Consequently, much of the recent work on EDP has focused on understanding the polynomial-time approximability of the problem. While constant or poly-logarithmic approximation algorithms are known for restricted classes of graphs such as trees, meshes, and expanders [4, 12, 15, 18, 22, 23], the approximability of EDP in general graphs is not well understood. The best approximation algorithm for EDP in directed graphs has a ratio of $\tilde{O}(\min(n^{2/3}, \sqrt{m}))$ [11, 24, 25, 30, 31] where n and m denote the number of vertices and edges respectively in the input graph. For undirected graphs and directed acyclic graphs, this factor improves to an $O(\sqrt{n})$-approximation ratio [10]. In directed graphs, the approximation ratio is matched by an $\Omega(m^{\frac{1}{2}-\varepsilon})$-hardness due to Guruswami *et al.* [19]. In contrast, only APX-hardness was known for undirected EDP until the recent work in [1] which showed an $\Omega(\log^{\frac{1}{3}-\varepsilon} n)$ hardness, unless $NP \subseteq ZPTIME(n^{\text{polylog } n})$.

In this paper we study EDP together with a natural generalization, *edge-disjoint paths with congestion* (EDPwC), in which the goal is to route as many terminal pairs as possible subject to the constraint that at most c paths are routed through any edge. For constant congestion $c \geq 2$, there exists an $O(n^{1/c})$ approximation [5, 6, 25]. When

*Bell Laboratories, Lucent Technologies, Murray Hill, NJ. {andrews,ylz}@research.bell-labs.com

†CSAIL, MIT and Dept. of CIS, University of Pennsylvania, cjulia@csail.mit.edu

‡Dept. of CIS, University of Pennsylvania, Philadelphia PA. sanjeev@cis.upenn.edu. Supported in part by an NSF Career Award CCR-0093117.

the congestion is allowed to be $O(\log n/\log\log n)$ we get a constant approximation via randomized rounding [26]. For planar graphs, when congestion 2 is allowed, an $O(\log n)$-approximation has recently been derived [8, 9]. We note that the performance of an approximation algorithm for EDPwC is measured with respect to an optimal solution with no congestion.

Another related problem is the *all-or-nothing* (ANF) flow problem where for each routed pair, it suffices to provide a unit of (not necessarily integral) flow. Thus ANF is a relaxation of EDP. Recent work has shown that in undirected graphs, ANF is $O(\log^2 n)$-approximable [7, 9]. The $\Omega(\log^{\frac{1}{3}-\varepsilon} n)$ hardness result in [1] extends to ANF as well. We also study the variant of ANF where congestion is allowed, referred to as *ANF with Congestion* (ANFwC).

Our results: This paper represents a merging of the two papers [3, 13]. The main result of these papers is the following theorem.

Theorem 1 *For any constant $\varepsilon > 0$ and any congestion $c = o(\log\log n/\log\log\log n)$, there is a constant γ_c such that there is no $(\log^{\frac{1-\varepsilon}{\gamma_c}} n)$-approximation algorithm for undirected EDPwC and ANFwC, unless $NP \subseteq ZPTIME(n^{polylog\ n})$. In addition, there is a matching integrality gap of the multicommodity flow relaxation.*

The constructions used in [3, 13] to establish the above theorem are quite different. We present in this paper both proofs as our constructions may be of independent interest. For EDP with no congestion, both [3, 13] give $\gamma_1 = 2$ in Theorem 1. For larger values of $c = o(\log\log n/\log\log\log n)$, the construction in [3] yields $\gamma_c = c + 1$ while $\gamma_c = \frac{3}{2}c + \frac{1}{2}$ in [13]. When $c \leq \eta \log\log n/\log\log\log n$ for some constant η, our constructions imply a superconstant inapproximability ratio.

We note that an immediate consequence of Theorem 1 is that for any integer i, the gap between $1/i$-integral multicommodity flow (i.e. each flow path carries a multiple of $1/i$ units of flow) and fractional multicommodity flow is super-constant in undirected graphs. To our knowledge, prior to our work, it was not known if there was a super-constant gap even between half-integral flow and fractional flow in directed or undirected graphs. We present a simple family of instances that directly establish these integrality gap results.

We note that similar results have been obtained independently in [20].

Overview of Techniques: We start by giving an overview of the hardness result for the simplest setting, namely, EDP with no congestion. We show here that EDP is $\Omega(\log^{\frac{1}{2}-\varepsilon} n)$-hard, building on the framework of [1]. The construction in [13], hereafter referred to as the *CK construction*, establishes this result by directly working with the PCP characterization of NP due to [29], thus avoiding an intermediate step taken by [1] of creating an independent set instance. The high-level idea of the reduction is as follows. Given an instance ϕ of 3SAT, we construct a graph G_ϕ which contains a sufficiently large collection of edge-disjoint paths for each accepting configuration u of the verifier on ϕ. These paths are referred to as the *canonical paths* of u. The canonical path collections for any two accepting configurations u and v that disagree on some proof bit are made to "randomly intersect" with each other to encode this conflict. The random intersections ensure that the resulting graph has "high girth". The construction in [3], hereafter referred to as the *AZ construction*, is similar except that instead of using a PCP it uses the Raz two-prover interactive proof system [27] as its starting point, building on the ideas of [2].

The graph G_ϕ serves as the input graph for an EDP instance and the source-sink pairs are simply the end-points of the canonical path collections. The pairs that are routed along canonical paths conflict with high probability whenever the underlying configurations are in conflict with each other. However, these conflicts can be avoided if pairs choose paths that are not canonical. The high girth property ensures that on average, a non-canonical path is much longer than a canonical path and thus consumes much more of the routing capacity of the graph. As a result, whenever ϕ is not satisfiable, with high probability, a much smaller fraction of pairs can be routed in the graph G_ϕ. This gap enables us to establish our hardness result.

One property of the above graph G_ϕ in the PCP-based CK construction is that at most two canonical paths pass through any edge. These two paths correspond to accepting interactions that differ on some bit of the PCP proof. In order to prove hardness of EDPwC using similar ideas we need a larger number of canonical paths to pass through an edge. The CK construction achieves this by recursively building a sequence of instances $G_1, G_2, \ldots$ such that G_i is a hard instance for EDP with congestion $2^i - 1$. In particular the base case is a hard instance for EDP. The AZ construction uses the fact that each query in this proof system has multiple possible answers. We can take a canonical path that corresponds to each possible answer and let these paths all pass through the same edge. We then show that any solution to EDPwC that routes a large number of terminal pairs with small congestion can be translated into a pair of provers that convince the verifier to accept with high probability. The hardness of EDPwC follows from the error probability of the proof system.

Organization: We present the CK and the AZ hardness constructions in Sections 2 and 3, respectively. While the integrality gap result follows from the hardness constructions, we present a much simpler direct proof of this result in Section 2.

2. The CK Construction

Our starting point is a PCP characterization of NP, proved by Samorodnitsky and Trevisan in [29]. We briefly summarize the construction here. Let ϕ be an instance of 3SAT on n variables. For any constant $k > 0$, the ST construction gives a PCP verifier that uses $r = O(\log n)$ random bits to generate $q = k^2$ locations to probe in the proof. The verifier reads these q bits in the given proof Π and decides whether or not ϕ is satisfiable. Given a random string r of the verifier, let $b_1(r), \ldots, b_q(r)$ be the indices of the proof bits read. A *configuration* is $(r, a_1, \ldots, a_q)$, where $a_1, \ldots, a_q \in \{0, 1\}$ are values of $\Pi_{b_1(r)}, \ldots, \Pi_{b_q(r)}$. We say that a configuration $(r, a_1, \ldots, a_q)$ is *accepting*, if, for a random string r of the verifier and the values $a_1, \ldots, a_q$ of proof bits $\Pi_{b_1(r)}, \ldots, \Pi_{b_q(r)}$, the verifier accepts. If ϕ is a YES-INSTANCE (i.e., ϕ is satisfiable), there exists a proof Π such that the probability that the verifier accepts is at least $1/2$. Otherwise, if ϕ is a NO-INSTANCE (i.e., it is non-satisfiable), for all proofs Π, the verifier accepts with probability at most 2^{-k^2}. Abusing the notation, we will denote by r both the random string of the verifier and the number of random bits (i.e., the length of the string).

For our reductions, we would assume that this protocol is independently repeated $\lambda = \frac{2\beta \log\log n}{k^2} = O(\log\log n)$ times where $\beta >> k^2$ is a large constant. The verifier now accepts iff the original verifier accepts in each protocol repetition. The resulting PCP has the following properties. Let R denote the set of all possible random string, then $|R| = 2^{\lambda r}$, where $\lambda r = O(\log n \log\log n)$. The number of query bits is $q = \lambda k^2 = O(\log\log n)$. W.l.o.g., assume that the verifier reads exactly q bits of proof for every random string. A YES-INSTANCE is accepted with probability at least $2^{-\lambda}$ while a NO-INSTANCE is accepted with probability at most $2^{-\lambda k^2}$. For each random string, there are $2^{\lambda(2k-1)}$ accepting configurations. For every random string r, for every $j : 1 \le j \le q$, the number of accepting configurations where the value of $\Pi_{b_j(r)} = 0$ equals the number of accepting configurations where $\Pi_{b_j(r)} = 1$. For each proof bit Π_j let Z_j be the set of all the accepting configurations in which bit Π_j participates with value 0, and let O_j be the set of all the accepting configurations in which Π_j participates with value 1. We denote $n_j = |Z_j| = |O_j|$. Then $n_j \ge 2^{\lambda r/2}$. Let $\mathcal{C}$ denote the set of all the accepting configurations, $|\mathcal{C}| \le 2^{\lambda r} \cdot 2^{2\lambda k}$.

2.1. Hardness of Approximating EDP

We start by establishing that EDP is hard to approximate to within a factor of $\Omega(\log^{1/2-\epsilon} N)$ for any $\epsilon > 0$. This construction will also serve as a building block for establishing hardness of EDPwC. The starting point of our reduction is a PCP verifier for 3SAT as summarized above. Let ϕ be an instance of 3SAT on n variables. Consider a PCP verifier V for ϕ as described in the preceding section. We will use V to construct an EDP instance on a graph G_ϕ such that if ϕ is satisfiable, at least P_{YI} pairs can be routed, and if ϕ is unsatisfiable, with high probability, only $P_{YI}/\log^{1/2-\epsilon} N$ pairs can be routed; here $N = n^{\text{polylog(n)}}$ denotes the size of G_ϕ. Recall that k is a large constant, and $\lambda = \frac{2\beta \log\log n}{k^2}$. The gap between the yes and the no instances in the PCP construction is close to $2^{\lambda k^2} = \log^{2\beta} n$. In our construction, we will make the gap between the yes and the no instances close to $2^{\lambda k^2}$, while the graph size N will be close to $2^{2^{2\lambda k^2}}$, thus proving $\Omega(\log^{\frac{1}{2}-\epsilon} N)$-hardness.

We construct our graph in two steps. First, we construct, for each proof bit Π_i, a gadget denoted by $G(i)$. In the second step, we create the final graph, by connecting all the gadgets representing the proof bits, and by adding source and sink pairs.

2.1.1 The Bit Gadget

We will use two parameters M and X to describe the gadget. Consider some proof bit Π_i. We now show how to construct a corresponding gadget $G(i)$. Recall that Z_i, O_i are the collection of all the accepting configurations, in which the value of Π_i is 0 or 1, respectively, with $|Z_i| = |O_i| = n_i$. For each configuration $\alpha \in Z_i \cup O_i$, for each $m : 1 \le m \le M + 1$, there are X vertices $v_{x,m}(\alpha, i)$, for $1 \le x \le X$, called *level m vertices*, representing this configuration.

Additionally, for each $m : 1 \le m \le M$, we have Xn_i edges, called *special edges at level* m, and denoted by $(\ell_{a,m}, r_{a,m})$, $1 \le a \le Xn_i$. We also denote the set of left endpoints of these edges by $L_m(i) = \{\ell_{a,m}\}_{a=1}^{Xn_i}$, and the set of right endpoints of these edges by $R_m(i) = \{r_{a,m}\}_{a=1}^{Xn_i}$.

Finally, we show how to connect the vertices representing the configurations with the special edges. This is done by the means of *regular edges*, as follows. Consider level m vertices, for $1 \le m \le M$. We have Xn_i level m vertices, representing configurations in Z_i (denote this set of vertices by $Z_m(i)$), and Xn_i level m vertices, representing configurations in O_i (these vertices are denoted by $O_m(i)$). We perform a random matching between $Z_m(i)$ and $L_m(i)$, and also we perform a random matching between $O_m(i)$ and $L_m(i)$. Additionally, for each $m : 2 \le m \le M + 1$, we perform random matchings between $Z_m(i)$ and R_{m-1}, and between $O_m(i)$ and R_{m-1}. The edges participating in these matchings are added to the gadget as regular edges (see Figure 1).

This concludes the definition of bit gadget. We now define, for each configuration $\alpha \in Z_i \cup O_i$, a collection of X edge-disjoint paths, called *canonical paths*, representing α in gadget $G(i)$. A canonical path $P_x(\alpha, i)$, for $1 \le x \le X$,

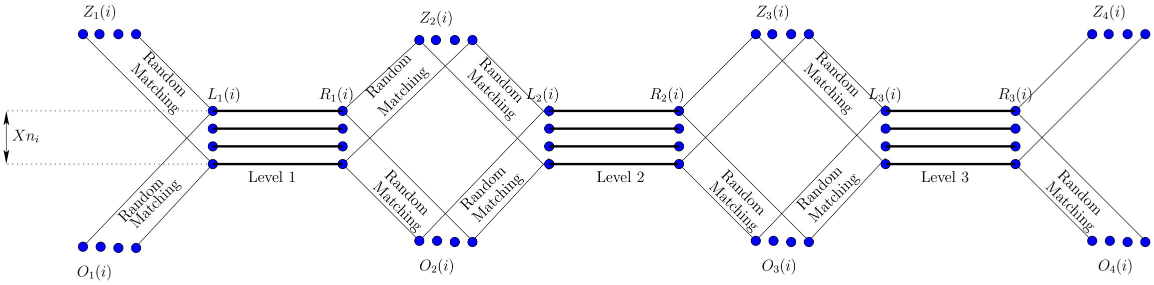

Figure 1. Gadget representing proof bit Π_i for $M = 3$

is defined as: $(v_{x,1}(\alpha, i), \ell_{a_1,1}(i), r_{a_1,1}(i), v_{x_1,2}(\alpha, i), \ldots, \ell_{a_M,M}(i), r_{a_M,M}(i), v_{x_M,M+1}(\alpha, i))$. The indices x_m, a_m for $1 \le m \le M$ are determined by the corresponding matchings. Therefore, we have X edge-disjoint paths representing α in gadget G_i. Moreover, for all the configurations in Z_i, their Xn_i canonical paths are edge disjoint. The same is true for all the configurations in O_i.

Let $1 \le m \le M$, and consider the collection of special edges at level m. Each such edge participates in exactly one canonical path representing a configuration in Z_i, and exactly one canonical path representing a configuration in O_i. Thus, the set of special level m edges defines a random matching between the paths representing the configurations in Z_i and the paths representing the configurations in O_i. In total, gadget G_i defines M random matchings (one matching for each level) between these two sets of paths, and these random matchings are completely independent. Observe that the length of each canonical path is $3M$, and the degree of every vertex is at most 3.

Bit Gadget Analysis Set $\Delta = \frac{M}{8 \log M}$, so that $M \ge 8\Delta \log \Delta$ holds. Consider the gadget representing some proof bit Π_i. Let $\mathcal{P}_0$ be the set of canonical paths representing configurations in Z_i, and let $\mathcal{P}_1$ be the set of canonical paths representing configurations in O_i. Recall that $|\mathcal{P}_0| = |\mathcal{P}_1| = Xn_i$.

We say that the gadget is *bad* if there is a pair of subsets $A \subseteq \mathcal{P}_0$, $B \subseteq \mathcal{P}_1$, where $|A| = |B| = \frac{Xn_i}{\Delta}$, such that all the paths in $A \cup B$ are edge disjoint. We say that bad event $\mathcal{B}_1$ happens, if at least one of the gadgets is bad. The proof of the lemma below is similar to a lemma in [1].

Lemma 2 *The probability that gadget $G(i)$ is bad is at most e^{-n}.*

Corollary 1 *The probability that bad event $\mathcal{B}_1$ happens is at most $\frac{1}{\mathrm{poly}(n)}$.*

2.1.2 The Final Instance

Let α be some accepting configuration, and let $i_1, i_2, \ldots, i_q$ be the indices of proof bits participating in α, with $q \le \lambda k^2$. Consider bit gadget $G(i_j)$, for some $1 \le j \le q$. There are X level 1 vertices representing α in $Z_1(i_j) \cup O_1(i_j)$, denote them by $V_j = \{v_{1,1}(\alpha, i_j), \ldots, v_{X,1}(\alpha, i_j)\}$. There are also X level $M+1$ vertices representing α in $Z_{M+1}(i_j) \cup O_{M+1}(i_j)$, denote them by $U_j = \{v_{1,M+1}(\alpha, i_j), \ldots, v_{X,M+1}(\alpha, i_j)\}$.

We add a set of X source vertices representing configuration α, $S(\alpha) = \{s_1(\alpha), \ldots, s_X(\alpha)\}$, and X destination vertices $T(\alpha) = \{t_1(\alpha), \ldots, t_X(\alpha)\}$ (we show how to divide them into pairs later).

We perform a random matching between $S(\alpha)$ and V_1, and also a random matching between $T(\alpha)$ and U_q. Additionally, for each $j : 1 \le j < q$, we perform a random matching between U_j and V_{j+1}. All the edges in the random matchings are added to the graph as regular edges.

For each configuration α, we define X canonical paths $P_x(\alpha)$, $1 \le x \le X$, representing α, as follows. Let $i_1, \ldots, i_q$ be the indices of the proof bits participating in α, $q \le \lambda k^2$. For each $x : 1 \le x \le X$, we have $P_x(\alpha) = (s_x(\alpha), P_{x_1}(\alpha, j_1), \ldots, P_{x_q}(\alpha, j_q), t_{x'})$, where $x_1, \ldots, x_q, x'$ are determined according to the corresponding matchings.

The graph has the following properties: (i) the length of a canonical path is at most $4M\lambda k^2$; (ii) for each configuration α, there are X edge-disjoint canonical paths representing α; and (iii) the degree of each vertex is at most 3.

Graph size: observe that each graph vertex and edge participate in at least one canonical path. The number of canonical paths is: $X \cdot 2^{\lambda r} \cdot 2^{\lambda(2k-1)}$, and the length of each canonical path is at most $4M\lambda k^2$. It remains to specify the values of the parameters M and X. We will use $M = 2^{\lambda(k^2+k)} = \mathrm{poly} \log n$, and $X = 2^{2^{2\lambda(k^2+4k)}} = 2^{\mathrm{poly} \log n}$. Therefore, the size of the graph is bounded by: $N \le X \cdot 2^{\lambda r} \cdot M \cdot 2^{2\lambda k} \le X \cdot 2^{O(\log n \log \log n)} \cdot 2^{O(\log \log n)} \cdot 2^{O(\log \log n)} \le X \cdot 2^{O(\log n \log \log n)}$.

The source-sink pairs are defined as follows: For

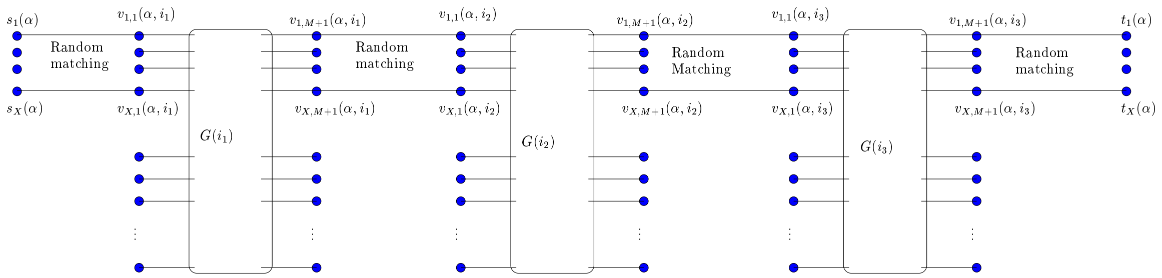

Figure 2. Source-sink pairs for a configuration α.

each accepting configuration α, the canonical paths $P_1(\alpha), \ldots, P_X(\alpha)$ define a matching between the sources and the destinations corresponding to α. We use this matching to define the source-destination pairs. Let $\mathcal{P}$ denote the set of all the canonical paths.

2.1.3 Yes Instance: ϕ is Satisfiable

In the YES-INSTANCE , there is a PCP proof, for which the acceptance probability is at least $2^{-\lambda}$. For each random string r satisfied by this proof, we can choose all the canonical paths representing the corresponding accepting configuration. All the paths thus chosen are edge disjoint, and the number of chosen paths is at least $P_{YI} \geq X2^{\lambda r-\lambda} \geq \frac{|\mathcal{P}|}{2^{2\lambda k} \cdot 2^{\lambda}}$.

2.1.4 No Instance: ϕ is Unsatisfiable

Suppose we have a no instance, and a collection $\mathcal{P}'$ of edge disjoint source-sink paths. We will show that $|\mathcal{P}'|$ can roughly be bounded by $\frac{|\mathcal{P}|}{2^{\lambda k^2}}$. We partition $\mathcal{P}'$ into three subsets, as follows. Let $g = 2^{2\lambda(k^2+k)}$. A non-canonical path is called *long* if its length is more than g. Otherwise, it is called *short*. Let $\mathcal{P}_1 \subseteq \mathcal{P}'$ be the subset of canonical paths, $\mathcal{P}_2, \mathcal{P}_3 \subseteq \mathcal{P}'$ be the subsets of long and short non-canonical paths, respectively. We bound the size of each subset separately.

Canonical Paths: Assume B_1 does not happen. Then in each bit gadget G_i, either the number of paths representing Z_i is less than $n_i X/\Delta$, or the number of paths representing O_i is less than $n_i X/\Delta$. Therefore, if we remove at most $\sum_i n_i X/\Delta$ paths from $\mathcal{P}'$, we obtain a new collection $\mathcal{P}'_1$ of canonical paths, such that in each gadget $G(i)$, we have only paths from Z_i or only paths in O_i. We can thus define a PCP proof as follows: the value of bit Π_i is 0 iff paths representing Z_i are present in $\mathcal{P}'_1$, and it is 1 otherwise. Since we are in a NO-INSTANCE , and there are X paths representing each configuration, $|\mathcal{P}'_1| \leq X \cdot 2^{\lambda r}/2^{\lambda k^2}$, which is at most $P_{YI}/2^{\lambda k^2-\lambda}$.

On the other hand, $\sum_i n_i$ can be bounded by $|\mathcal{C}|q \leq 2^{\lambda r+2\lambda k}\lambda k^2$. Also, recall that $\Delta = \frac{M}{8\log M} = \frac{2^{\lambda(k^2+k)}}{8\lambda(k^2+k)}$. Thus $\sum_i \frac{Xn_i}{\Delta} \leq \frac{X2^{\lambda r+2\lambda k}\lambda k^2}{\Delta} \leq \frac{X2^{\lambda r}}{2^{\lambda k^2-2\lambda k}}$ when k is sufficiently large. We can now bound $|\mathcal{P}_1 \setminus \mathcal{P}'_1| \leq \frac{X2^{\lambda r}}{2^{\lambda k^2-2\lambda k}} \leq P_{YI}/2^{\lambda k^2-2\lambda k-\lambda}$. Summing up, $|\mathcal{P}_1| \leq 2P_{YI}/2^{\lambda k^2-2\lambda k-\lambda}$.

Long Non-Canonical Paths: The length of a non-canonical path is at least g. The total number of edges in our graph is at most $|\mathcal{P}| \cdot 4M\lambda k^2$. Therefore, the size of $\mathcal{P}_2$ is bounded by $\frac{|\mathcal{P}|\cdot 4M\cdot\lambda k^2}{g}$. We will show that $\frac{g}{4M\cdot\lambda k^2} \geq 2^{\lambda k^2}$. Recall that $g = 2^{2\lambda(k^2+k)}$, while $M = 2^{\lambda(k^2+k)}$, and thus $4M\lambda k^2 \cdot 2^{\lambda k^2} \leq 4\lambda k^2 \cdot 2^{2\lambda k^2+k\lambda} \leq 2^{2\lambda(k^2+k)} \leq g$. So $|\mathcal{P}_2| \leq \frac{|\mathcal{P}|}{2^{\lambda k^2}} \leq \frac{P_{YI}}{2^{\lambda k^2-2\lambda k-\lambda}}$.

Short Non-Canonical Paths: Suppose there is a short non-canonical path $P \in \mathcal{P}_3$ connecting some source and destination pair (s, t). This path must form a cycle of length at most $g + 4M\lambda k^2 \leq 2g$ with the canonical $s-t$ path. Moreover, at least one edge on the cycle participates in P. Let K denote the number of cycles of length at most $2g$ in our graph. Then $|\mathcal{P}_3| \leq 2g \cdot K$. Our goal is to show that with high probability, K is small. The proof of the claim below is similar a claim in [1].

Lemma 3 *With probability at least $\frac{2}{3}$, $K \leq 2^{4\lambda rg}$.*

We say that the bad event $\mathcal{B}_2$ happens if $K > 2^{4\lambda rg}$. Assuming B_2 does not happen, we get: $|\mathcal{P}_3| \leq 2g \cdot 2^{4\lambda rg} \leq 2^{5\lambda rg} = 2^{5\lambda r \cdot 2^{2\lambda(k^2+k)}} \leq 2^{2^{2\lambda(k^2+3k)+\log\log n}}$ since $r = O(\log n)$. Recall that $\lambda = \beta \log\log n/k^2$ for very large constant $\beta >> k^2$, and thus we can assume that $\lambda k \geq \log\log n$, and $|\mathcal{P}_3| \leq 2^{2^{2\lambda(k^2+4k)}} \leq X \leq P_{YI}/2^{\lambda k^2}$.

2.1.5 Putting it Together

If the events $\mathcal{B}_1$ and $\mathcal{B}_2$ do not happen, then $|\mathcal{P}'| = |\mathcal{P}_1| + |\mathcal{P}_2| + |\mathcal{P}_3| \leq P_{YI}/2^{\lambda(k^2-3k)}$, and thus the gap is $\Omega(2^{\lambda(k^2-3k)})$. Recall that $N = X \cdot 2^{O(\log n \log\log n)} = 2^{2^{2\lambda(k^2+4k)}+O(\log n \log\log n)} \leq 2^{2^{2\lambda(k^2+5k)}}$. Therefore, the gap is $\log^{\frac{1}{2}-\epsilon} N$, where ϵ is a constant that depends on k and can be made arbitrarily small by choosing k to be sufficiently large.

Now suppose at least one of the events $\mathcal{B}_1$ or $\mathcal{B}_2$ does happen. Then $|\mathcal{P}'|$ may be much larger than the above bound even though ϕ is not satisfiable. But the probability of $B_1 \cup B_2$ is at most $1/\operatorname{poly} n + 1/3 \leq 1/2$. Thus a $\log^{\frac{1}{2}-\epsilon} N$-approximation algorithm for EDP would give us a co-RPTIME($n^{\mathrm{polylog(n)}}$) algorithm for 3SAT. Since 3SAT is in NP, we can use a standard result to convert this into a ZPTIME($n^{\mathrm{polylog(n)}}$) algorithm for 3SAT, giving us our main result.

2.2. Hardness of EDP with Congestion

We will now establish hardness of approximating EDP with congestion $c \geq 2$. We will focus here on the case when c is any constant. As earlier, we perform a reduction from 3SAT using the PCP characterization presented above. The parameters q, λ, and r stay the same when c is a constant. Towards the end, we briefly describe how the parameters change when c is allowed to be as large as $(\log\log n)/(\log\log\log n)^2$, and we also sketch how to extend these results to ANFwC.

In what follows, let z be the least integer such that $c < 2^z$. We will iteratively define sample spaces of EDP instances, namely $H_1, H_2, ..., H_z$, such that the sample space H_i is defined in terms of H_{i-1} for $2 \leq i \leq z$. We will prove that if ϕ is a YES-INSTANCE , then any instance of H_z has a collection of edge disjoint paths in H_z of size at least P_{YI}, while if ϕ is a NO-INSTANCE , then with high probability, at most P_{NI} source-sink pairs can be routed with congestion at most $2^z - 1$. We show that for any constant $\epsilon : 0 < \epsilon < 1$, $P_{YI}/P_{NI} \geq (\log N)^{(1-\epsilon)/(\frac{3}{2}2^z-1)}$, where N is the size of instance H_z.

2.2.1 Construction

We will use as our building block the bit gadget in Section 2.1.1. We will vary the parameters M, X based on the sample space H_i. The sample space H_1 is same as in Section 2.1, except that instead of parameters M, X, we use new parameters M_1, X_1, which are specified later. For each accepting configuration α, let $\mathcal{P}^1_\alpha$ denote the set of X_1 canonical paths representing α in H_1. For $i \geq 2$, we generate an instance of H_i by connecting together several random instances of H_{i-1}. Graph H_i will contain a set of regular edges and a set of special edges, whose sizes are the same for all the instances of H_i. An instance of H_i contains X_1 source-sink pairs for each ordered i-tuple $(\alpha_1, \ldots, \alpha_i)$ of accepting configurations, and for each pair a canonical path is defined. Let η_i denote the number of canonical paths in any instance of H_i, $\eta_i = |\mathcal{C}|^i X_1$. In order to define the recursive construction of H_i, we need first to define the notion of concatenation of instances of H_{i-1}.

Concatenation of EDP Instances: Suppose G_1, G_2 are two instances of H_{i-1}, for some $i \geq 2$. Then *concatenation* of G_1 and G_2 is a new instance G defined as follows. Let $(\alpha_1, \ldots, \alpha_{i-1})$ be an ordered $(i-1)$-tuple of accepting configurations. Recall that each instance of H_{i-1} contains X_1 source-sink pairs representing $(\alpha_1, \ldots, \alpha_{i-1})$. Let S_1, T_1 and S_2, T_2 be the corresponding sets of source and sink vertices in G_1 and G_2, respectively. We randomly unify the vertices in T_1, S_2 in a pairwise manner. Consider any two source-sink pairs (s_1, t_1) and (s_2, t_2) corresponding to $(\alpha_1, \ldots, \alpha_{i-1})$ in G_1 and G_2, respectively, such that t_1 and s_2 are unified in G. Then (s_1, t_2) becomes a source-sink pair for graph G, and its canonical path is defined as a concatenation of the two canonical paths in G_1 and G_2. Observe that in graph G, the number of source-sink pairs remains $|\mathcal{C}|^{i-1}X_1$, the same as in G_1 and G_2. We define a concatenation of arbitrary number of instances of H_{i-1} in a similar fashion.

Definition of H_i: An instance of H_i is constructed by a recursive composition of instances of H_{i-1} and bit gadgets. We will use parameter M_i, X_i for constructing H_i. For $i \geq 2$, we define $M_i = M_1^3 M_2 ... M_{i-1}$. We also define $X_i = (|\mathcal{C}| q M_{i-1} X_{i-1})/2$ for $i \geq 2$. By our choice of parameters, we ensure that the number of special edges in an instance of H_{i-1} is X_i.

- For each accepting configuration α, and for each $j : 1 \leq j \leq q$, we build an instance $B^{i-1}(\alpha, j)$ of H_{i-1}. Each of these instances is constructed independently.

- For each accepting configuration α, we define a graph $G^i(\alpha)$ to be the concatenation of $B^{i-1}(\alpha, 1), \ldots, B^{i-1}(\alpha, q)$. A source-sink pair in the concatenated graph correponding to an $(i-1)$-tuple $(\alpha_1, \ldots, \alpha_{i-1})$ can now be viewed as a pair that corresponds to the i-tuple $(\alpha_1, \ldots, \alpha_{i-1}, \alpha)$ in $G^i(\alpha)$.

- For each proof bit Π_j, we build a bit gadget $G^i(j)$ representing it, with parameters M_i, X_i.

- The above two parts are composed together as follows. Consider some accepting configuration α, and let $a_1, \ldots, a_q$ be the corresponding query bits. Fix some $j : 1 \leq j \leq q$.

  On one hand, we have a bit gadget $G^i(a_j)$, which contains X_i canonical paths corresponding to α. Let S_j, T_j denote the set of sources and destinations of these paths. For each source $s \in S_j$, let $f(s) \in T_j$ denote its corresponding destination.

On the other hand, graph $G^i(\alpha)$ contains as sub-graph instance $B^{i-1}(\alpha, j)$ of H_{i-1}, which has X_i special edges. Let A denote this set of special edges, and let L and R denote the sets of their left and right endpoints. We remove these edges from our graph. Instead, we unify vertices in L and S_j (in pairwise manner), and we unify vertices in R and T_j, as follows. Let $e = (\ell, r) \in A$, and assume we unified ℓ with some source $s \in S_j$. We then unify r and $f(s)$.

Source-sink pairs of the new instance are the union of the source-sink pairs in graphs G^i_α for $\alpha \in \mathcal{C}$. The set of special edges in the new instance of H_i is the union of the special edges in bit gadgets $G^i(j)$, for all proof bits j. All the other edges are regular. Notice that the number of special edges in H_i is indeed X_{i+1}: Recall that for each configuration $\alpha \in \mathcal{C}$, graph $G^i(\alpha)$ is a concatenation of q instances of H^{i-1}, each of them containing X_i special edges. Each such special edge is replaced by a canonical path in $G^i(j)$ for some proof bit j. A canonical path of a bit gadget has M_i special edges, and each special edge is shared by two such paths. Therefore, the total number of special edges in H_i is $|\mathcal{C}|qX_i\frac{M_i}{2} = X_{i+1}$.

Also, note that the total number of canonical paths that go through a special edge in any instance of H_i is exactly 2^i.

Size of an instance of H_z: We will set the base parameters as $M_1 = 2^{\lambda k^2}$ and $X_1 = 2^{2^{\lambda k^2(\frac{3}{2}2^z - 1) + \lambda k}}$. Let us now bound M_i. It is easy to see that $M_2 = M_1^3$, and for all $i : 2 < i \le z$, $M_i = M_{i-1}^2 = M_1^{\frac{3}{4}2^i}$. Therefore, for all $2 < i \le z$, we have $M_i = 2^{\lambda k^2 \frac{3}{4} 2^i} < 2^{\lambda k^2 2^i}$.

Let N_i denote the size of an instance of H_i, and let ℓ_i denote the length of each canonical paths in an instance of H_i. Recall that $\eta_i = |\mathcal{C}|^i X_1$ is the number of canonical paths in H_i. Clearly, $N_i \le \ell_i \eta_i$.

To bound ℓ_1, recall that each canonical path traverses q gadgets, and length of a canonical path inside each gadget is at most $3M_1$. So, $\ell_1 \le 4qM_1$. The recursive formula for ℓ_i, where $i > 1$ is calculated as follows. A canonical path in H_i consists of q canonical paths in H_{i-1}. Additionally, each special edge of H_{i-1} is replaced with a canonical path in gadget $G^i(j)$ (where j is some proof bit index). The length of the canonical path inside $G^i(j)$ is at most $3M_i$, and the number of special edges on path ℓ_i is at most $q\ell_{i-1}$. Therefore, $\ell_i \le q\ell_{i-1} + q\ell_{i-1} \cdot 3M_i \le 4qM_i\ell_{i-1} \le (4q)^i M_1 M_2 \cdots M_i \le (4q)^i M_1^{\frac{3}{4}2^{i+1}-2}$. Thus the size N_z of an instance of H_z can be bounded as $N_z \le \ell_z \eta_z \le |\mathcal{C}|^z X_1 (4q)^z M_1^{\frac{3}{4}2^{z+1}-2} \le X_1 2^{2\lambda r z}$ when $z = O(\log\log\log n)$.

Notice that $X_1 = 2^{2^{\lambda k^2(\frac{3}{2}2^z - 1) + \lambda k}}$. As $r = O(\log n)$, the overall construction size is $O(n^{\mathrm{poly}\log n})$.

2.2.2 Yes Instance: ϕ is Satisfiable

In the YES-INSTANCE , there is a PCP proof, for which the acceptance probability is at least $2^{-\lambda}$. For each random string r satisfied by this proof, let $c(r)$ be the corresponding accepting configuration. The proof of the following lemma is omitted due to lack of space.

Lemma 4 *If ϕ is a* YES-INSTANCE *, then for each $i : 1 \le i \le z$, graph H_i contains a collection of $P_{YI} = |\mathcal{C}|^i X_1 / 2^{(2\lambda k + \lambda)i}$ edge-disjoint canonical paths.*

2.2.3 No Instances: ϕ is Unsatisfiable

Assume ϕ is a NO-INSTANCE . As before, we will bound the number of canonical paths ($\mathcal{P}_1$), long non-canonical paths ($\mathcal{P}_2$), and short canonical paths ($\mathcal{P}_3$) in any solution that has congestion at most $2^z - 1$.

Canonical Paths: Recall that in order to construct our final graph H_z, we construct, for each proof bit Π_j, for each $i : 1 \le i \le z$, many instances of bit gadget $G^i(j)$, with parameter M_i. We define a parameter $\Delta_i = \frac{M_i}{8\log M_i}$, which replaces the parameter Δ in the definition of a bad gadget. Let $\mathcal{B}_1$ be the (bad) event that any of these bit gadgets is bad. The following is a simple corollary of Lemma 2.

Corollary 2 *The probability of the bad event $\mathcal{B}_1$ is bounded by $\frac{1}{\mathrm{poly}\, n}$.*

Theorem 5 *If event $\mathcal{B}_1$ does not occur, then for each $i : 1 \le i \le z$, any collection of more than $\frac{|\mathcal{C}|^i \cdot (9q^2)^i \cdot X_1}{M_1}$ canonical paths in graph H_i, causes congestion of 2^i.*

Proof: The proof is by induction on i. For each i, we bound the maximum number of canonical paths, for which congestion is less than 2^i. The analysis of the base case, where $i = 1$ is similar to the analysis presented in Section 2.1. Recall that the number of canonical paths in any solution with congestion 1 is at most $\frac{X_1}{\Delta_1}\sum_j n_j + \frac{2^{\lambda r} X_1}{2^{\lambda k^2}} \le \frac{|\mathcal{C}|qX_1 \cdot 8\log M_1}{M_1} + \frac{|\mathcal{C}|X_1}{2^{\lambda k^2}} \le \frac{|\mathcal{C}|qX_1 \cdot 8\lambda k^2}{M_1} + \frac{|\mathcal{C}|X_1}{2^{\lambda k^2}} \le \frac{|\mathcal{C}|(9q^2)X_1}{M_1}$.

Assume now the theorem holds for $i - 1$, and consider H_i. Let $\mathcal{P}_1^i$ be any collection of canonical paths in H_i, such that their congestion is less than 2^i. We partition the set $\mathcal{P}_1^i$ as follows: for each configuration $\alpha \in \mathcal{C}$, let $\mathcal{Q}_\alpha^i$ be the paths of $\mathcal{P}_1^i$ that correspond to paths in $G^i(\alpha)$.

Definition: Let $\alpha \in \mathcal{C}$ be an accepting configuration. We say that α is *congested* iff $|\mathcal{Q}_\alpha^i| \ge 2\frac{|\mathcal{C}|^{i-1} \cdot (9q^2)^{i-1} \cdot X_1}{M_1}$.

We now proceed in two steps. First, we prove that if α is congested, then for each $j : 1 \le j \le q$, many of the special edges in $B^{i-1}(\alpha, j)$ have congestion 2^{i-1}. The second step is proving that the number of congested configurations is small (otherwise the overall congestion is 2^i).

Lemma 6 *Suppose α is congested and event $\mathcal{B}_1$ does not occur. Then for each $j \in \{1..q\}$, at least $\frac{X_i}{M_1^2 M_2 \cdots M_{i-1}}$ special edges in instance $B^{i-1}(\alpha, j)$ of H_{i-1}, have congestion 2^{i-1}.*

Lemma 7 *If ϕ is a* NO-INSTANCE *and the event $\mathcal{B}_1$ does not occur, then in any solution of H_i with congestion at most $2^i - 1$, no more than $\frac{2^{i+1}|\mathcal{C}|q^2}{M_1}$ configurations can be congested.*

We are now ready to bound the number of canonical paths in $\mathcal{P}_1 = \mathcal{P}_1^i$. Each congested configuration contributes at most $|\mathcal{C}|^{i-1}X_1$ paths to $\mathcal{P}_1$, and by Lemma 7, we have at most $(2^{i+1}|\mathcal{C}|q^2)/M_1$ congested configurations. Each non-congested configuration contributes at most $2\frac{|\mathcal{C}|^{i-1}\cdot(9q^2)^{i-1}\cdot X_1}{M_1}$ paths. Thus, $|\mathcal{P}_1^i| \le |\mathcal{C}|^{i-1}X_1 \cdot \frac{2^{i+1}|\mathcal{C}|q^2}{M_1} + 2\frac{|\mathcal{C}|^{i-1}\cdot(9q^2)^{i-1}\cdot X_1}{M_1} \cdot |\mathcal{C}| \le \frac{2^{i+1}|\mathcal{C}|^i q^2 X_1}{M_1} + 2\frac{|\mathcal{C}|^i\cdot(9q^2)^{i-1}\cdot X_1}{M_1} \le \frac{|\mathcal{C}|^i X_1 (9q^2)^i}{M_1}$. □

Since $\mathcal{P}_1 = \mathcal{P}_1^z$, we get $\frac{P_{YI}}{|\mathcal{P}_1|} \ge \frac{2^{\lambda k^2}}{(9q^2)^z\cdot 2^{(2\lambda k+\lambda)z}} \ge 2^{\lambda k^2 - 2\lambda kz - \lambda z - 3z\log(\lambda k)} \ge 2^{\lambda k^2 - 3\lambda kz}$.

Long Non-Canonical Paths: Recall that the length of each canonical path in an instance of H_z is $\ell_z \le (4q)^z M_1^{\frac{3}{4}(2^{z+1}-2)} \le (4q)^z 2^{\lambda k^2 \frac{3}{4}(2^{z+1}-2)}$. A non-canonical path is called *long* if its length is at least $g = \ell_z \gamma$ where $\gamma = 2^{\lambda k^2}$. Otherwise, it is called *short*. Let $\mathcal{P}_2$ denote the set of long non-canonical paths in any solution that has congestion less than c. Each edge in our final graph participates in at least one canonical path. Thus, the total number of edges is at most $\eta_z \ell_z$. As the congestion on each edge is less than 2^z, we have that $|\mathcal{P}_2| \le \frac{2^z \eta_z \ell_z}{g} = \frac{2^z|\mathcal{C}|^z X_1}{\gamma} \le P_{YI}\frac{2^z 2^{(2\lambda k+\lambda)z}}{2^{\lambda k^2}} \le \frac{P_{YI}}{2^{\lambda(k^2-3kz)}}$.

Short Non-Canonical Paths: We next bound the size of $\mathcal{P}_3$, the set of short non-canonical paths in our solution. Suppose there is a short non-canonical path $P \in \mathcal{P}_3$ connecting some source and destination pair (s, t). This path must form a cycle of length at most $g + \ell_z \le 2g$ with the canonical $s - t$ path. Moreover, at least one edge on the cycle lies on P. Let K denote the number of cycles of length at most $2g$ in our graph. Then $|\mathcal{P}_3| \le 2cg \cdot K$ (since congestion is at most c). Our goal is to show that with high probability, K is small. The proof of the following lemma is similar to Lemma 3.

Lemma 8 *With probability at least $\frac{2}{3}$, $K \le 3|\mathcal{C}|^{(2g+2)z}$.*

We say that the *bad event* $\mathcal{B}_2$ occurs if $K > 3|\mathcal{C}|^{(2g+2)z}$. If $\mathcal{B}_2$ does not happen, then we have: $|\mathcal{P}_3| \le 2cg \cdot (3|\mathcal{C}|^{(2g+2)z}) \le |\mathcal{C}|^{3gz} \le 2^{(\lambda(r+2k)z)\cdot(4q)^z 2^{\lambda k^2(\frac{3}{4}2^{z+1}-2)}\cdot 2^{\lambda k^2}} \le 2^{2^{\lambda k^2(\frac{3}{4}2^{z+1}-1)+\lambda k}} \le X_1 \le \frac{P_{YI}}{2^{\lambda(k^2-3kz)}}$.

2.2.4 Putting it Together

If the events $\mathcal{B}_1$ and $\mathcal{B}_2$ do not happen, the gap between the yes and the no instances is $\Omega(2^{\lambda(k^2-3kz)})$. For any $\epsilon' > 0$, we can choose sufficiently large k such that the gap is $(\log N_z)^{(1-\epsilon')/(\frac{3}{4}2^{z+1}-1)} = (\log N_z)^{(1-\epsilon')/(\frac{3}{2}c+\frac{1}{2})}$ and the size of the instance is bounded by $n^{\mathrm{polylog(n)}}$.

When c is allowed to grow up to $\frac{\log\log n}{(\log\log\log n)^2}$ for any $\epsilon > 0$, the gap term $\Omega(2^{\lambda(k^2-3kz)})$ yields the desired gap only when we allow k to grow to $z = \log\log\log n$. Following [21], it can be shown that using $r = O(k^2\log n)$ random bits, we can get once again completeness at least $1/2$ and soundness at most $1/2^{k^2}$. To keep the construction size bounded by $(n^{\mathrm{polylog(n)}})$, we now choose λ to be a large constant. The rest of the proof remains similar to the one presented above.

Extension to ANFwC: To show the hardness of ANFwC, classify each routed pair to be of type A or B based on how much flow is routed on canonical versus non-canonical paths. It is type A if more than a $(c-1)/c$-fraction of the flow is routed on the pair's canonical path, and type B otherwise. It is easy to see that no more than c type A pairs can traverse a special edge without causing a congestion greater than $c-1$. Thus essentially the same analysis as given above for $\mathcal{P}_1$ applies. For type B pairs, we proceed as above for $\mathcal{P}_2$ and $\mathcal{P}_3$ noting that for each routed pair, we have only $1/c$-fraction of the flow to be supported.

2.3. A Simple Integrality Gap Construction

We will construct, for each integral $c \le O((\log\log n)/(\log\log\log n))$, an EDPwC instance of size $O(n\log n)$ for which the integrality gap of the multicommodity flow relaxation is $\Omega((\frac{\log n}{(\log\log n)^2})^{1/c})/c)$ when congestion is restricted to be *strictly less than* c. Our construction will use two parameters, $\beta_1 = \frac{1}{4}(\frac{\log n}{150(\log\log n)^2})^{1/c}$ and $\beta_2 = 6(2\beta_1)^{c-1}\ln\beta_1$. The integrality gap of our EDP instance will be $\Omega(\beta_1/c)$.

2.3.1 Auxiliary Hypergraph Construction

Our starting point is a random hypergraph H with vertex set $V(H) = \{v_1, \ldots, v_n\}$, and $\beta_2 n$ hyper-edges, $h_1, \ldots, h_{n\beta_2}$. Each hyper-edge h_i, for $1 \le i \le n\beta_2$ is a c-tuple of vertices, chosen randomly and independently. Our EDP instance will be derived from the hypergraph H.

We now establish some properties of H. Let $S \subseteq V(H)$ be a subset of vertices of size n/β_1. We say that S is *bad* if it contains none of the $n\beta_2$ hyper-edges. We say that event $\mathcal{E}_1$ happens, iff there is at least one bad subset $S \subseteq V(H)$ of size n/β_1. Also, given a vertex $v \in V(H)$, we say that it is a *high-degree* vertex, if it participates in more than $100\beta_2 c$ hyper-edges in H. We say that event $\mathcal{E}_2$ happens, if the number of high-degree vertices in H is greater than n/β_1. The proof of the lemma below is straightforward.

Lemma 9 *The probability that either event $\mathcal{E}_1$ or $\mathcal{E}_2$ occurs is at most* $1/2$.

2.3.2 The EDPwC Instance

The construction of the EDPwC instance G is based on hyper-graph H defined above. For each vertex $v \in V(H)$, graph G contains a source and sink pair $(s(v), t(v))$. Additionally, for each hyper-edge $h_i : 1 \leq i \leq \beta_2 n$, it contains two vertices ℓ_i, r_i, which are connected by a *special edge*. Consider now some vertex $v \in V$, and assume it participates in hyper-edges $h_{i_1}, h_{i_2}, \ldots, h_{i_k}$, where $i_1 < i_2 < \cdots < i_k$. We add the following *regular edges* to graph G: $(s(v), \ell_{i_1})$, $(r_{i_k}, t(v))$, and for each $j : 1 \leq j \leq k-1$, we add a regular edge $(r_{i_j}, \ell_{i_{j+1}})$. We define a *canonical path* corresponding to v as follows: $P(v) = (s(v), \ell_{i_1}, r_{i_1}, \ldots, \ell_{i_k}, r_{i_k}, t(v))$.

Let $g > 2$ be some fixed integer, and let K_g be the total number of cycles of length at most g in G. We say that event $\mathcal{E}_3$ happens, if $K_g > (6\beta_2 c)^{g+1}$.

Lemma 10 *The probability that $\mathcal{E}_3$ happens is at most* $\frac{1}{4}$.

With probability at least $1/4$, none of the events $\mathcal{E}_1$, $\mathcal{E}_2$, and $\mathcal{E}_3$ happen; we assume this from now on.

2.3.3 Integrality Gap Analysis

The fractional solution can route at least $\frac{n}{c}$ units of flow, by sending $\frac{1}{c}$ units of flow on each canonical path. On the other hands, if events $\mathcal{E}_1$, $\mathcal{E}_2$, and $\mathcal{E}_3$ do not occur, then we can show that no integral solution can route more than $4n/\beta_1$. Thus the integrality gap is at least $\frac{\beta_1}{4c}$, giving us the desired bound.

3. The AZ Construction

The AZ construction uses a reduction from the Raz verifier for MAX3SAT(5) and builds an EDPwC instance in a similar manner to the hardness example for the Congestion Minimization problem presented in [2]. A 3SAT(5) formula has n variables and $5n/3$ clauses where each variable appears in exactly 5 distinct clauses and each clause contains exactly 3 literals. The MAX3SAT(5) problem aims to find an assignment that maximizes the number of satisfied clauses. A 3SAT(5) formula is called a *yes-instance* if it is satisfiable; it is called a *no-instance* if no assignment satisfies more than a $1-\varepsilon$ fraction of the clauses for some constant $\varepsilon > 0$. It follows from the PCP theorem that it is NP-hard to distinguish between yes-instances and no-instances. Given a 3SAT(5) instance ϕ we show that if ϕ is a yes-instance then many demands can be routed on edge-disjoint paths. If ϕ is a no-instance then with high probability we can only route a small number of demands unless some edge has congestion higher than c. From now on, we use w, instead of c, to denote the congestion parameter since we use c to denote "clauses" in ϕ. In order to avoid confusion with the size of ϕ we shall use N and M to denote the number of nodes and edges in the EDPwC instance. Most of the proofs in the section are deferred to the full version [3].

3.1 Construction

Raz verifier. A Raz verifier with ℓ repetitions is defined as follows [27]. A verifier interacts with 2 provers, a clause prover (c-prover) and a variable prover (v-prover). Given a 3SAT(5) formula ϕ, the verifier sends the c-prover a clause query (c-query) that consists of ℓ clauses $c_1, \ldots, c_\ell$ chosen uniformly at random. It also sends the v-prover a variable query (v-query) that consists of one variable $v_1, \ldots, v_\ell$ chosen uniformly at random from each of the ℓ clauses. The c-prover sends back the assignment of every variable in clauses $c_1, \ldots, c_\ell$ and the v-prover sends back the assignment of the variables $v_1, \ldots, v_\ell$. The verifier *accepts* ϕ if all the ℓ clauses are satisfied and the two provers give consistent assignment to the ℓ variables. The verifier *rejects* ϕ otherwise.

Suppose ϕ has n variables, then ϕ has $5n/3$ clauses. Clearly, a Raz verifier with ℓ repetitions has $Q_c := (5n/3)^\ell$ distinct c-queries each of which has $A_c := 7^\ell$ answers. It also has $Q_v := n^\ell$ distinct v-queries each of which has $A_v := 2^\ell$ answers. Since the verifier only queries variables that appear in a c-query, the number of distinct clause-variable query pairs is $R := (5n)^\ell$. Each clause appears in $R/Q_c = 3^\ell$ c-v query pairs and each variable appears in $R/Q_v = 5^\ell$ c-v query pairs. It is clear that for each answer to a c-query there is exactly one answer to a v-query that would cause the verifier to accept (since the assignment to the variable in the answer to the v-query must match the corresponding assignment in the answer to the c-query). Hence for each c-v pair there are $F := 7^\ell$ accepting interactions.

Theorem 11 **[27]** *There is a universal constant $\alpha > 1$ such that if ϕ is a yes-instance there is a proof system in which the verifier always accepts; if ϕ is a no-instance the verifier accepts with probability less than $\alpha^{-\ell}$.*

Given a 3SAT(5) formula ϕ we first construct the two-prover interactive proof system and represent it in a graph that we call the *proof system graph*, P. We then transform this proof system graph into an EDPwC instance on a graph that we call the *transformed graph*, T.

In defining these two graphs, we do not use the convention of specifying their node sets and edge sets. Instead we first specify a set of paths that exist in the graph. We then add edges to the graph to ensure that these paths are realizable. Our graph construction involves many parameters

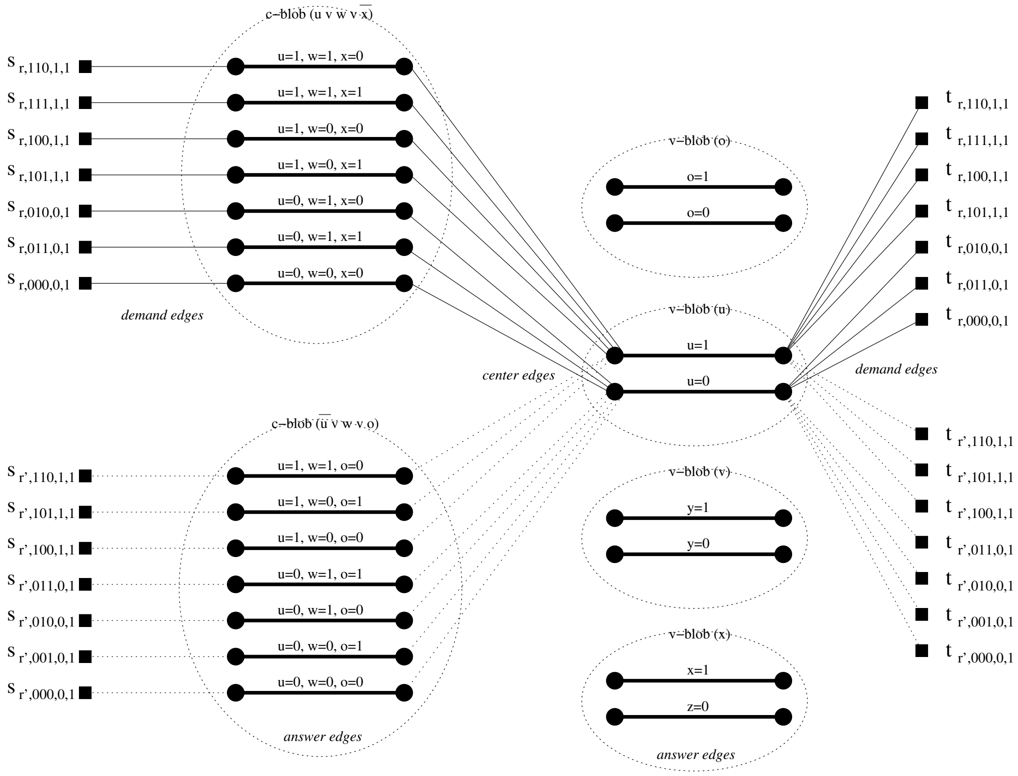

Figure 3. Graph P for a proof system, $\phi = (u \vee w \vee \bar{x}) \wedge (\bar{u} \vee w \vee o)$, $\ell = 1$ and $Y = 1$. The figure shows 14 out of 42 demands.

which we define at the end of this section. Throughout the paper we use subscript c and v when discussing quantities pertaining to clause and variable, and we omit the subscript when we do not distinguish them.

Proof System Graph P. In the proof system graph P, for each possible answer a we have an answer edge which we also denote a. For each query q we group together all the possible answers to q and refer to this group as a query blob which we also denote q. A clause query blob (c-blob) contains A_c edges that we call *answer* edges and a variable query blob (v-blob) contains A_v answer edges. We use r to denote a c-v query pair. For each accepting interaction (r, a_c, a_v) we have Y demands $d_{r,a_c,a_v,y}$, $1 \leq y \leq Y$, each of which has a source node $s_{r,a_c,a_v,y}$, a destination node $t_{r,a_c,a_v,y}$ and routes one unit of flow. [1] Demand $d_{r,a_c,a_v,y}$ has a special path p that we refer to as a *canonical path*.

This path starts at node $s_{r,a_c,a_v,y}$, passes through edges a_c, a_v and ends at node $t_{r,a_c,a_v,y}$. In order for this to be possible, we place a *center* edge between a_c and a_v, a *demand* edge between $s_{r,y}$ and a_v, and a demand edge between a_c and $t_{r,y}$. We allow parallel demand edges and center edges so that no two canonical paths share a common edge other than answer edges.

Let us use an example to illustrate the above construction of P, shown in Figure 3. Let the repetition parameter $\ell = 1$. Let $\phi = (u \vee w \vee \bar{x}) \wedge (\bar{u} \vee w \vee o)$. For a query pair r that queries clause $u \vee w \vee \bar{x}$ and variable u, there are 7 accepting interactions, namely the 7 satisfying assignments for the clause and the assignment of u consistent with the clause. Suppose $Y = 1$. We define one demand $d_{r,a_c,a_v,1}$ for each of the 7 accepting interactions. Each of these demands has an associated canonical path. These 7 canonical paths are shown in solid lines in Figure 3. The demand for query pair r' that queries clause $\bar{u} \vee w \vee o$ and variable u also has 7 demands. The associated canonical paths share the answer edges in the v-blob u with canonical paths associated with r. The canonical paths for r' are shown in dotted lines in Figure 3. We did not draw $4 \times 7 = 28$ other demands and their canonical paths.

If $Y > 1$ then for each accepting interaction (r, a_c, a_v) we define Y demands, each of which has its own source node, destination node and canonical path. The canonical paths for these Y demands share the answer edges in c-blob $u \vee w \vee \bar{x}$ and v-blob u, but have distinct center edges and demand edges.

[1] In [2] demands corresponded to c-v query pairs in the proof system. Each demand is associated with multiple canonical paths and each canonical path corresponded to an accepting interaction. In this construction for EDPwC, each demand corresponds to an accepting interaction and has only a single canonical path.

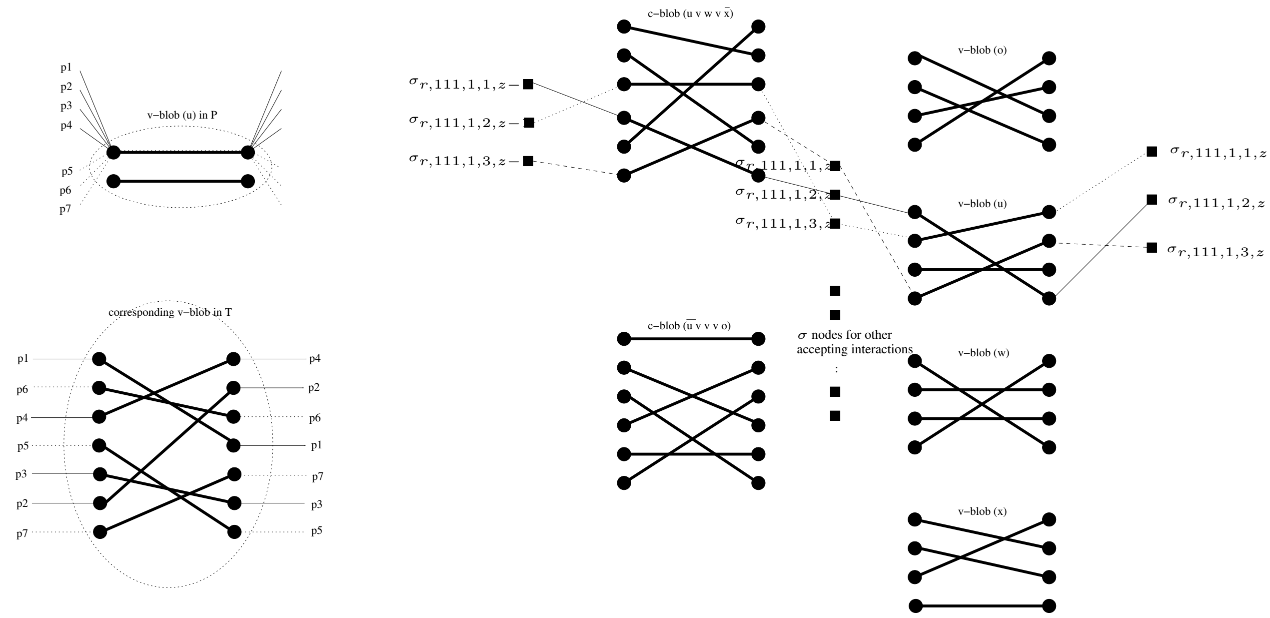

Figure 4. (Left) An illustration of property T-1. (Upper left) Using the example shown in Figure 3, seven paths $p_1, \ldots, p_7$ go through the answer edge $u = 1$ for the blob for variable u in graph P. (Lower left) Image edges in each of the Z corresponding v-blobs in T are created via random bipartite matching. Paths $p_1, \ldots, p_7$ are randomly mapped to distinct image edges in each blob.

(Right) Level z of the transformed graph T. Consider $q_c = u \vee w \vee \bar{x}$ and $q_v = u$. Let $a_c = 111$ represent $u = w = x = 1$ and let $a_v = 1$ represent $u = 1$. Let (r, a_c, a_v) be an accepting interaction. For $Y = 3$, the figure shows the canonical paths of 3 demands $d_{r,a_c,a_v,y}$, where $1 \le y \le Y$. For the canonical path shown in solid line, $y_{2z-2} = 1$, $y_{2z-1} = 2$ and $y_{2z} = 2$. (Fewer than the actual number of image edges are shown in each blob.)

The following property P-1 holds for graph P. To see this, we note that for a fixed clause and a fixed variable, the number of satisfying assignments for the clause for which the variable is set to 1 is either 3 or 4 and the number of satisfying assignments for which the variable is set to 0 is also either 3 or 4.

P-1. *Consider the set of demands that correspond to the same c-v query pair and the same y. No two demands from the set can share a common answer edge a_c in a c-blob. At most 4^ℓ demands from the set can share an answer edge a_v in a v-blob.*

Among all demands, exactly $I_c := YR/Q_c$ canonical paths share each answer edge a_c and at most $I_v := 4^\ell YR/Q_v$ canonical paths share each answer edge a_v.

EDPwC Instance on Transformed Graph T. We now construct an EDPwC instance on the transformed graph T. The graph T consists of Z levels. Each level of T consists of Q_c c-blobs and Q_v v-blobs and they correspond one-to-one to those in P. We use $b_{q,z}$ to denote the blob at level z of T that corresponds to query blob q of P, Each c-blob (resp. v-blob) in T consists of a *random bipartite matching* between I_c (resp. I_v) left nodes and I_c (resp. I_v) right nodes. We use the term *image edge* to refer to the edges in these random matchings. For each accepting interaction (r, a_c, a_v), we also have 2 groups of *intermediate nodes* σ at each level z, one group situated between blobs $b_{q_c,z}$ and $b_{q_v,z}$ and the other situated between $b_{q_v,z}$ and $b_{q_c,z+1}$. Each group consists of Y nodes of the form $\sigma_{r,a_c,a_v,y,z}$ where $y = 1, \ldots, Y$.

Like P, we have Y demands $d_{r,a_c,a_v,y}$, $1 \le y \le Y$, for each accepting interaction (r, a_c, a_v), and each demand has one canonical path. The canonical path for demand $d_{r,a_c,a_v,y}$ in T goes through $2Z + 1$ intermediate nodes and we denote them $\sigma_{r,a_c,a_v,y_0,0}$, $\sigma_{r,a_c,a_v,y_{2z-1},z}$ and $\sigma_{r,a_c,a_v,y_{2z},z}$, for $1 \le z \le Z$. The node $\sigma_{r,a_c,a_v,y_0,0}$ also acts as the source and $\sigma_{r,a_c,a_v,y_{2Z},2Z}$ as the destination of the demand.

We specify the edge set in T by describing the mapping of each canonical path from P to T. The essence of the mapping is that any two canonical paths sharing an answer edge in a blob q in P are mapped to distinct image edges in the corresponding blobs $b_{q,z}$, $1 \le z \le Z$, in T. This property is summarized later on in property T-1. The intermediate nodes σ are introduced so that there is some randomness when connecting two consecutive image edges on a canonical path. In particular, the Y canonical paths corresponding to the same accepting interaction (r, a_c, a_v) go through a random permutation after going through every query blob in T. We quantify this randomness in property T-2. and exploit this property to prove the high-girth property of T in Theorem 18.

In the following we describe the canonical paths. We walk along the canonical path for each demand from left to right. Demand $d_{r,a_c,a_v,y}$ starts with its source node $s_{r,a_c,a_v,y}$ on the left, which is also $\sigma_{r,a_c,a_v,y_0,0}$. This σ-node goes to a *random* left node of the blob $b_{q_c,1}$ (or more generally blob $b_{q_c,z}$ at level z). This left node is chosen subject to the constraint that *no two canonical paths that share a common answer edge a_c in P can go to the same left node in $b_{q_c,1}$* (or more generally $b_{q_c,z}$). This is always possible due to property P-1 and the fact that $b_{q_c,z}$ has I_c left nodes. We add an edge between the chosen σ-node and the chosen left node. The canonical path then passes through the image edge incident to the chosen left node. To proceed from the right node in $b_{q_c,z}$ we consider the Y demands corresponding to the same accepting interaction (r, a_c, a_v). We put a *random bipartite matching* between the Y right nodes in $b_{q_c,z}$ that these Y demands go through and the Y σ-nodes of the form $\sigma_{r,a_c,a_v,y,z}$, where $1 \leq y \leq Y$, that are to the right of $b_{q_c,z}$. This matching defines node $\sigma_{r,a_c,a_v,y_{2z-1},z}$ for demand $d_{r,a_c,a_v,y}$.

From $\sigma_{r,a_c,a_v,y_{2z-1},z}$, demand $d_{r,a_c,a_v,y}$ goes through blob $b_{q_v,z}$ in a similar manner. That is, it goes to a *random* left node of the blob $b_{q_v,z}$, which is chosen subject to the constraint that *no two canonical paths that share a common answer a_v in P can go to the same left node in $b_{q_v,z}$*. The canonical path then passes through the image edge incident to this left node. To proceed from the right node in $b_{q_v,z}$ we consider the Y demands corresponding to the same accepting interaction (r, a_c, a_v). Again, we put a *random bipartite matching* between the Y right nodes in $b_{q_v,z}$ that these Y demands go through and the Y σ-nodes of the form $\sigma_{r,a_c,a_v,y,z}$, where $1 \leq y \leq Y$, that are to the right of $b_{q_v,z}$.

After passing through all Z levels, the canonical path for demand $d_{r,a_c,a_v,y}$ ends at the node $\sigma_{r,a_c,a_v,y_{2Z},Z}$, which we make the destination node of the demand $d_{r,a_c,a_v,y}$. See Figure 4 (Right) for an illustration of the canonical path for one demand. Graph T has the following two properties:

T-1 *Consider any answer a_c to query q_c. The I_c demands in P that are routed through the answer edge a_c have canonical paths in T pass through* distinct *image edges in $b_{q_c,z}$. Similarly for any answer edge a_v, the at most I_v demands in P that are routed through a_v have canonical paths in T that pass through distinct image edges in $b_{q_c,z}$.*

T-2 *Consider all the edges that could potentially exist in T before the random choices are made. The probability that any such edge exists in T is at most $1/Y$. Moreover, consider any potential edge e. If any fixed set of g edges exist in T and none are to the right of e, then the probability that e exists is at most $1/(Y - g)$.*

Parameters. Given a $3SAT(5)$ formular ϕ with n variables, let ℓ be the Raz verifier repetition parameter. We use the following additional parameters in constructing graphs P and T.

$Q_c = (5n/3)^\ell$	no. of c-blobs in P and per level in T; no. of c-queries	(1)
$Q_v = n^\ell$	no. of v-blobs in P and per level in T; no. of v-queries	(2)
$A_c = 7^\ell$	no. of answer edges per c-blob in P; no. of answers to a c-query	(3)
$A_v = 2^\ell$	no. of answer edges per v-blob in P; no. of answers to a v-query	(4)
$I_c = Y \cdot 3^\ell$	no. of image edges per c-blob in T	(5)
$I_v = Y \cdot 20^\ell$	no. of image edges per v-blob in T	(6)
$R = (5n)^\ell$	no. of c-v query pairs	(7)
$F = 7^\ell$	no. of accepting interactions per c-v query pair	(8)
$D_c = Y \cdot 3^\ell$	$D_cF =$ no. of demands per c-blob in P and T	(9)
$D_v = Y \cdot 5^\ell$	$D_vF =$ no. of demands per v-blob in P and T	(10)
FYR	no. of demands	

In order to define Y, the number of demands per c-v accepting interaction, and Z, the number of levels in T, we introduce two new parameters, k and h. The free parameter $h \geq 2$ is used to define the concept of *heavy* blobs in P. The Raz repetition parameter ℓ and the congestion parameter w are defined in terms of h. The parameter k is related to the length of non-canonical paths in T.

h	def. of heavy blob in P	(11)
$w = h - 1$	max allowed congestion	(12)
$\ell = h^{-1} \log \log n$	Raz repetition parameter	(13)
$\beta = \alpha^\ell/(h-1)^2$	def. of heavy edge in P	(14)
$Z = (2\beta \cdot 8^\ell)^h$	no. of levels in T	(15)
$k = Z\sqrt{Z}$	noncanonical path length in T	(16)

$$Y = k \cdot (A_c + 1)^k \cdot 3(24 \cdot 7^\ell ZR)^{k+1}$$

no. of demands per c-v query pair (17)

Let M be the number of edges in T. We have YRZ image edges from c-blobs and $YRZ \cdot 4^\ell$ image edges from v-blobs. Canonical paths do not share common edges outside blobs in T and there are $7^\ell(4Z)YR$ such edges. Therefore,

$$M = YRZ + YRZ \cdot 4^\ell + 7^\ell(4Z)YR. \tag{18}$$

Note that we choose h such that $\ell h = \log \log n$. This ensures that $Z = \mathrm{polylog}(\ )n$ and hence $M = n^{\text{polylog } n}$.

3.2 Analysis

At a high level our proof proceeds as follows. We use a solution to the EDPwC instance on T to determine whether the original 3SAT(5) instance ϕ (that defined the proof system) is a yes-instance or a no-instance. We first argue,

Lemma 12 *If ϕ is a yes-instance then YR demands can be routed in T on edge-disjoint paths.*

In this case, the two provers can agree on a satisfying assignment for ϕ and this assignment defines one accepting interaction per query pair. Each accepting interaction corresponds to Y demands and we include these YR demands in the solution. Note that this satisfying assignment also defines one answer edge per query blob in P such that the canonical paths for these YR demands only go through these chosen edges. By property T-1, these canonical paths are mapped to distinct image edges in every blob of T. Therefore, we have chosen YR demands that have edge-disjoint canonical paths in T.

No-instance. We concentrate on the case that ϕ is a no-instance. If many demands are routed along canonical paths in a solution to the EDPwC instance on T, we show that these demands must be routed through a large number of answer edges in *some* blob in P. Otherwise, a small number of answer edges would be used in *every* blob and we could use them to define a pair of provers that would violate the error probability of the proof system defined in Theorem 11. (In the extreme example in which only one answer edge is used in every query blob in P, these edges define the two provers.) This property is summarized in Lemma 13.

We first define some terminologies. Note that our random construction defines a random mapping $\mathcal{M}$ between canonical paths in P and canonical paths in T. Consider a solution $\mathcal{T}$ to the EDPwC instance on T and let us focus on the demands that are routed along their canonical paths under $\mathcal{T}$. We let $\mathcal{P} = \mathcal{P}(\mathcal{T}, \mathcal{M})$ be the corresponding canonical paths in P under the random mapping $\mathcal{M}$. We use $\mathcal{T} = \mathcal{T}(\mathcal{P}, \mathcal{M})$ to denote the inverse mapping. We define an answer edge a in a blob q in P to be *heavy* if at least $D/(\beta A)$ canonical paths go through a under $\mathcal{P}$. In other words, an edge is heavy if of all canonical paths that go through a blob in P a fraction of at least $1/(\beta AF)$ go through the edge. We say a query blob is *heavy* if it has at least h heavy answer edges under $\mathcal{P}$. We note that heavy edges and heavy blobs in P are defined under a particular solution. If a solution $\mathcal{T}$ routes more than $3YR/\beta$ demands on canonical paths, then we call it a *canonical solution* for T.

Lemma 13 *If $\mathcal{T}$ is a canonical solution, P has a heavy blob under $\mathcal{P}(\mathcal{T}, \mathcal{M})$.*

In the following, we use property T-1 to show that for a heavy blob in P, one of the Z corresponding blobs in T necessarily has congestion at least h with high probability. Therefore, if congestion is less than h on all edges, only a small number of demands can be routed on canonical paths. For this purpose we define a set of bad events. Let q be a heavy query blob in P, let $H = \{a_1, \ldots, a_h\}$ be a set of h answer edges and let E_{a_i}, for $1 \leq i \leq h$, be a set of $D/\beta A$ canonical paths in P that pass through the answer edge a_i. Let $B(q, E_{a_1}, \ldots, E_{a_h})$ be the bad event that under $\mathcal{M}$, the images of the paths in $E_{a_1}, \ldots, E_{a_h}$ in T create congestion less than h.

The bad event $B(q, E_{a_1}, \ldots, E_{a_h})$ is related to the following balls-and-bins game. We are given I bins and we throw n_i balls into n_i distinct bins during the ith round where $n_i \leq I$. What is the maximum number of balls in a bin at the end of A rounds? Here, A represents the number of edges in q and I represents the number of edges in a blob in T that corresponds to q. The n_i's represent the number of demands that are routed along the ith answer edge in q. In particular, $n_{a_1}, \ldots, n_{a_h}$ represent the counts for heavy edges and are at least $D/(\beta A)$. The bad event $B(q, E_{a_1}, \ldots, E_{a_h})$ happens when the maximum number of balls in a bin at the end of A rounds is smaller than h.

Lemma 14 *The probability that every bin has fewer than h balls is at most $e^{-(2\beta Aa)^{-h}D}$ where $a = I/D$.*

By construction, $a = 4^\ell$ for a v-blob and $a = 1$ for a c-blob. Lemma 14 and the choice of Z immediately imply that,

Corollary 15 *For* fixed $q, E_{a_1}, \ldots, E_{a_h}$, *the probability that $B(q, E_{a_1}, \ldots, E_{a_h})$ occurs is at most*

$$e^{-(2\beta A_c)^{-h} D_c Z} \leq e^{-D_c} \qquad \text{if } q \text{ is a c-blob},$$
$$e^{-(2\beta A_v 4^\ell)^{-h} D_v Z} \leq e^{-D_v} \qquad \text{if } q \text{ is a v-blob}.$$

This probability is with respect to the random mapping $\mathcal{M}$.

We now count the number of bad events. If q is a v-blob then at most $D_v 4^\ell$ canonical paths pass through any answer edge in q. Therefore, an upper bound on the total number of events of the form $B(q, E_{a_1}, \ldots, E_{a_h})$ for a v-blob q is $Q_v \binom{A_v}{h} \binom{D_v 4^\ell}{D_v/(\beta A_v)}^h$ which can be shown to be $e^{o(D_v)}$. A similar calculation shows that an upper bound on the total number of events of the form $B(q, E_{a_1}, \ldots, E_{a_h})$ for a c-blob q is $Q_c \binom{A_c}{h} \binom{D_c}{D_c/(\beta A_c)}^h$ which can be shown to be $e^{o(D_c)}$. By a union bound, the probability that *some* event $B(q, E_{a_1}, \ldots, E_{a_h})$ happens is at most $e^{-D_c} Q_c \binom{A_c}{h} \binom{D_c}{D_c/(\beta A_c)}^h + e^{-D_v} Q_v \binom{A_v}{h} \binom{D_v 4^\ell}{D_v/(\beta A_v)}^h \leq 1/poly(n)$.

Now suppose that under mapping $\mathcal{M}$ no such bad event occurs. For any heavy solution $\mathcal{P}$ in P, by definition we can find a query blob q with h answer edges such that for each such edge $D/(\beta A)$ demands are routed on a canonical path that passes through the edge. Since no bad event occurs, the images of these canonical paths in T create congestion at least h. In other words, the canonical solution $\mathcal{T}(\mathcal{P},\mathcal{M})$ in T has congestion at least h. We have therefore shown,

Lemma 16 *With probability* $1-1/poly(n)$ every *heavy solution $\mathcal{P}$ corresponds to a solution $\mathcal{T}(\mathcal{P},\mathcal{M})$ in which the congestion is at least h in T.*

Lemma 13 states that if $\mathcal{T}$ is a canonical solution then $\mathcal{P}(\mathcal{T},\mathcal{M})$ must be a heavy solution in P. Therefore, by Lemma 16, with high probability over the choice of $\mathcal{M}$, $\mathcal{T}$ must have congestion at least h since $\mathcal{T} = \mathcal{T}(\mathcal{P}(\mathcal{T},\mathcal{M}),\mathcal{M})$. Recall that a solution is canonical if more than $3YR/\beta$ demands are routed along canonical paths. In summary,

Theorem 17 *With probability $1-1/poly(n)$, every canonical solution $\mathcal{T}$ has congestion at least h.*

We now show not many demands can be routed on non-canonical paths. We make use of Property T-2 and a lemma similar to the Erdös-Sachs theorem [14] and show T is almost a high-girth graph. The proof of the following theorem resembles that for the hardness example in [1].

Theorem 18 *With probability $\frac{2}{3}$, the maximum number of demands that can be routed on non-canonical paths is $o(YR/\beta)$.*

Wrapping Up Let the congestion w in EDPwC be $h-1$. For a yes-instance we can route at least YR demands with congestion 1 in T. For a no-instance we have shown in Theorem 17 that at most $3RY/\beta$ demands can be routed along canonical paths with congestion at most w, and in Theorem 18 that $o(RY/\beta)$ demands can be routed along non-canonical paths with congestion at most w. Therefore $\Omega(\beta)$ is the gap between the yes-instance and no-instance. From its definition in (14) we rewrite β as $\beta = \log n^{\frac{\log\alpha}{h-1}-\frac{2\log(h-1)}{\log\log n}}$ using $\ell h = \log\log n$. We denote the exponent of β by b. We now express β in terms of M, the size of the EDPwC instance. From the definitions of M, Y and Z, we have $\log M = \Theta(\log Y)$, $\log Y = Z\sqrt{Z}\cdot o(\sqrt{Z})$, and $Z = (2\beta)^h 8^{\ell h}$. Plugging in the definition of β in (14), $Z = (\log n)^{hb+4}$. Therefore, $\log M < (\log n)^{2(hb+4)+\varepsilon}$ for any constant $\varepsilon > 0$. This implies that the gap $\beta > (\log M)^{\frac{1}{2}\frac{b}{bh+4}-\varepsilon}$, for any constant $\varepsilon > 0$. For any constant congestion w, $h = w+1$ is a constant. Since $\alpha > 1$ is a universal constant, the exponent of $\log M$ is some positive constant as well. Therefore, for any constant congestion, we have derived a polylogarithmic gap.

Note that since $M = n^{\text{polylog } n}$, our randomized reduction can be performed in time $n^{\text{polylog } n}$. We have therefore shown hardness under the assumption that $NP \not\subseteq coRTIME(n^{\text{polylog } n})$. A standard result states that if $NP \subseteq coRTIME(n^{\text{polylog } n})$ then $NP \subseteq ZPTIME(n^{\text{polylog } n})$. In summary,

Theorem 19 *For any constant w, there exists a constant γ_w such that EDPwC with congestion w has no $(\log^{\gamma_w} M)$-approximation unless $NP \subseteq ZPTIME(n^{polylog\ n})$.*

We note that the hardness result directly translates into an integrality gap since for all instances (including no-instances), if we route a $1/7^\ell$ fraction of each demand then the congestion on any edge is at most 1. Therefore, for no-instances we can fractionally route $7^\ell YR/7^\ell$ with congestion 1 but the maximum number of demands that we can route integrally with congestion w is at most $O(YR/\beta)$.

We also note that essentially the same analysis can be used to show hardness of ANFwC. Instead of classifying demands depending on whether or not they are routed on canonical paths, we classify demands depending on whether or not more than a $(1-\frac{1}{h^2})$ of the demand is routed along its canonical path. We omit the details.

3.3 Improvement

We now show how to obtain the improved inapproximability ratio stated in Theorem 1. Note that the value of Z is critical to this ratio. A smaller value of Z gives a smaller value of M and hence a better ratio. The value of Z is set to

$$Z = \max\{(2\beta A_c)^h, (2\beta A_v 4^\ell)^h\} = (2\beta\cdot 8^\ell)^h$$

so that Corollary 15 can ensure that a bad event $B(q, E_{a_1},\ldots,E_{a_h})$ happens with small enough probability. In the following we offer 2 modifications, one is to remove the factor 4^ℓ and the other to reduce the number of answer edges to each query. The first is accomplished by introducing new intermediate nodes ρ with desired properties; and the second is accomplished by introducing multiple provers instead of 2 provers in the proof system.

Modified Construction: ρ-nodes. We first aim to remove the factor 4^ℓ in the definition of Z. Recall that 4^ℓ comes from property P-1: at most 4^ℓ demands from the same query pair but different accepting interactions may share a common answer edge in a v-blob in P. In the following modification we allow some such demands to share a common answer edge in T. (However, we still enforce that demands corresponding to different query pairs, and demands corresponding to the same accepting interaction, go through distinct image edges in T.)

Our modification is accomplished by introducing new intermediate nodes ρ to replace the σ-nodes. Each c-blob $b_{q_c,z}$ at level z in T is surrounded by two groups of ρ-nodes, one to the left and one to the right. The nodes are of the form $\rho_{r,a_c,y,z}$ where r is a query pair that involves query q_c and a_c is a possible answer in an accepting interaction under r. Similarly, each v-blob is surrounded by 2 groups of ρ-nodes one to the left and one to the right. The nodes are of the form $\rho_{r,a_v,y,z}$. See Figure 5 for an illustration.

Consider the ρ-nodes that are to the left of a v-blob $b_{q_v,z}$. We connect these nodes to $b_{q_v,z}$ as follows. Each *fixed* answer a_v to query q_v defines $5^\ell Y$ ρ-nodes of the form $\rho_{r,a_v,y,z}$ to the left of q_v, where $y = 1, \ldots, Y$ and r is a query pair that has a_v as part of an accepting interaction. There are I_v left nodes in $b_{q_v,z}$, where I_v is redefined by $I_v = 5^\ell Y$. We put a *random bipartite matching* between these $5^\ell Y$ ρ-nodes and the left nodes in the blob. To connect $b_{q_v,z}$ to the ρ-nodes to its right, we consider each *fixed* query pair r and each *fixed* possible answer a_v that is part of an accepting interaction under query pair r. This fixed r and fixed a_v define Y right nodes in $b_{q_v,z}$ and Y ρ-nodes to the right of $b_{q_v,z}$. We put a *random bipartite matching* between them. In a similar manner, we connect a c-blob to the ρ-nodes surrounding it.

We now connect two neighboring groups of ρ-nodes. Without loss of generality, let us connect the right group of $b_{q_c,z}$ to the left group of $b_{q_v,z}$. We consider each *fixed* accepting interaction (r, a_c, a_v). At level z, there are Y ρ-nodes of the form $\rho_{r,a_c,y,z}$ and Y of the form $\rho_{r,a_v,y,z}$. We put a *random bipartite matching* between them. Connecting the right group of $b_{q_v,z}$ to the left group of $b_{q_c,z+1}$ is similar.

In this modified graph T the canonical path for demand $d_{r,a_c,a_v,y}$ starts at the source node $\rho_{r,a_c,y,1}$. There is only one edge leaving the source node, which is already defined by a random matching. The demand follows this edge to a left node in $b_{q_c,1}$ (more generally $b_{q_c,z}$ at level z). The demand then reaches a right node in $b_{q_c,z}$ following the random matching within $b_{q_c,z}$. This right node is connected to multiple ρ-nodes to the right of $b_{q_c,z}$. However, only one ρ-node has the correct subscript for demand $d_{r,a_c,a_v,y}$, namely (r, a_c, y_{2z}, z) for some y_{2z} in $[1, Y]$. This is the node where the demand goes. This node is also connected to multiple ρ-nodes to the left of $b_{q_v,z}$. Again, only one has the correct subscript, namely (r, a_v, y'_{2z-1}, z) for some y'_{2z-1} in $[1, Y]$. In a similar manner, the demand goes through the blob $b_{q_v,z}$ until it gets to a ρ-node to the right of the last blob $b_{q_v,Z}$. This last node is the destination node of demand $d_{r,a_c,a_v,y}$.

Further Modification: Multiple Provers. In this section we look at how to reduce the number of answer edges A_c and A_v by introducing multiple provers. Suppose we have λ clause provers and λ variable provers, for some parameter λ. All of the clause provers are sent all of the ℓ clauses in the query. The ith clause prover, where $1 \le i \le \lambda$, is only asked to provide a satisfying assignment for ℓ/λ of them, clauses $(i-1)\ell/\lambda + 1$ through $i\ell/\lambda$. Similarly, all of the variable provers are sent all of the ℓ variables in the query but the ith prover is only asked to provide a satisfying assignment for ℓ/λ of them, variables $(i-1)\ell/\lambda + 1$ through $i\ell/\lambda$. The verifier creates a single clause answer by concatenating all the answers from the λ clause provers and a single variable answer by concatenating all the answers from the λ variable provers. The verifier declares ϕ satisfiable if and only if the concatenated clause answer is consistent with the concatenated variable answer. It is immediate that the error probability is still less than $\alpha^{-\ell}$ for the universal constant α defined in Theorem 11. Each accepting interaction can now be expressed as $(r, a_c^1, \ldots, a_c^\lambda, a_v^1, \ldots, a_v^\lambda)$, where for $1 \le i \le \lambda$, a_c^i represents the answer from the ith clause prover and a_v^i represents the answer from the ith variable prover. Since we now have more than 2 provers we shall refer to r as a *query-sequence* rather than a query-pair.

The graph P is now constructed in a similar manner as before. Each prover has its own blob which contains answers representing satisfying assignment. For example, each clause blob has $7^{\ell/\lambda}$ answers and each variable blob has $2^{\ell/\lambda}$ answer edges. The canonical paths for accepting interactions now pass through 2λ blobs, one for each answer in the accepting interaction.

The graph T is also constructed in a similar manner as before. It has Z levels each of which contains blobs with one-to-one correspondence to those in P. Each blob in T is surrounded by 2 groups of ρ-nodes as described in the previous section. The key feature of the new construction is that the number of answer edges per clause query in P has decreased from 7^ℓ to $7^{\ell/\lambda}$ and the number of answer edges per variable query in P has decreased from 2^ℓ to $2^{\ell/\lambda}$. The edges between the ρ-nodes to the right of $b_{q_c^i,z}$ and the ρ-nodes to the left of $b_{q_c^{i+1},z}$ are defined in a similar manner.

This modified construction allows us to prove the following Theorem. All proofs can be found in [3].

Theorem 20 *For any constant $\varepsilon > 0$ and any congestion $w = o(\log\log M / \log\log\log M)$ there is no $\log^{\frac{1-\varepsilon}{w+1}} M$-approximation algorithm for EDPwC and ANFwC unless $NP \subseteq ZPTIME(n^{polylog\ n})$. For larger congestions $w \le \eta \log\log M/\log\log\log M$ for some constant η, the inapproximability ratios are superconstant.*

Acknowledgements

The second and the third authors thank Madhu Sudan for many useful discussions.

References

[1] M. Andrews and L. Zhang. Hardness of the undirected edge-disjoint paths problem. *Proc. of STOC*, 2005.

[2] M. Andrews and L. Zhang. Hardness of the undirected congestion minimization problem. *Proc. of STOC*, 2005.

[3] M. Andrews and L. Zhang. Hardness of the edge-disjoint paths problem with congestion. http://cm.bell-labs.com/~andrews/pub.html, 2005.

[4] Y. Aumann and Y. Rabani. An $O(\log k)$ approximate min-cut max-flow theorem and approximation algorithm. *SIAM Journal on Computing*, 27(1):291–301, February 1998.

[5] Y. Azar and O. Regev. Strongly Polynomial Algorithms for the Unsplittable Flow Problem. IPCO 2001: 15-29.

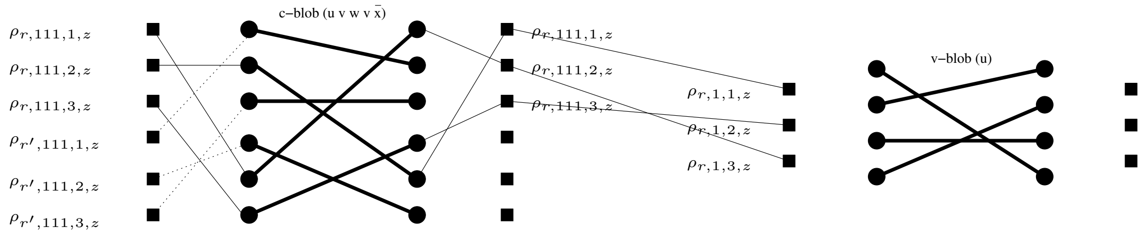

Figure 5. Consider $q_c = u \vee w \vee \bar{x}$ and $q_v = u$. Let $a_c = 111$ representing $u = w = x = 1$ and $a_v = 1$ representing $u = 1$. Let (r, a_c, a_v) be an accepting interaction, and let r' be another query pair that includes a_c in accepting interactions. (Figure not shown all possible query pairs.) For $Y = 3$, the figure shows how we connect q_c to the ρ-nodes surrounding it, and how we connect the right ρ-nodes of q_c to the left ρ-nodes of q_v.

[6] A. Baveja and A. Srinivasan. Approximation algorithms for disjoint paths and related routing and packing problems. *Mathematics of Operations Research*, Vol. 25, pp. 255–280, 2000.

[7] C. Chekuri, S. Khanna, and F. B. Shepherd. The All-or-Nothing Multi-commodity Flow Problem. *Proc. of STOC*, June 2004.

[8] C. Chekuri, S. Khanna, and F. B. Shepherd. Edge Disjoint Paths in Planar Graphs. *Proc. of FOCS*, 2004.

[9] C. Chekuri, S. Khanna, and F. B. Shepherd. Multicommodity Flow, Well-linked Terminals, and Routing Problems *Proc. of STOC*, 2005.

[10] C. Chekuri, S. Khanna, and F. B. Shepherd. An $O(\sqrt{n})$-approximation for EDP in Undirected Graphs and Directed Acyclic Graphs. *Manuscript*, 2005.

[11] C. Chekuri and S. Khanna. Edge Disjoint Paths Revisited. *Proc. of SODA*, 2003.

[12] C. Chekuri, M. Mydlarz, and F. B. Shepherd. Multicommodity Demand Flow in a Tree and Packing Integer Programs. Submitted. Preliminary version in *Proc. of ICALP*, 2003.

[13] J. Chuzhoy and S. Khanna. New hardness results for undirected edge disjoint paths, http://www.cis.upenn.edu/ sanjeev/postscript/edpwchardness.ps.gz 2005.

[14] P. Erdös and H. Sachs. Reguläre graphen gegebener Taillenweite mit minimaler Knotenzahl. *Wiss. Z. Uni. Halle-Wittenburg (Math. Nat.)*, 12:251–257, 1963.

[15] A. Frank. Edge-disjoint paths in planar graphs. *J. of Combinatorial Theory, Ser. B.*, No. 2 (1985), 164-178.

[16] S. Fortune, J. Hopcroft and J. Wyllie. The directed subgraph homeomorphism problem. *Theoretical Computer Science*, Vol. 10, No. 2 (1980), pp. 111–121.

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

[18] N. Garg, V. Vazirani, M. Yannakakis. Primal-Dual Approximation Algorithms for Integral Flow and Multicut in Trees. *Algorithmica*, 18(1):3-20, 1997. Preliminary version appeared in *Proc. of ICALP*, 1993.

[19] V. Guruswami, S. Khanna, R. Rajaraman, F. B. Shepherd, and M. Yannakakis. Near-Optimal Hardness Results and Approximation Algorithms for Edge-Disjoint Paths and Related Problems. To appear in *JCSS*. Preliminary version appeared in *Proc. of STOC*, 1999.

[20] V. Guruswami and K. Talwar. Hardness of low congestion routing in undirected graphs. *Manuscript*, 2005.

[21] J. Hastad, A. Wigderson. Simple Analysis of Graph Tests for Linearity and PCP. Random Structures and Algorithms, Vol 22, no. 2, pp 139-160, 2003.

[22] J. M. Kleinberg and É. Tardos. Approximations for the disjoint paths problem in high-diameter planar networks. *Journal of Computer and System Sciences*, 57:61–73, 1998. Preliminary version in the *Proc. of STOC*, 1995.

[23] J. M. Kleinberg and É. Tardos. Disjoint Paths in Densely Embedded Graphs. *Proc. of FOCS*, pp. 52–61, 1995.

[24] J. M. Kleinberg. Approximation algorithms for disjoint paths problems. PhD thesis, MIT, Cambridge, MA, May 1996.

[25] S. G. Kolliopoulos and C. Stein. Approximating Disjoint-Path Problems Using Greedy Algorithms and Packing Integer Programs. IPCO 1998: 153-168.

[26] P. Raghavan and C. D. Thompson. Randomized rounding: A technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7:365–374, 1987.

[27] R. Raz. A parallel repetition theorem. *SIAM Journal on Computing*, 27(3):763–803, 1998.

[28] N. Robertson and P. D. Seymour. Outline of a disjoint paths algorithm. In B. Korte, L. Lovász, H. J. Prömel, and A. Schrijver, Eds., *Paths, Flows and VLSI-Layout*. Springer-Verlag, Berlin, 1990.

[29] A. Samorodnitsky and L. Trevisan. A PCP characterization of NP with optimal amortized query complexity. In *Proceedings of the 32nd Annual ACM Symposium on theory of Computing*, 2000.

[30] A. Srinivasan. Improved approximations for edge-disjoint paths, unsplittable flow, and related routing problems. *Proc. of the FOCS*, pp. 416–425, 1997.

[31] K. Varadarajan and G. Venkataraman. Graph Decomposition and a Greedy Algorithm for Edge-disjoint Paths. *Proc. of SODA*, 2004.